Byzantine-resilient Economical Operation Strategy Based on Federated Deep Reinforcement Learning for Multiple Electric Vehicle Charging Stations Considering Data Privacy

Bin Feng, *Student Member*, *IEEE*, Huating Xu, Gang Huang, *Senior Member*, *IEEE*, Zhuping Liu, Chuangxin Guo, *Senior Member*, *IEEE*, and Zhe Chen, *Fellow*, *IEEE*

***Abstract*—With the goal of low-carbon energy utilization, electric vehicles (EVs) and EV charging stations (EVCSs) are becoming increasingly popular. The economical operation strategy is always a primary concern for EVCSs, while users' behavior and operating data leakage problems in EVCSs have not been taken seriously. Herein, federated deep reinforcement learning, a privacy-preserving method, is applied to learn the optimal strategy for multiple EVCSs. However, it is prone to Byzantine attacks. It is urgent to achieve an economical operation strategy while preserving data privacy and defending against Byzantine attacks. Therefore, this paper proposes a Byzantine-resilient federated deep reinforcement learning (BR-FDRL) method to address these problems. First, the distributed EVCS data are utilized by the federated deep reinforcement learning to train an economical operation strategy while preserving privacy by only transmitting gradients. The sampling efficiency is enhanced by both federated learning and stochastically controlled stochastic gradient. Then, the Byzantine-resilient gradient filter (BRGF) designs two distance rules to keep malicious gradients out. The case study verifies the effectiveness of the proposed BRGF in resisting Byzantine attacks and the effectiveness of federated deep reinforcement learning in improving convergence speed and reward and preserving privacy. The resluts show that the BR-FDRL method minimizes the operation cost by an average of 35% compared with the rule-based method while meeting the state of charge demand as much as possible.**

***Index Terms*—Byzantine resilience, federated learning, deep reinforcement learning, electric vehicle, privacy-preserving, economical operation.**

I. Introduction

ELECTRIC vehicles (EVs) help reduce oil consumption to achieve carbon neutrality. EV charging stations (EVCSs) own several charging points and provide battery charging services to EVs [1]. According to the global EV outlook 2022 by the International Energy Agency [2], publicly accessible charging points worldwide approached 1.8 million in 2021 and will be 12.9 million in 2030. EVCSs must choose suitable charging and discharging strategies to ensure that all EV charging needs are met while reducing operation costs. As the EV energy storage device is the battery, it can also discharge electricity to the grid, also known as vehicle to grid (V2G). V2G can effectively balance the peak-valley load demand to save operation costs.

The simplest operation strategy for EVCS is rule-based method, with a charging and discharging rule designed in advance based on expert experience. Although the rule-based method is more explainable, it is hard to apply in complex systems and is not adaptable to unseen situations. Further, the operation strategy for EVCS has been modeled as model-based optimization problems. Reference [3] considers the operation strategy for EVCS under microgrid integration/island operation to minimize charging costs and maximize energy storage benefits. Reference [4] uses the cognitive stochastic approximation-based optimization to realize EV charging at the lowest cost. Reference [5] considers the costs of charging, unit start/stop, power generation, pollution, and wind curtailment, then uses multi-objective collaborative optimization to achieve the lowest total cost. However, all of these methods have huge computational burdens and cannot give a real-time result when the exogenous environment changes. The uncertainty inherent in renewable energy and users' behavior makes the economical operation strategy more challenging [6], [7].

Data-driven methods such as deep reinforcement learning (DRL) [8] have lower computational demands enabling real-time computation, which have been used in virtual power plant management [9] and voltage control [10]. Reference [11] uses deep *Q*-learning to train the charging and discharging strategy. However, it discretizes continuous actions, re-

Manuscript received: November 3, 2023; revised: January 18, 2024; accepted: February 1, 2024. Date of CrossCheck: February 1, 2024. Date of online publication: April 5, 2024.

This work was supported by the National Natural Science Foundation of China (No. 52007173), the Joint Funds of National Natural Science Foundation of China (No. U22B2098), and the National Key Research and Development Program of China (No. 2023YFB3107603).

B. Feng, H. Xu, G. Huang (corresponding author), Z. Liu, and C. Guo are with the College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: fengbinhz@zju.edu.cn; xu_huating@zju.edu.cn; huanggang@zju.edu.cn; zhupingliu@foxmail.com; guochuangxin@zju.edu.cn).

Z. Chen is with Aalborg University, Aalborg 9220, Denmark (e-mail: zch@energy.aau.dk).

DOI: 10.35833/MPCE.2023.000850

sulting in a loss of some action space. Reference [12] uses a gradient ascent method combined with safety constraints for the EV charging management, which provides a continuous action space while ensuring that the EV is fully charged as much as possible before leaving. Reference [13] uses DRL to minimize charging costs and form an optimal driving path considering the transportation network.

However, [11] - [13] train the operation strategy only by one EVCS, which leads to an inefficient data collection efficiency, a slow training process, and a low-quality policy. Therefore, it is necessary to collect data from more EVCSs simultaneously to train a high-quality operation strategy. Distributed machine learning (DML) [14] is an effective method for learning from data that are scattered across EVCSs. Reference [15] uses the rooftop wind energy of different buildings to realize EV charging control through DML. However, DML has high communication requirements and construction costs for communication facilities. Further, the central node of DML has complete control over the data of distributed nodes. While due to business competition and users' privacy concerns, different EVCSs are reluctant to share their operation data, which leads to privacy-preserving issues in DML.

To solve the privacy leakage and communication issues, federated learning (FL), a collaborative machine learning, was first proposed by Google in 2016 [16]. The FL training process is overseen by a trusted third-party global server that aggregates information from different local nodes. The data transmitted by FL are usually neural network gradients or parameters, rather than original data, so that the privacy of each local node is preserved. Excessive communication requirements are also avoided by transmitting only the neural network gradients or parameters. FL is used to predict the load of EVCSs while preserving the privacy of each EVCS as well [17]. With the combination of DRL, federated deep reinforcement learning (FDRL) [18] has been proposed to enhance the effectiveness of DRL while preserving data privacy. FDRL has been utilized to address issues in multi-home energy management [19] and cloud robotic systems [20].

The Byzantine general problem was proposed by Landlord in 1982 [21]. Several divisions of the Byzantine Empire's army are stationed outside the enemy city, each commanded by its own general. The generals can only communicate with each other through messengers. After observing the situation of the enemy, they must formulate a common action plan such as an attack or a retreat. Victory can only be achieved when more than half of the generals jointly launch an attack. However, the Byzantine general problem arises when some of these generals may be traitors, attempting to prevent loyal generals from reaching an agreement on the action plan. The problem is how to achieve a consistent agreement on the attack when traitors are present.

The Byzantine general problem in the distributed system and DML is also known as the Byzantine attack problem, which means that Byzantine nodes may submit malicious information. The Byzantine attack caused by the Byzantine nodes compromise normal training or message-sending processes. Some researchers have focused on the Byzantine fault-tolerant blockchain in electricity trading pricing [22], as well as secure charging services for EVs through the execution of smart contracts [23]. In FDRL, the global server cannot directly obtain the user's local training data to verify its correctness, so the aggregating process is highly vulnerable to attacks from malicious users, which results in the reduced convergence speed and the difficulty in achieving the best reward or even divergence.

Therefore, it is necessary to protect the FDRL training from Byzantine attacks. This approach is expected to achieve rapid training convergence with Byzantine resilience while preserving data privacy. Herein, this paper proposes a Byzantine-resilient federated deep reinforcement learