IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 16, 2020, accepted September 27, 2020, date of publication October 1, 2020, date of current version October 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3028097

Detection of Lying Electrical Vehicles in Charging Coordination Using Deep Learning

AHMED A. SHAFEE[1], MOSTAFA M. FOUDA[2,3], (Senior Member, IEEE), MOHAMED M. E. A. MAHMOUD[1], (Senior Member, IEEE), ABDULAH JEZA ALJOHANI[4,5], (Member, IEEE), WALEED ALASMARY[6], (Senior Member, IEEE), AND FATHI AMSAAD[7], (Member, IEEE)

[1]Department of Electrical and Computer Engineering, Tennessee Tech University, Cookeville, TN 38505, USA
[2]Department of Electrical and Computer Engineering, College of Science and Engineering, Idaho State University, Pocatello, ID 83209, USA
[3]Department of Electrical Engineering, Faculty of Engineering at Shoubra, Benha University, Cairo 11629, Egypt
[4]Department of Electrical and Computer Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[5]Center of Excellence in Intelligent Engineering Systems (CEIES), King Abdulaziz University, Jeddah 21589, Saudi Arabia
[6]Department of Computer Engineering, Umm Al-Qura University, Makkah 24381, Saudi Arabia
[7]School of Information Security and Applied Computing (SISAC), Eastern Michigan University (EMU), Ypsilanti, MI 48197, USA

Corresponding author: Ahmed A. Shafee (aashafee42@tntech.edu)

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under Grant DF-181-135-1441.

ABSTRACT Because charging coordination is a solution for avoiding grid instability by prioritizing charging requests, electric vehicles may lie and send false data to illegally receive higher charging priorities. In this article, we first study the impact of such attacks on both the lying and honest electric vehicles. Our evaluations indicate that lying electric vehicles have a higher chance of charging, whereas honest electric vehicles may not be able to charge or may charge late. Then, an anomaly-based detector based on a deep neural network is devised to identify lying electric vehicles. The idea is that since each electric vehicle driver has a particular driving pattern, the data reported by the corresponding electric vehicle should follow this pattern, and any deviation due to reporting false data can be detected. To train the detector, we first create an honest dataset for the charging coordination application using real driving traces and information provided by an electric vehicle manufacturer, and we then propose a number of attacks as a basis for creating malicious data. We train and evaluate a gated recurrent unit model using this dataset. Our evaluations indicate that our detector can detect lying electric vehicles with high accuracy and a low false alarm rate even when tested on attacks that are not represented in the training dataset.

INDEX TERMS Security, false data injection, charging coordination, electric vehicles, smart grid.

NOMENCLATURE

ADASYN	Adaptive synthetic sampling.
AUC	Area under the curve.
CC	Charging coordinator.
DNN	Deep neural network.
EV	Electric vehicle.
GA	Genetic algorithm.
GRU	Gated recurrent unit.
MLP	Multilayer perceptron.
NSGA	Non-dominated Sorting Genetic Algorithm.
RNN	Recurrent neural network.
ROC	Receiver operating characteristic.
SMOTE	Synthetic Minority Over-sampling Technique.
SoC	State of charge.
TCC	Time to complete charge.

The associate editor coordinating the review of this manuscript and approving it for publication was Salvatore Favuzza.

I. INTRODUCTION

Recently, there has been growing interest in adopting more green transportation in an effort to reduce the associated carbon emissions and reduce the dependency on crude oil. One promising possibility is to replace gasoline-powered vehicles with electric vehicles (EVs). In recent years, the number of EVs on the road has been experiencing a dramatic increase [1]. Another important factor promoting the widespread use of EVs is the recent trend of adopting more renewable energy sources on the consumer side by installing solar cells on rooftops. Due to the intermittent nature of these sources, EVs are advantageous as a way to store excess energy. Nevertheless, despite the numerous advantages offered by EVs, several challenges also need to be addressed regarding the deployment of a large number of EVs.

The uncoordinated and simultaneous charging of EVs may stress the power system and, in severe cases, destabilize the grid [2], [3]. Accordingly, charging coordination mechanisms have been developed to avoid this problem by balancing the charging load and power supply [4]. The idea is that each EV should report data (such as the state of charge (SoC) of its battery) to a charging coordinator (CC) that then uses these data to prioritize the EVs' charging requests, allowing EVs with high-priority requests (typically those with low SoC values) to charge in the current time slot without exceeding the maximum charging energy capacity and deferring other requests to future time slots [5]. Several previous papers in the literature have presented charging coordination mechanisms [6]–[8]. However, these mechanisms assume *that the electric vehicles report correct data; unfortunately, this assumption cannot be guaranteed to be satisfied because electric vehicles are motivated to report false data* (i.e., lower state of charge values) to guarantee that they can charge before their charging requests expire.

In this article, we first evaluate the impact of lying EVs that report false data on charging coordination. Specifically, we focus on evaluating the gains achieved by the lying EVs and the harm they cause to honest EVs. Our evaluations confirm that reporting false data is beneficial for EVs because they are able to charge sooner, whereas honest EVs are harmed by being unable to charge or being required to charge late. To resolve this issue, we devise a machine learning model to detect lying EVs. The idea is that each EV should report its battery SoC periodically, e.g., every 30 minutes, and the reported data can then be used both for load prediction (for energy management) and for the detection of lying EVs. Since each EV driver has a particular driving pattern, the SoC values of the corresponding EV should follow this pattern, and deviations from this pattern due to false SoC reports can be detected through machine learning.

To train the model, we first create a dataset for the charging coordination application. To do so, we use the real driving traces of vehicles provided in [9] and real EV information provided by the automotive manufacturer Kia in [10]. Then, we propose several attacks in which EVs report false SoC values to the charging coordination mechanism. To balance the data, we use adaptive synthetic sampling (ADASYN) [11] to compensate the number of samples in the minority class to be equal, or nearly equal, to the number of samples in the majority class. Finally, we use the created dataset to develop a machine learning model to detect lying EVs.

Deep learning has been used in many applications, such as facial recognition, intrusion detection, and speech analysis, because of its high accuracy [12]. In this article, a deep recurrent neural network (RNN) [13] is selected as the machine learning model because it is suitable for handling the time series nature of our dataset. However, choosing the best set of hyperparameters for the deep neural network is a hard optimization problem. Therefore, we use a version of a genetic optimization [14] technique to find the best architecture for our detector. Extensive experiments conducted to evaluate our detector are reported. Our evaluations confirm that our RNN detector can identify lying EVs with high accuracy and a low false alarm rate. To evaluate how considering the time series nature of our data can improve accuracy, we compare our detector to a multilayer perceptron (MLP) [15] detector, and our evaluations indicate that our detector outperforms the MLP detector. Tests of our detector on attacks that are not represented in the training dataset show that it can also detect such unseen attacks with high accuracy.

This article makes the following main contributions to the literature:

1) We investigate the impacts of EVs reporting false data on the charging coordination mechanism. Then, we create a new dataset that contains both benign and malicious data for charging coordination application.
2) We propose a deep learning model for identifying lying EVs and present extensive experiments conducted to evaluate this model.

The remainder of this article is organized as follows. The related work is discussed in Section II. Section III describes the system model. The evaluation of the impacts of lying EVs on the charging coordination mechanism is presented in Section IV. The dataset created for the charging coordination application is presented in Section V. The proposed deep-learning-based detector is discussed in detail in Section VI. Our evaluations of the proposed detector and experimental results are discussed in Section VII. Our conclusions are presented in Section VIII.

II. RELATED WORK

In this section, we first present the existing works on charging coordination mechanisms. Then, we discuss works that address the uncertainty of EV charging. Finally, we discuss the existing datasets for EVs.

A. CHARGING COORDINATION MECHANISMS

Several works in the literature have investigated charging coordination mechanisms for EVs. Arias *et al.* [16] proposed an optimized charging coordination mechanism for finding the best charging schedule that satisfies the EVs' requirements while respecting the operational capacity of the electrical distribution system. The authors considered three optimization algorithms: tabu search, a greedy randomized adaptive search procedure, and a new hybrid optimization algorithm that combines these two algorithms.

Hajforoosh *et al.* [17] proposed a charging coordination mechanism based on the fuzzy discrete particle swarm optimization algorithm and a fuzzy genetic algorithm. The main objective of the proposed mechanism is to maximize the amount of electrical power delivered to the EVs while minimizing grid losses, distribution transformer loading and the cost associated with energy generation.

Franco *et al.* [18] proposed a charging coordination mechanism for unbalanced electrical distribution systems using a mixed integer linear programming technique. The proposed mechanism aims to reduce the energy cost of EV charging by considering several factors, such as the three-phase circuit representation and the load imbalance, to produce a nonlinear programming model. This model can be converted into a linear model using linearization techniques to allow it to be easily solved.

Mahmoud *et al.* [19] introduced a privacy-aware charging coordination mechanism that preserves the privacy of EV drivers while still optimizing the power supplied to the EVs without exceeding the total charging capacity. The idea is that each EV encrypts its charging request in addition to a one-time secret key using the public key of the CC so that only the CC can decrypt the request. Then, the CC uses the provided data to compute the charging schedules and encrypts them using the secret keys sent by the EVs. Additionally, random noise is added to the data sent by the EVs to prevent the ability to link different charging requests sent from the same EV to preserve privacy. In addition, a modified version of the knapsack optimization algorithm is used to schedule the charging of the EVs.

Baza *et al.* [5] introduced a decentralized charging coordination mechanism using blockchain and smart contract technology. Baza *et al.* [20] proposed two privacy-preserving charging coordination mechanisms. The first mechanism has a centralized architecture and uses a blind signature cryptosystem for anonymous authentication. The second mechanism has a decentralized architecture in which the various EVs run the mechanism in a distributed manner. The idea is that one EV is selected to act as a head node that decrypts the EVs' aggregated charging demand and broadcasts the aggregated demand to the EVs without being able to learn the data of the individual EVs to preserve privacy. Then, each EV can compute its own charging schedule such that the maximum charging energy capacity is not exceeded.

To the best of our knowledge, *the existing works in the literature assume that the electric vehicles report correct data* to the charging coordination mechanism. However, this assumption cannot be guaranteed to be satisfied because the *electric vehicles are motivated to report false data to guarantee that they can charge* before their charging requests expire. In addition, *no existing work has studied the impact of reporting false data on the charging coordination mechanism or proposed a solution for identifying electric vehicles that report incorrect data.*

B. UNCERTAINTY OF EV CHARGING

Numerous works in the literature have investigated the uncertainties in EV charging applications. Xu and Chung [21] identified the main uncertainties encountered in the charging of EVs, including the rounding of time units, charging component failure, punctuality, errors in energy consumption forecasting, aggregator failure, EV absence, and grid realization. Then, techniques for incorporating these uncertainties into the system were introduced.

Ghosh and Aggarwal [22] proposed a pricing mechanism for EV charging under uncertainty in EV arrival. The idea is that the charging stations provide contracts specifying the amount of energy, price, time to charge, etc. Then, the EV driver selects the best option for charging. The authors attempted to solve two optimization problems to maximize both the profit of the charging stations and the social welfare achieved.

Pradhan *et al.* [23] used information gap decision theory to address price uncertainties that could arise at a charging node. The proposed mechanism guarantees a minimum amount of profit in the case of fluctuations in the charging price.

These mechanisms *aim to resolve the issues presented by various uncertainties* in the charging application, such as an electric vehicle that does not charge as planned or charges less power than it requested, *but they cannot solve our problem, in which some electric vehicles report false state of charge values* to gain higher charging priorities than honest EVs.

C. DATASETS FOR EVs

Akhavan-Hejazi *et al.* [24] created a dataset for hybrid EVs (powered by both electricity and gasoline) that contains minute-by-minute SoC data as well as the nodal and temporal charging loads of plug-in hybrid EVs. The dataset was created using driving traces from 536 gasoline-powered taxi vehicles equipped with Global Positioning System (GPS) devices in San Francisco and the nominal operation data of various plug-in hybrid EV brands. The main goal was to find the SoC and charging patterns of vehicles with the same movement patterns but equipped with various plug-in hybrid EV technologies. As these vehicles are moving, they switch between using electrical energy and gasoline. The dataset considers two types of plug-in hybrid EVs. The first is the charge depleting type, which switches to gasoline once the electric battery is depleted. The second is the charge blending type, which may switch to the gasoline engine to increase the torque of the vehicle even if the electric battery is not depleted.

Wang [25] proposed a model for predicting the amount of energy stored in the EVs in a specific area. A neural network model combined with a linear chain conditional random field was used to create the model. The dataset used to train the neural network was created using the GPS trajectories of a set of taxis in Beijing. In that paper, it was assumed that the SoC of each EV varies linearly with respect to the distance driven. However, the linear model used to compute the SoC values was not described. The paper also assumed that all EVs recharge to full capacity at the beginning of each day, which may not be guaranteed in practice. Moreover, the dataset has not been shared publicly.

Oh *et al.* [26] provided a dataset for vehicles including gasoline-powered vehicles, hybrid EVs, plug-in hybrid EVs, and EVs. The dataset was collected based on real-world

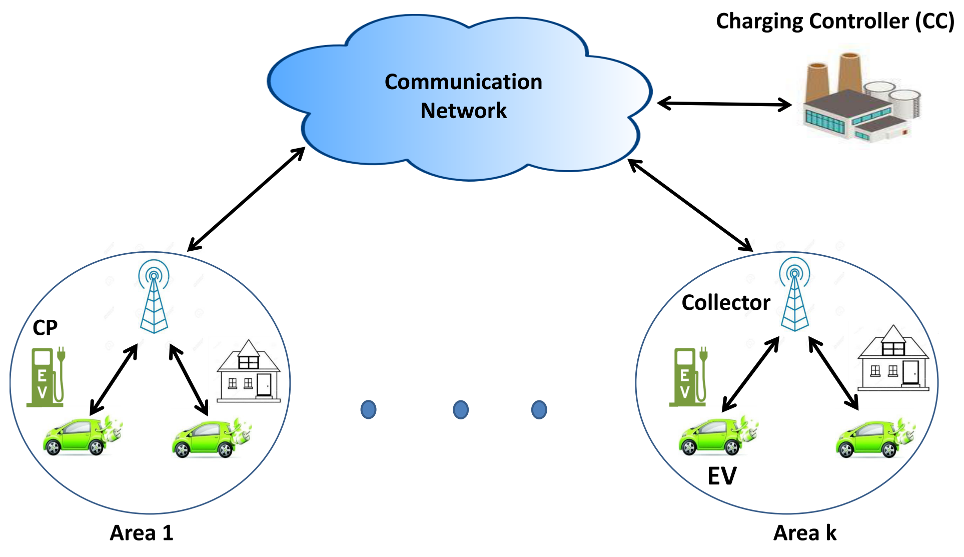


FIGURE 1. The considered system model.

driving scenarios for a set of vehicles, including driving in various environments, from highways to traffic-dense regions. The information in this dataset is divided into static and dynamic information. The static information consists of the vehicle parameters, such as the vehicle type and engine configuration. The dynamic information consists of time-stamped driving records for each vehicle, such as the vehicle's speed, battery current, voltage, and SoC. However, this dataset contains the data for only three EVs; thus, it is a very small dataset for training machine learning models.

In contrast to these works, in this article, we create *a new dataset for the charging coordination application, including data for lying electric vehicles, by proposing various attacks as the basis for creating malicious data.*

III. NETWORK AND THREAT MODELS

As shown in Fig. 1, the considered system model consists of the following entities:

- Charging coordinator (CC): The CC receives charging requests from the EVs, computes the charging schedules and sends them to the EVs. It must be ensured that the scheduled charging amounts do not exceed the available charging energy.
- Electric vehicles (EVs): The EVs send charging requests containing information such as the battery SoC to the CC, which uses this information to compute the charging schedules for the EVs.
- Collectors: A collector is responsible for collecting the charging requests of the EVs in a given area and sending them to the CC. It also receives charging schedules from the CC and forwards them to the EVs in its area.
- Charging points (CPs): The CPs provide the EVs with the needed electrical power for charging.

As shown in Fig. 1, the charging coordination mechanism is executed by the CC for each geographic area, where each area is usually connected to one electrical bus. The total amount of charging energy for the EVs in each area should not exceed the maximum charging energy capacity that is known to the CC. Time is divided into slots, e.g., 30 minutes per slot, and the charging coordination mechanism is run at the beginning of each time slot. When an EV needs to charge, it sends a charging request to the CC; this request includes information such as the battery SoC and the time to complete charge (TCC). The SoC is the amount of energy stored in the battery, represented as a number between zero and one, where $\text{SoC} = 1$ when the battery is fully charged and $\text{SoC} = 0$ when the battery is fully depleted. The TCC is the expiration time of the charging request, i.e., the EV needs to charge before the TCC has elapsed. After the CC has received the charging requests from the EVs in a certain area, if the total charging demand does not exceed the maximum charging energy capacity, all EVs can charge. Otherwise, the CC must execute a charging coordination mechanism to select a subset of the EVs to charge in the current time slot and defer the charging of the other EVs to later time slots. This mechanism should prioritize all requests and select the highest-priority requests for charging such that the charging energy limit is not exceeded. Typically, higher priority is given to EVs with lower SoC values. In the literature, various approaches for selecting the subset of EVs that can charge in the current time slot have been presented [16]–[18].

Regarding the threat model, we focus on attacks launched by lying EVs that send false SoC values in their charging requests to illegally gain higher priorities for charging. In this article, we propose various ways in which EVs can report false SoC values. We also report the training of a machine learning detector that uses the SoC values reported by an EV to classify the EV as honest or lying.

IV. EVALUATION OF THE IMPACT OF LYING EVs

In this section, we present multiple experiments conducted to evaluate the impact of falsely reported SoC values on the charging coordination mechanism. Specifically, we are interested in investigating how such attacks (1) benefit the lying EVs and (2) harm honest EVs.

A. SETUP AND METRICS

We use MATLAB [27] to simulate the following charging coordination scenario. Consider a set of n EVs ($\mathbb{E} = \{\mathrm{EV}_1, \mathrm{EV}_2, \ldots, \mathrm{EV}_n\}$), a set of days ($\mathbb{D} = \{1, \ldots, d\}$), and a set of time slots of equal length ($\mathbb{T} = \{1, 2, \ldots, T\}$). For $\mathrm{EV}_i \in \mathbb{E}$, the SoC value on day $d \in \mathbb{D}$ at time $t \in \mathbb{T}$ is denoted by $S_i(d, t)$. At time t, EV_i sends a charging request that contains information such as the SoC of the EV ($S_i(d, t)$) to the CC. A charging coordination mechanism that takes the EVs' charging requests as input is executed to determine which EVs can charge in the current time slot. However, a lying EV reports a false SoC value (i.e., a lower value), denoted by $\beta * S_i(d, t)$, where $0 \leq \beta < 1$, to the CC to illegally gain a higher charging priority to ensure that it can charge before its request expires.

In our experiments, we assume that the maximum capacity of each EV's battery is 200 units of energy and that the total number of EVs sending charging requests in each time slot is 100, including both lying and honest EVs. Each simulation is run for a total of 30 time slots. In addition, simulations are run using two maximum charging energy capacities per time slot, namely, 800 and 1500 units of energy, to evaluate the impact of falsely reported SoC values on the charging coordination mechanism for two different levels of the available charging supply. Initially, the SoC value of each EV is set to a random number between 0.25 and 0.75, and the TCC is equal to four time slots.

In our experiments, we apply the charging coordination mechanism proposed in [19], which is based on the knapsack algorithm, as described in Algorithm 1. The algorithm needs to calculate a priority index (PI) for each charging request; this index takes values between 0 and 1, and the priority of a request increases as PI increases. PI is calculated as follows:

$$\mathrm{PI} = \varepsilon f_1(\mathrm{SoC}) + (1 - \varepsilon) f_2(\mathrm{TCC}),$$

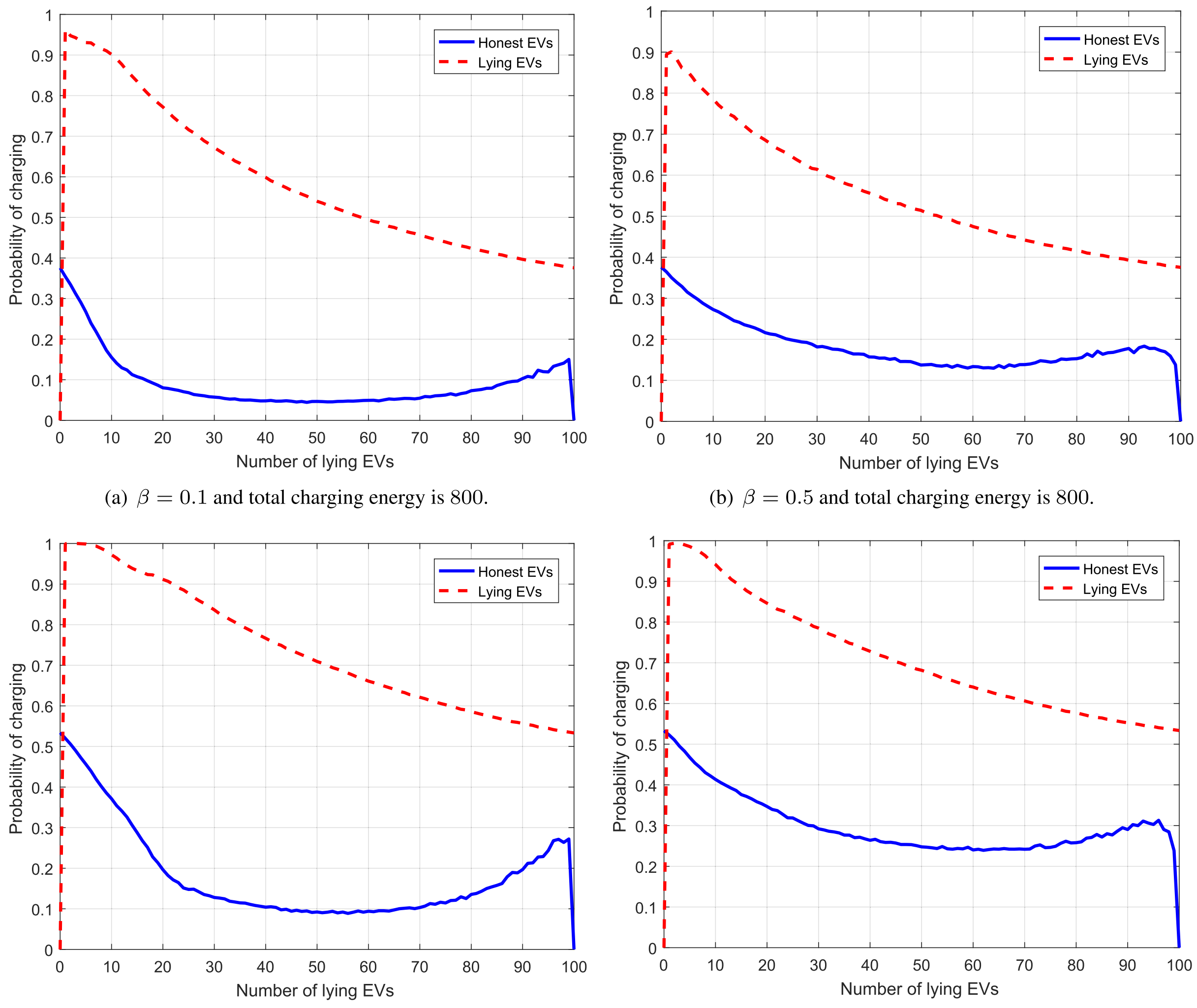
where ε, which takes values between 0 and 1, determines the relative weights given to the SoC and TCC and $f_1()$ and $f_2()$ are two functions that map the SoC and TCC, respectively, to values between 0 and 1. We assume that ε is 0.6, that $f_2(x)$ is equal to 0.4 when $0 < x \leq 4$ and is equal to 0.2 otherwise, and that $f_1(x) = 1 - x$. In addition, the SoC reported by an honest EV_i is $S_i(d, t)$, while the SoC reported by a lying EV_i is $\beta * S_i(d, t)$, where $0 \leq \beta < 1$ and $S_i(d, t)$ is the true SoC value. As shown in line 10 of Algorithm 1, the CC divides the priority index of each vehicle (PI_i) by its energy demand (P_i) and then selects the EVs with the highest ratios for charging such that the maximum charging energy capacity (C_{sys}) is not exceeded. We assume that all EVs need to fully charge

Algorithm 1 Charging Coordination Mechanism Used in [19]

```
1  function Charging_Coordination (SoC, TCC, P, EnergyCapacity)
       // $P_i$: the charging amount requested by
       //     $EV_i$
       // $X$: the amount of energy each EV can
       //     charge in this time slot
2      $C_{sys} \leftarrow$ EnergyCapacity
3      $\varepsilon \leftarrow 0.6$
4      for $i \in X$
5          $X_i \leftarrow 0$
6      end
       // PI: compute the priorities of the EVs
7      for $i \in$ PI
8          $\mathrm{PI}_i \leftarrow \varepsilon f_1(\mathrm{SoC}_i) + (1 - \varepsilon) f_2(\mathrm{TCC}_i)$
9      end
10     $A \leftarrow \mathrm{PI}_1/P_1 \geq \mathrm{PI}_2/P_2 \cdots \geq \mathrm{PI}_n/P_n$
11     for $i \in A$
12         if $P_i \leq C_{sys}$
13             $X_i \leftarrow P_i$
14             $C_{sys} \leftarrow C_{sys} - P_i$
15             $A \leftarrow A - i$
16         end
17     end
       // If there is any remaining power, provide
       // it to the EV with the highest priority
18     $L \leftarrow argmax_A \mathrm{PI}$
19     $X_L \leftarrow C_{sys}$
20     Return X
```

their batteries, i.e., to charge until the SoC is 1; therefore, $P_i = (1 - \mathrm{SoC}) \times \mathrm{Battery\_Capacity}$.

The key metrics used to evaluate the impact of EVs reporting false SoC values on the charging coordination mechanism are as follows. The first metric is the *probability that a lying electric vehicle will be charged* before the expiration of its charging request. The second metric is the *probability that an honest electric vehicle will be able to charge* before the expiration of its charging request in the presence of lying EVs. In the experiments, we vary β to study different cheating behaviors, where as β decreases, the SoC value reported by a lying EV is lower than the real value to a greater degree.



(a) $\beta = 0.1$ and total charging energy is 800.
(b) $\beta = 0.5$ and total charging energy is 800.
(c) $\beta = 0.1$ and total charging energy is 1500.
(d) $\beta = 0.5$ and total charging energy is 1500.

FIGURE 2. The charging selection probabilities for honest and lying EVs under different values of β and the total charging energy.

B. SIMULATION RESULTS

Fig. 2 shows the selection probabilities for honest and lying EVs under different values of β and the total charging energy. The probability that a lying EV will be selected for charging is calculated as the number of lying EVs that can fully charge before their charging requests expire divided by the total number of lying EVs. Similarly, the probability that an honest EV will be selected for charging is calculated as the total number of honest EVs that are selected for charging before their charging requests expire divided by the total number of honest EVs.

The figure shows that as the number of lying EVs increases, the selection probability for honest EVs decreases. This is because more lying EVs are selected for charging due to their higher priorities. With a certain number of lying EVs, the selection probability for honest EVs becomes very low. This is because in this case, very few honest EVs are selected for charging, whereas most of the lying EVs are selected. For instance, when β is 0.1, the total charging energy is 800, and the number of lying EVs is 50, the selection probability for honest EVs is approximately 0.05, whereas the lying EVs have a much higher probability of being selected because they have higher priorities.

It can also be seen that the selection probability for honest EVs eventually increases again at very high numbers of lying EVs, i.e., low numbers of honest EVs. This is not because more honest EVs are selected but rather because the total number of honest EVs decreases as the number of lying EVs increases, and thus, the selection of only a few honest EVs for charging substantially increases the selection probability for honest EVs. Note that the total number of EVs (including both honest and lying EVs) is always 100. Moreover, as β decreases, fewer honest EVs are selected for charging. For instance, in Figs. 2(a) and 2(b), where β is 0.1 and 0.5, respectively, when the number of lying EVs is 50, the selection probabilities for honest EVs are 0.05 and 0.15, respectively.

For lying EVs, the selection probability is consistently high until the number of lying EVs reaches a certain value, after which the probability starts to decrease. This behavior can be explained by the fact that when the system contains many lying EVs, some of the lying EVs may not be selected for charging before their charging requests expire.

Moreover, by comparing Figs. 2(a), 2(c), 2(b) and 2(d), it can be seen that when the total charging energy is reduced from 1500 to 800, fewer total EVs are selected for charging in each time slot. This results in the selection of fewer honest EVs and also reduces the selection probability for lying EVs because of the smaller total number of selected EVs. Therefore, lying EVs have a more severe impact on honest EVs as the available charging energy decreases.

Based on the above results, we can draw the following conclusions. (1) Reporting lower SoC values is beneficial for EVs because it can significantly increase their chance of charging before their requests expire. (2) Honest EVs have a very low chance of charging before their charging requests expire, especially as the total charging capacity decreases, as the number of lying EVs increases, and as the lying EVs report lower SoC values (i.e., as β decreases).

V. DATASET

In this section, we explain how we created a dataset for the charging coordination application using real driving traces of vehicles and technical information released by an EV manufacturer. Although our focus was to create a dataset for use in training a detector to identify lying EVs, our dataset can also be used for other applications, such as energy demand prediction for energy management applications.

A. HONEST DATASET CREATION

To create the honest dataset, we used the driving routes of 536 taxis in San Francisco, CA, released in [9]. Data recording began on May 17, 2008, and finished on June 10, 2008. For each taxi, each row of data contains a time stamp, latitude, and longitude. In our dataset, we consider the Kia Soul EV [10]. Therefore, in addition to using the real routes in [9], we also used information released by Kia. This information is summarized in Table 1. The EV uses a charging technique in which the grid provides the EV with the required amount of electrical power using alternating current (AC), which is converted into direct current (DC) by the EV's own rectifier (converter) [10].

To compute the SoC every minute for each EV, we randomly initialized the SoC value and then, for every minute, checked whether the EV was moving. If it was moving, then we first computed the distance traveled using the EV speed and the time, subsequently computed the power consumption of the EV by multiplying this distance by the power consumption rate given in Table 1, and finally updated the SoC value by subtracting the ratio of the power consumed to the maximum battery capacity from the current SoC value. In other words, the SoC is linearly dependent on the distance driven by the EV [25] and is computed as

TABLE 1. Operational data of the Kia Soul EV.

Brand	Kia
Model	Soul
Battery Capacity (kWh)	64
Avg. Electric Range (mi)	230
Max. Charge Rate (kW)	7.2
Power Consumption Rate (Wh/mi)	275

$$\text{NewSoC} = \text{OldSoC} - \frac{\text{ConsumptionRate} \times \text{ Distance}}{\text{MaxCapacity}},$$

where the amount of power consumed is calculated by multiplying the distance traveled by the EV in miles by the EV's power consumption rate. If the EV was not moving, then we checked whether it was charging. If it was charging, then the SoC value was updated by adding the ratio of the amount of the power drawn to the EV's capacity. If the EV was not charging, then the SoC value was left unchanged.

In this way, a benign dataset was created for each of the 536 EVs to describe its normal behavior (i.e., driving pattern). In the dataset, the behavior of each EV is captured by 24 rows representing the normal behavior of that EV over 24 days. Each row contains T features, representing the SoC values reported every τ minutes by that EV. For the construction of the dataset, τ was selected to be 30 minutes, resulting in a set of $T = 48$ features in each row. Thus, the benign dataset contains a total of 12, 864 data samples.

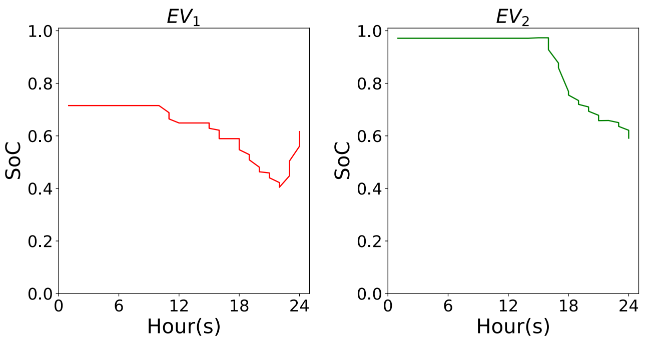


FIGURE 3. The SoC values of two EVs from the dataset throughout one day.

Fig. 3 shows the SoC values of two EVs throughout one day. As observed from the plots, the initial SoC values are different, and the SoC values begin to decrease at a certain hour, which indicates that the EVs are moving; at certain times, the SoC values also increase, indicating that the EVs are charging at these times.

Fig. 4 shows the average SoC values of the same two EVs over 24 days. These plots show that each of the two EVs

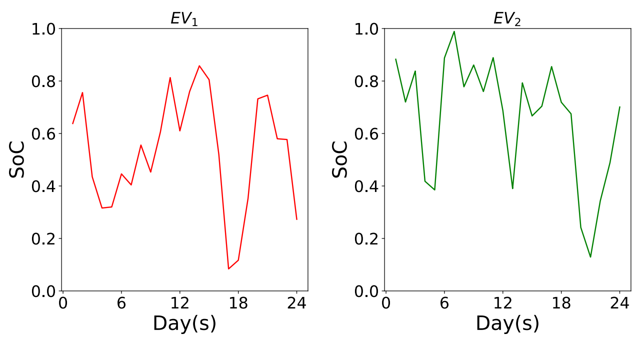


FIGURE 4. **The average SoC values of two EVs from the dataset over 24 days.**

has a different SoC pattern corresponding to charging vs. not charging and moving vs. remaining still. This observation suggests that *each electric vehicle has its own normal behavior, reflected by its state of charge pattern; accordingly, our detector will attempt to detect electric vehicles that appear to deviate from their normal behavior and label them as liars.*

TABLE 2. **Attacks for reporting a false SoC.**

Attack no.	Attack type	Formula
Attack 1	Partial reduction attack	$RS_i(d,t) = \alpha S_i(d,t)$
Attack 2		$RS_i(d,t) = \beta_i(d,t) S_i(d,t)$
Attack 3	Selective time filtering attack	$RS_i(d,t) = \begin{cases} 0 & t_b \leq t \leq t_e \\ S_i(d,t) & else \end{cases}$
Attack 4		$RS_i(d,t) = \begin{cases} \beta_i(d,t) S_i(d,t_b) & t_b \leq t \leq t_e \\ S_i(d,t) & else \end{cases}$

B. MALICIOUS DATASET CREATION

For the creation of the malicious dataset, various attack scenarios were formulated regarding the reporting of false SoC values to the CC. These attacks are summarized in Table 2. Four types of attacks are considered: two versions of a partial reduction attack and two versions of a selective time filtering attack [28]. The SoC of EV$_i$ at time t on day d is denoted by $S_i(d, t)$, and the SoC reported by EV$_i$ to the CC is denoted by RS$_i(d, t)$. If EV$_i$ is honest, then RS$_i(d, t)$ is equal to $S_i(d, t)$ at all times.

As illustrated in Table 2, in *Attack* 1, the attacker attempts to deceive the CC by multiplying the correct SoC value by a constant value α of less than one to report a lower SoC in order to gain a higher charging priority. In *Attack* 2, the EV attempts to deceive the CC by multiplying the reported SoC value by a time-dependent function $\beta_i(d, t)$ whose value is less than one at all times. In *Attack* 3, the EV uses a selective time filtering technique in which it reports an SoC of zero (or a small value) during an interval $t_b \leq t \leq t_e$ but reports its correct SoC ($S_i(d, t)$) otherwise. *Attack* 4 is similar to *Attack* 3 except that EV$_i$ selects a specific time interval $t_b \leq t \leq t_e$ in which to report SoC values that are multiplied by a time-dependent function $\beta_i(d, t)$ whose value is always less than one, whereas at other times, EV$_i$ reports its actual SoC ($S_i(d, t)$). These four attacks were applied on each row in the benign dataset to produce a malicious dataset containing $51,456$ $(= 12,864(\textit{benign samples}) \times 4)$ records.

C. DATASET PREPARATION

In this subsection, we discuss the preparation of the dataset used to train and evaluate the detector. As discussed in Section V-A, an honest dataset was created with $12,864$ data samples, while the malicious dataset introduced in Section V-B was created with $51,456$ data samples. The original dataset obtained by combining these two datasets contains $64,320$ data samples. As shown in Table 3, this dataset is divided into three parts: a training dataset for training the detector, a validation dataset for avoiding overfitting during the training process, and a test dataset for evaluating the model. For both the evaluation and testing phases, the two classes are balanced, with $2,000$ samples in each class. However, in the training dataset, the two classes are imbalanced, with the malicious class containing more samples than the honest one.

TABLE 3. **Samples in our dataset.**

	Before augmentation		After augmentation	
	Benign	Malicious	Benign	Malicious
Training	8,864	47,456	47,596	47,456
Validation	2,000	2,000	2,000	2,000
Testing	2,000	2,000	2,000	2,000

The *class imbalance problem* arises when one class in a dataset contains more samples than other classes [29]. In our training dataset, the honest class is the minority class that needs to be balanced with the malicious class. Training a classifier model using imbalanced data can produce a classifier that is biased towards the class (or classes) represented by more samples in the dataset. Therefore, to create accurate classifiers, data balancing techniques should be applied when there are some data distributions that significantly dominate other data distributions in the sample space. One solution to this problem is to use a data augmentation technique to synthetically increase the number of samples in the minority class.

Data augmentation is the process of increasing the number of data samples for the purpose of creating a more accurate and robust model [30]. It is known that as the number of samples in the training dataset increases, the trained model becomes more accurate [31]. The most common data augmentation techniques used to balance a dataset are oversampling techniques. The two most common categories of oversampling techniques [11] are as follows:

- *Sampling techniques:* In these techniques, the imbalanced dataset is compensated by copying some of the samples in the minority class.

- *Synthetic data generation:* In these techniques, the imbalance problem is solved by generating synthetic data samples to balance the original dataset. Common techniques in this category are the Synthetic Minority Over-sampling Technique (SMOTE) [32] and the adaptive synthetic sampling approach (ADASYN) [11].

For our dataset, we used a synthetic data generation technique because sampling techniques can lead to overfitting due to the creation of exact copies of the original data [33]. Specifically, we selected *ADASYN*, which is considered an improved extension of SMOTE [11], for balancing the minority class in the training dataset. The main advantage of ADASYN is that it uses a density distribution to automatically determine the number of synthetic samples that should be generated for each sample in the minority class, in contrast to the SMOTE algorithm, which generates the same number of synthetic samples for each original minority sample. For the application of ADASYN [11] to our dataset, the dataset should contain only two classes: a minority class (the honest class) and a majority class (the malicious class). This is because ADASYN attempts to balance the minority class with the majority class by calculating the ratio between the numbers of samples in the two classes and then adding synthetic samples to the minority class until the number of minority class samples approximately reaches the number of majority class samples. As indicated in Table 3, the ADASYN data augmentation technique was applied only to the training dataset, resulting in a total of $95,052$ data samples, with $47,596$ being benign samples and $47,456$ being malicious samples. The ADASYN algorithm can be summarized as consisting of the following steps:

1) The degree of class imbalance in the dataset is computed:

$$R = \frac{m_{\min}}{m_{\text{maj}}},$$

where $m_{\min}$ and m_{maj} are the numbers of samples in the minority and majority classes, respectively, and $R \in (0, 1]$.

2) If $R < R_{th}$, where R_{th} is the maximum tolerance threshold for the class imbalance ratio, then the following steps are performed:
   - The total number of synthetic minority data samples that need to be generated by ADASYN is calculated as

   $$G = (m_{\text{maj}} - m_{\min})\xi,$$

   where $\xi \in [0, 1]$ is a parameter used to specify the required degree of balance after synthetic data generation. When $\xi = 1$, this indicates that the dataset is required to be fully balanced when the generation process is complete.
   - For each minority sample, the k nearest neighbors are considered to calculate the r value:

   $$r_i = \frac{\partial_i}{k},$$

   where ∂_i is the number of samples belonging to the majority class among the k nearest neighbors of sample i based on the Euclidean distance rule in n-dimensional space. The r_i value indicates the dominance of the majority class in the neighborhood.
   - Before the number of synthetic samples to be generated is finally determined, the r_i values for all minority samples are normalized such that their sum is one:

   $$\hat{r}_i = \frac{r_i}{\sum r_i},$$

   where $\sum \hat{r}_i = 1.$
   - The number of synthetic samples to be generated in each neighborhood is calculated as follows:

   $$g_i = \hat{r}_i * G.$$

   Since g_i is calculated using the corresponding r_i value for every minority class sample, ADASYN creates more synthetic samples in neighborhoods where the ratio of majority to minority observations is greater.
   - For each sample x_i in the minority class, g_i samples are generated by looping from 1 to g_i and doing the following:
     - A data sample x_j is randomly chosen from among the k nearest neighbors of data sample x_i.
     - A synthetic observation is generated:

     $$s_i = x_i + (x_j - x_i)\lambda,$$

     where s_i is the generated synthetic sample, x_i is sample i from the minority class, and λ is a random number between 0 and 1.

VI. PROPOSED LYING EV DETECTOR

In this section, we discuss the details of the proposed lying EV detector. An abstract representation of the operations needed to develop the proposed detector is shown in Fig. 5. The proposed detector is based on a deep neural network (DNN) classifier, and its development process consists of two stages. The *first* stage is for learning (training) and hyperparameter optimization, in which the structure and parameters of the detector are defined. For our detector, we choose an RNN detector with a gated recurrent unit (GRU) structure to exploit the time series nature of our data. During the training process, the model's hyperparameters are tuned using a genetic algorithm. The *second* stage is for testing, in which the detector is evaluated using the test dataset.

In the following subsections, we will briefly introduce the GRU classifier that is used in the lying EV detector as well as the optimization algorithm used to tune the model parameters.

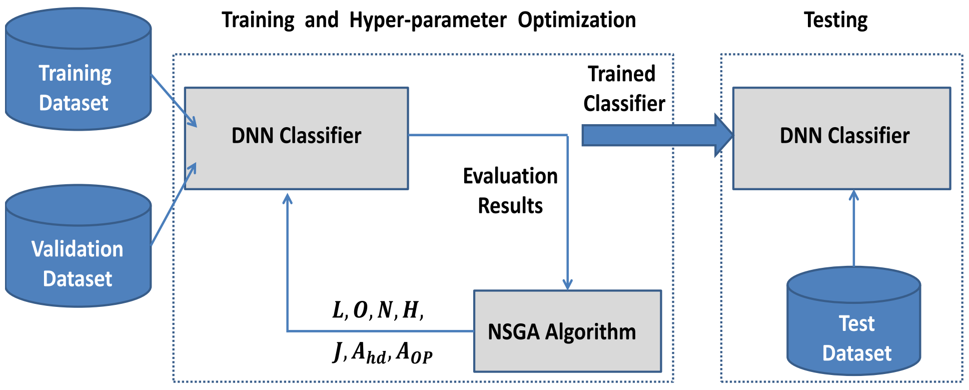


FIGURE 5. An abstract representation of the operations needed to develop the proposed detector.

A. GRU DETECTOR

An RNN is a type of artificial neural network in which the node-to-node connections form a directed graph with a temporal sequence and thus can reflect temporal dynamic behavior. Unlike feedforward neural networks, RNNs can process input sequences using their hidden states (memory), which makes them perfect for tasks such as unsegmented handwriting recognition or voice recognition [34]. Feedback loops are used to process the input data sequence to produce the final output sequence. Such feedback loops allow the retention of information, and this capability is often regarded as a memory functionality that preserves all measured parameters and links the inputs together, thus allowing sequential and temporal information to be processed by RNNs. The algorithm that is used in RNNs to update the network weights is backpropagation through time (BPTT) [35].

RNNs can suffer from the vanishing gradient problem; that is, when the gradient of a weight becomes too small as it is backpropagated through time, it can no longer contribute sufficiently during the learning process. Accordingly, a special kind of RNN structure has been introduced in the literature to solve this problem, namely, the *GRU* structure [36].

As shown in Fig. 6, our GRU detector consists of an input layer, a set of hidden layers and an output layer. The input layer receives the input vector $\text{RS}_i(d,*)$, which represents the SoC values of EV_i during day d in different reporting time periods $t \in \mathbb{T}$; this vector consists of T values. Following the input layer are L hidden layers, each of which consists of N neurons with a given activation function. The input layer receives an input vector and sends data to the hidden layers, which perform computations and send the resulting data to the output layer, which consists of two neurons for classifying a given input as corresponding to either an honest or lying EV. Accordingly, the output of the last layer is a two-element vector of the form $y(\text{RS}_i(d,*)) = (0,1)^T$ for a benign (honest) EV_i or $y(\text{RS}_i(d,*)) = (1,0)^T$ for a lying EV_i.

Given L hidden layers, for each layer $l \in \{1, 2, \ldots, L\}$, the output is denoted by o_t^l, and $o_t^0 = RS_i(d,*)$. Each hidden layer $l \in \{1, 2, \ldots, L-1\}$ has the following parameters:

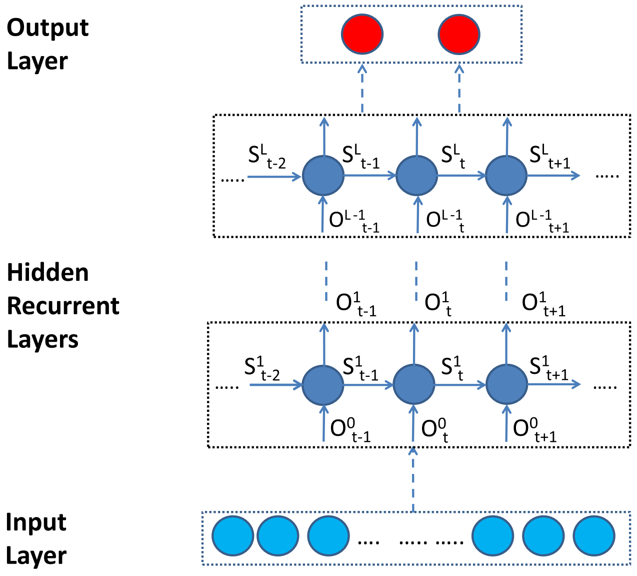


FIGURE 6. The architecture of the GRU-based detector.

- The input at time t is o_t^{l-1}, which is the output of the previous layer $l-1$.
- The hidden state s_{t-1}^l represents the memory computations using the previous layer's hidden state.
- The update gate for layer l and state t is computed using the input vector o_t^{l-1} and the previous hidden state s_{t-1}^l as follows: $z_t^l = \sigma(o_t^{l-1}U_z^l + s_{t-1}^l W_z^l + b_z^l)$. Here, $\sigma(\cdot)$ is the activation function, while b_z^l is the bias for the neurons in layer l. During the learning process, the weight matrices U_z^l and W_z^l are modified to reduce the error.
- The *reset* gate is calculated as follows: $r_t^l = \sigma(o_t^{l-1}U_r^l + s_{t-1}^l W_r^l + b_r^l)$, where U_r^l and W_r^l are weights that are learned during the training process.
- Finally, the current hidden state is computed as $s_t^l = (1 - z_t^l) \odot h_t^l + z_t^l \odot s_{t-1}^l$, where $\odot$ is the Hadamard product and $h_t^l = tanh(o_t^{l-1}U_h^l + (s_{t-1}^l \odot r_t^l)W_h^l + b_h^l)$. The output at time t is $o_t^l = softmax(V_o^l s_t^l + b_o^l)$, where V_o^l, U_h^l, and W_h^l are weight matrices that are determined via the training process.

B. HYPERPARAMETER OPTIMIZATION

Optimizing the parameters improves the performance of a detector. However, tuning a detector's hyperparameters is a difficult and time-consuming process because of the extensive computation required, and an exhaustive search for the optimal hyperparameters is practically impossible. Accordingly, we use the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [37], [38] to find an efficient solution in less time than would be required for an exhaustive grid search [39].

A genetic algorithm (GA) [40] uses a search strategy to find the best or fittest solution to a given (optimization) problem. In a GA, a population of candidate solutions is represented using a genetic representation. The GA then moves toward a better solution by applying evolutionary concepts such as natural selection and survival of the fittest.

In this article, NSGA-II is used to tune the hyperparameters of the GRU architecture, such as the number of hidden layers (L), the number of neurons per layer (N), the type of optimization algorithm applied (O), the method used for parameter initialization (H), the dropout rate (D), the weight constraint (J), and the activation functions used in the hidden layers (A_{hd}) and the output layer (A_{op}). NSGA is a multiple-objective optimization algorithm and is an instance of an evolutionary algorithm. There are two versions of this algorithm: the classical NSGA and the updated and currently canonical form, called NSGA-II. The objective of NSGA is to enhance the adaptive fit of a population of candidate solutions to a Pareto front restricted by a set of objective functions. The algorithm uses an evolutionary process with surrogates, using selection, genetic crossover, and genetic mutation as the evolutionary operators. The population is sorted into a hierarchy of subpopulations based on Pareto superiority. Optimal selection of the hyperparameters significantly improves the detection performance.

We used the dataset described in Section V-C for GRU model training, hyperparameter tuning, and testing. The model architecture produced in this way consists of 4 hidden layers of 512 neurons each, the tanh activation function in the hidden layers, and the softmax activation function in the output layer.

VII. EXPERIMENTS AND RESULTS

In this section, the evaluation methodology used to measure the performance of the proposed detector is explained, and then the evaluation results are discussed.

A. KEY PERFORMANCE METRICS AND EXPERIMENTAL METHODOLOGY

We define the following key performance metrics used in our evaluation process:

- Detection Accuracy (ACC): The ratio of the number of true positives and true negatives to the total number of EVs.

$$\text{Accuracy (ACC)} = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \tag{1}$$

T_P represents the number of true positives and is defined as the number of lying EVs that are correctly classified as lying. T_N is the number of true negatives and is defined as the number of honest EVs that are correctly classified as honest. F_P denotes the number of false positives and is defined as the number of honest EVs that are incorrectly classified as lying. F_N represents the number of false negatives and is defined as the number of lying EVs that are incorrectly classified as honest.

- False Alarm Rate (FA): The ratio of the number of false positives to the total number of honest EVs.

$$\text{False Alarm Rate (FA)} = \frac{F_P}{T_N + F_P} \tag{2}$$

- Detection Rate (DR): The ratio of the number of correctly detected lying EVs to the total number of lying EVs.

$$\text{Detection Rate (DR)} = \frac{T_P}{T_P + F_N} \tag{3}$$

- Highest Difference (HD): The difference between the detection rate (DR) and the false alarm rate (FA).

$$\text{Highest Difference (HD)} = \text{DR} - \text{FA} \tag{4}$$

- Area Under the Curve (AUC): The area under the receiver operating characteristic (ROC) curve. The ROC curve is a graphical means of capturing the relationship between DR and FA [41].

TABLE 4. Cases considered for training and evaluation of the detector.

Case	Training Dataset	Test Dataset
1	Honest and *Attacks 1*, *2*, *3*, and *4*	Honest and *Attacks 1*, *2*, *3*, and *4*
2	Honest and *Attacks 2*, *3*, and *4*	Honest and *Attack 1*
3	Honest and *Attacks 1*, *3*, and *4*	Honest and *Attack 2*
4	Honest and *Attacks 1*, *2*, and *4*	Honest and *Attack 3*
5	Honest and *Attacks 1*, *2*, and *3*	Honest and *Attack 4*

To evaluate the performance of our GRU model, we used the dataset described in Section V-C. The various cases considered in our evaluations are summarized in Table 4. The main objective of *case 1* was to evaluate the performance of the classifier on a test dataset consisting entirely of attacks already seen during the training phase. To evaluate the ability of our detector to detect new attacks, the main objective of *cases 2*, *3*, *4*, and *5* was to evaluate the performance against new attacks other than those on which the detector was trained. Specifically, in each of these four cases, we trained the detector on three attacks and tested it on the fourth attack that was not represented in the data used in the training phase.

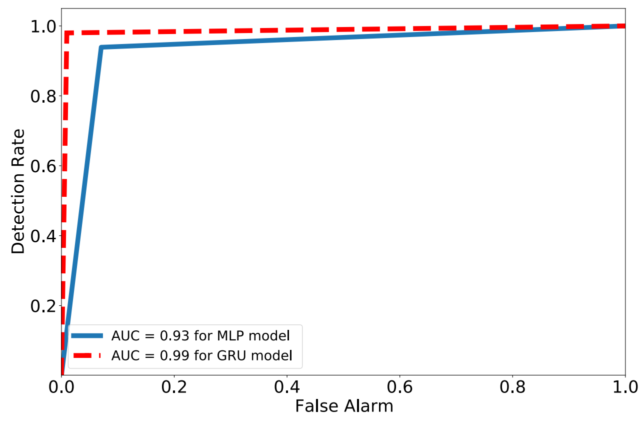


FIGURE 7. The ROC curves for the GRU and MLP detectors in case 1.

B. RESULTS AND DISCUSSION

In this subsection, the evaluation results for the proposed detection model are discussed.

1) EVALUATION OF THE PROPOSED DETECTOR

In case 1, the detector was trained using honest data as well as lying (malicious) data corresponding to all four attacks discussed in Section V-B. During the testing phase, the detector was similarly tested using honest samples in addition to malicious samples corresponding to all four attacks. For comparison with the GRU detector, another DNN detector based on a multilayer perceptron (MLP) model [42] was also trained and tested to evaluate how the GRU model benefits from the ability to exploit the time-series nature of the data. The results obtained with the best network architectures, i.e., the architectures that produced the best results, for both the GRU and MLP models are presented in Table 5. The best MLP results were obtained using a complicated architecture with 6 hidden layers of 768 neurons each, the sigmoid activation function in the output layer and the ReLU activation function in the hidden layers.

TABLE 5. Evaluation results for the DNN models in case 1.

Classifier	ACC	DR	FA	HD
GRU	0.985	0.991	0.02	0.971
MLP	0.934	0.930	0.061	0.869

It can be seen from Table 5 that the GRU detector achieves higher performance with a less complex architecture comprising only 4 hidden layers of 512 neurons each, the tanh activation function in the hidden layers and the softmax activation function in the output layer. The ROC curves in Fig. 7 visualize the performance of the two detectors in terms of the AUC metric, which has a value of 0.98 for the GRU detector and a value of 0.93 for the MLP detector.

As seen from Table 5 and Fig. 7, the GRU detector outperforms the MLP detector, with higher DR and ACC values and a lower FA value. These results can be interpreted to indicate that the GRU detector can better exploit the time series correlations in the SoC values reported by the EVs to achieve superior detection results. In addition, to achieve good detection performance, the architecture of the MLP detector must be more complex, i.e., with more layers and neurons.

2) DETECTION OF NEW ATTACKS

The main objective in this subsection is to investigate the ability of the detector to detect new attacks that are not represented in the training dataset. The detector was trained on a dataset containing honest data and data corresponding to three attacks. Then, we evaluated the detector against the fourth attack. As shown in Table 4, four different cases were considered, and the results are presented in Table 6.

TABLE 6. Evaluation results for GRU models in different cases.

Case	ACC	DR	FA	HD	AUC
2	0.836	0.778	0.106	0.672	0.840
3	0.929	0.933	0.075	0.8575	0.930
4	0.879	0.776	0.018	0.758	0.880
5	0.836	0.7385	0.0665	0.672	0.840

Case 2: In this case, the model was trained using a dataset containing honest data as well as malicious data corresponding to *Attacks 2*, *3*, and *4*. In the test phase, the dataset used consisted of samples of honest data as well as malicious samples corresponding to *Attack 1*. In this case, the GRU detector that achieved the highest performance contained 2 hidden layers of 512 neurons each, with a linear activation function in the hidden layers and the softmax activation function in the output layer.

Case 3: In this experiment, the training and validation datasets contained honest samples and malicious samples corresponding to *Attacks 1*, *3*, and *4*. Then, the model was tested using *Attack 2* samples in addition to honest samples. The GRU detector that achieved the highest performance had an architecture with 2 hidden layers of 768 neurons each, the tanh activation function in the hidden layers and the softmax activation function in the output layer.

Case 4: The training and validation datasets contained samples corresponding to *Attacks 1*, *2*, and *4* in addition to honest data. The test dataset contained honest data and samples corresponding to *Attack 3*. The GRU detector that achieved the highest performance was based on a model with only 1 hidden layer of 256 neurons each, the ReLU activation function in the hidden layer and the sigmoid activation function in the output layer.

Case 5: In this experiment, the training and validation phases were performed using a dataset consisting of malicious samples corresponding to *Attacks 1*, *2*, and *3* in addition to honest data. In the test phase, the dataset used contained honest samples and malicious samples corresponding to *Attack 4*. The GRU detector that achieved the highest performance had an architecture with 2 hidden layers of 128 neurons each, a linear activation function in the hidden layers and the softmax activation function in the output layer.

It can be seen from Table 6 that the false alarm rates are all low. The detection rate in case three is particularly high. For cases four and five, although the attacks are stealthy (i.e., the attacker sends false SoC values only during a short time interval and reports the correct SoC values the rest of the time), the detector is still able to detect them at a reasonable rate. In *Attack 2*, the attacker reports its correct SoC value multiplied by a factor of less than one. Although this is a difficult attack to detect if the detector is not trained on it because the pattern of the reported SoC values is close to that in the benign dataset, our detector is also able to detect this attack at a reasonable rate.

VIII. CONCLUSION

Recently, the adoption of electric vehicles has started to gain momentum due to their obvious advantages over gasoline vehicles. However, integrating electric vehicles into the power grid requires a charging coordination mechanism for balancing the charging load of the electric vehicles with the power supply. The existing charging coordination mechanisms assume that all electric vehicles report their correct state of charge values, although this is not beneficial to them. In this article, we first evaluated the impact of reporting false state of charge values on the charging coordination mechanism. Then, we proposed a detector based on deep learning techniques for identifying lying electric vehicles that report false state of charge values. To train the model, we created a new dataset; in particular, we proposed a number of attacks and used them to create a malicious dataset. Since the reported state of charge values exhibit time series correlations, a deep recurrent neural network was selected as the classifier for use in the detection process. The selection of the deep network architecture that provides the best performance was treated as an optimization problem with multiple objectives (detection rate and false alarm rate). Accordingly, to solve this problem, we used the Non-dominated Sorting Genetic Algorithm. The proposed detector was then experimentally evaluated, and the results indicate that the detector achieves a high detection rate with a low false alarm rate. Moreover, to investigate the ability of the detector to detect new attacks that are not represented in the training dataset, we also evaluated the detector on new attacks and found that it could detect them at a high detection rate.

ACKNOWLEDGMENT

The authors, therefore, gratefully acknowledge the DSR for technical and financial support.

REFERENCES

[1] K. Valogianni, W. Ketter, J. Collins, and G. Adomavicius, "Heterogeneous electric vehicle charging coordination: A variable charging speed approach," in *Proc. 52nd Hawaii Int. Conf. Syst. Sci.*, 2019, pp. 1–10.

[2] L. Hua, J. Wang, and C. Zhou, "Adaptive electric vehicle charging coordination on distribution network," *IEEE Trans. Smart Grid*, vol. 5, no. 6, pp. 2666–2675, Nov. 2014.

[3] E. Sortomme, M. M. Hindi, S. D. J. MacPherson, and S. S. Venkata, "Coordinated charging of plug-in hybrid electric vehicles to minimize distribution system losses," *IEEE Trans. Smart Grid*, vol. 2, no. 1, pp. 198–205, Mar. 2011.

[4] A. Schuller, "Charging coordination paradigms of electric vehicles," in *Plug in Electric Vehicles in Smart Grids* (Power Systems), vol. 88. Berlin, Germany: Springer, Nov. 2015, pp. 1–21.

[5] M. Baza, M. Nabil, M. Ismail, M. Mahmoud, E. Serpedin, and M. A. Rahman, "Blockchain-based charging coordination mechanism for smart grid energy storage units," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Atlanta, GA, USA, Jul. 2019, pp. 504–509.

[6] S. Deilami, A. S. Masoum, P. S. Moses, and M. A. S. Masoum, "Real-time coordination of plug-in electric vehicle charging in smart grids to minimize power losses and improve voltage profile," *IEEE Trans. Smart Grid*, vol. 2, no. 3, pp. 456–467, Sep. 2011.

[7] Y. Ota, H. Taniguchi, T. Nakajima, K. M. Liyanage, J. Baba, and A. Yokoyama, "Autonomous distributed V2G (vehicle-to-grid) satisfying scheduled charging," *IEEE Trans. Smart Grid*, vol. 3, no. 1, pp. 559–564, Mar. 2012.

[8] W. Tushar, J. A. Zhang, D. B. Smith, H. V. Poor, and S. Thiebaux, "Prioritizing consumers in smart grid: A game theoretic approach," *IEEE Trans. Smart Grid*, vol. 5, no. 3, pp. 1429–1438, May 2014.

[9] TaxiCabs. *TaxicabsRoutes*. Accessed Apr. 2020. [Online]. Available: http://crawdad.org/epfl/mobility/20090224/

[10] *kiaSoul Features*. Accessed Mar. 2020. [Online]. Available: https://ev-database.uk/car/1154/Kia-Soul-EV-64-kWh

[11] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intell.)*, Hong Kong, Jun. 2008, pp. 1322–1328.

[12] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Monticello, IL, USA, Sep. 2015, pp. 1310–1321.

[13] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Prague, Czech Republic, May 2011, pp. 5528–5531.

[14] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 287–297, Nov. 1999.

[15] H. Abderrahim, M. R. Chellali, and A. Hamou, "Forecasting PM10 in algiers: Efficacy of multilayer perceptron networks," *Environ. Sci. Pollut. Res.*, vol. 23, no. 2, pp. 1634–1641, Sep. 2015.

[16] N. B. Arias, J. F. Franco, M. Lavorato, and R. Romero, "Metaheuristic optimization algorithms for the optimal coordination of plug-in electric vehicle charging in distribution systems with distributed generation," *Electr. Power Syst. Res.*, vol. 142, pp. 351–361, Jan. 2017.

[17] S. Hajforoosh, M. A. S. Masoum, and S. M. Islam, "Real-time charging coordination of plug-in electric vehicles based on hybrid fuzzy discrete particle swarm optimization," *Electr. Power Syst. Res.*, vol. 128, pp. 19–29, Nov. 2015.

[18] J. F. Franco, M. J. Rider, and R. Romero, "A mixed-integer linear programming model for the electric vehicle charging coordination problem in unbalanced electrical distribution systems," *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 2200–2210, Sep. 2015.

[19] M. Mahmoud, M. Ismail, P. Akula, K. Akkaya, E. Serpedin, and K. Qaraqe, "Privacy-aware power charging coordination in future smart grid," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2016, pp. 1–6.

[20] M. Baza, M. Pazos-Revilla, M. Nabil, A. Sherif, M. Mahmoud, and W. Alasmary, "Privacy-preserving and collusion-resistant charging coordination schemes for smart grid," 2019, *arXiv:1905.04666*. [Online]. Available: http://arxiv.org/abs/1905.04666

[21] N. Z. Xu and C. Y. Chung, "Uncertainties of EV charging and effects on well-being analysis of generating systems," *IEEE Trans. Power Syst.*, vol. 30, no. 5, pp. 2547–2557, Sep. 2015.

[22] A. Ghosh and V. Aggarwal, "Control of charging of electric vehicles through menu-based pricing under uncertainty," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

[23] J. K. Pradhan, P. P. Verma, V. Khemka, V. E. Anoop, S. T. P. Srinivas, and K. S. Swarup, "Uncertainty handling for electric vehicle aggregator using IGDT," in *Proc. 20th Nat. Power Syst. Conf. (NPSC)*, Tiruchirappalli, India, Dec. 2018, pp. 1–6.

[24] H. Akhavan-Hejazi, H. Mohsenian-Rad, and A. Nejat, "Developing a test data set for electric vehicle applications in smart grid research," in *Proc. IEEE 80th Veh. Technol. Conf. (VTC-Fall)*, Vancouver, BC, Canada, Sep. 2014, pp. 1–6.

[25] Q. Wang, "Energy spatio-temporal pattern prediction for electric vehicle networks," 2018, *arXiv:1802.04931*. [Online]. Available: http://arxiv.org/abs/1802.04931

[26] G. S. Oh, D. J. Leblanc, and H. Peng, "Vehicle energy dataset (VED), a large-scale dataset for vehicle energy consumption research," 2019, *arXiv:1905.02081*. [Online]. Available: http://arxiv.org/abs/1905.02081
[27] *MATLAB Version 9.6 (R2019a)*, MathWorks, Natick, MA, USA, 2019.
[28] M. Nabil, M. Ismail, M. Mahmoud, W. Alasmary, and E. Serpedin, "PPETD: Privacy-preserving electricity theft detection scheme with load monitoring and billing for AMI networks," *IEEE Access*, vol. 7, pp. 96334–96348, Jun. 2019.
[29] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 63–77, Jan. 2006.
[30] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, New York, NY, USA, Apr. 2017, pp. 506–519.
[31] H. Ying, X. Ouyang, S. Miao, and Y. Cheng, "Power message generation in smart grid via generative adversarial network," in *Proc. IEEE 3rd Inf. Technol., Netw., Electron. Autom. Control Conf. (ITNEC)*, Mar. 2019, pp. 790–793.
[32] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.
[33] K. McCarthy, B. Zabar, and G. Weiss, "Does cost-sensitive learning beat sampling for classifying rare classes?" in *Proc. 1st Int. Workshop Utility-Based Data Mining (UBDM)*, New York, NY, USA, 2005, pp. 69–77.
[34] J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MA, USA, Dec. 2015, pp. 2980–2988.
[35] M. Fraccaro, S. Sønderby, U. Paquet, and O. Winther, "Sequential neural models with stochastic layers," in *Proc. 29th Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 2199–2207.
[36] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (GRU) neural networks," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Boston, MA, USA, Aug. 2017, pp. 1597–1600.
[37] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[38] D. Kalyanmoy, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proc. 6th Int. Conf. Parallel Problem Solving Nature*, Heidelberg, Germany, Sep. 2000, pp. 849–858.
[39] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proc. 1st IEEE Conf. Evol. Comput. IEEE World Congr. Comput. Intell.*, Orlando, FL, USA, Jun. 1994, pp. 82–87.
[40] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognit.*, vol. 33, no. 9, pp. 1455–1465, Sep. 2000.
[41] A. A. Awad, S. G. Sayed, and S. A. Salem, "A network-based framework for RAT-bots detection," in *Proc. 8th IEEE Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Vancouver, BC, Canada, Oct. 2017, pp. 128–133.
[42] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "DeepInspect: A black-box Trojan detection and mitigation framework for deep neural networks," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Macao, China, Aug. 2019, pp. 4658–4664.

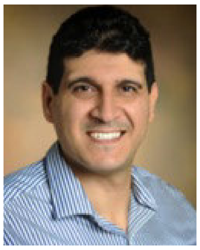
AHMED A. SHAFEE received the B.S. and M.S. degrees in computer engineering from Helwan University, Cairo, Egypt, in 2011 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Tennessee Tech University, USA. He is also a Graduate Teaching Assistant with the Department of Electrical and Computer Engineering, Tennessee Tech University. His research interests include machine learning, artificial intelligence, intrusion detection, cryptography and network security, smart grid and AMI networks, and electric vehicles.

MOSTAFA M. FOUDA (Senior Member, IEEE) received the B.S. (Hons.) and M.S. degrees in electrical engineering from Benha University, Egypt, in 2002 and 2007, respectively, and the Ph.D. degree in information sciences from Tohoku University, Japan, in 2011. He has served as an Assistant Professor with Tohoku University. He was a Postdoctoral Research Associate with Tennessee Tech University, USA. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Idaho State University, ID, USA. He also holds the position of an Associate Professor at Benha University. He has published over 30 papers in IEEE conference proceedings and journals. His research interests include cyber security, machine learning, blockchain, the Internet of Things (IoT), 5G networks, smart healthcare, and smart grid communications. He has served on the technical committees of several IEEE conferences. He is also a reviewer in several IEEE Transactions and magazines and an Associate Editor of IEEE Access.

MOHAMED M. E. A. MAHMOUD (Senior Member, IEEE) received the Ph.D. degree from the University of Waterloo, in 2011. From May 2011 to May 2012, he worked as a Postdoctoral Fellow with the Broadband Communications Research Group, University of Waterloo. From August 2012 to July 2013, he worked as a Visiting Scholar with the University of Waterloo and a Postdoctoral Fellow with Ryerson University. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Tennessee Tech University, USA. He is the author for more than 23 papers published in major IEEE conferences and journals, such as INFOCOM Conference, IEEE Transactions on Vehicular Technology, IEEE Transactions on Mobile Computing, and IEEE Transactions on Parallel and Distributed Systems. His research interests include security and privacy preserving schemes for smart grid communication networks, mobile *ad hoc* networks, sensor networks, and delay tolerant networks. He received the NSERC-PDF Award. He won the Best Paper Award from IEEE International Conference on Communications (ICC), Dresden, Germany, in 2009. He served as the Technical Program Committee Member for several IEEE conferences and a Reviewer for several journals and conferences, such as IEEE Transactions on Vehicular Technology, IEEE Transactions on Parallel and Distributed Systems, and *Peer-to-Peer Networking and Applications*. He serves as an Associate Editor for *Peer-to-Peer Networking and Applications* (Springer).

ABDULAH JEZA ALJOHANI (Member, IEEE) received the B.Sc. (Eng.) degree in electronics and communication engineering from King Abdulaziz University, Jeddah, Saudi Arabia, in 2006, and the M.Sc. (Hons.) and the Ph.D. degrees, awarded with no corrections, in wireless communication from the University of Southampton, Southampton, U.K., in 2010 and 2016, respectively. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, King Abdulaziz University. He is also an Active Member of the Center of Excellence in Intelligent Engineering Systems (CEIES). His research interests include machine learning, and optimization, distributed source coding, free-space optical communication, channel coding, cooperative communications, and MIMO systems.

WALEED ALASMARY (Senior Member, IEEE) received the B.Sc. degree (Hons.) in computer engineering from Umm Al-Qura University, Saudi Arabia, in 2005, the M.A.Sc. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2010, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, Canada, in 2015. During his Ph.D. degree, he was a Visiting Research Scholar with the Network Research Laboratory, University of California at Los Angeles (UCLA), in 2014. He was a Fulbright Visiting Scholar with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), from 2016 to 2017. He subsequently joined the College of Computer and Information Systems, Umm Al-Qura University, as an Assistant Professor of Computer Engineering. His mobility impact on the IEEE 802.11p article is among the most cited *Ad Hoc Networks* journal articles list, from 2016 to 2018. He published papers in prestigious conferences and journals, such as the IEEE Transactions on Vehicular Technology and the IEEE Communications Surveys and Tutorials. His current research interests include mobile computing, ubiquitous sensing, intelligent transportation systems, privacy, and anonymity.



FATHI AMSAAD (Member, IEEE) received the B.Sc. degree in computer science from the University of Benghazi (top ranked university in Libya), Libya, the dual M.Sc. degrees in computer science and in computer engineering from the University of Bridgeport, Bridgeport, CT, USA, with two professional certificates in networking and advanced programming, in 2012, and the Ph.D. degree in engineering, with an emphasis in computer science and engineering, from The University of Toledo, Toledo, OH, USA, in 2017. From August 2018 to August 2019, he was an Assistant Professor with The University of Southern Mississippi (USM), Hattiesburg, MS, USA. From August 2016 to August 2017, he was a Visiting Instructor with the University of South Florida (USF), Tampa, FL, USA. From August 2014 to August 2016, he worked as an Adjunct Instructor with Bowling Green State University (BGSU), Bowling Green, OH. From August 2012 to August 2016, he was a Graduate Assistant (a Teaching Assistant, a Research Assistant, and an Instructor) with The University of Toledo. He worked as a Teaching Assistant and the Senior Project Laboratory Administrator with the University of Benghazi. He worked as a Network and Security Engineer for several years prior coming to USA. He is currently an Assistant Professor with the School of Information Security and Applied Computing (SISAC), Eastern Michigan University (EMU). His research interests include cyber and physical systems security with a special interest in hardware oriented security and trust for embedded and smart systems security, security of the Internet-of-Things (IoT) applications, and AMI and autonomous systems cybersecurity. He served as a member and the Session Chair for several IEEE conferences and a Reviewer for several journals and conferences, such as IEEE Internet of Things Journal (IoT-J), IEEE Access, IEEE ISCAS conference, and MDPI journals. He received the prestigious IEEE Best Graduate Student Award from IEEE and the College of Engineering, The University of Toledo. He was a Nominee for the Best Ph.D. Dissertation Award.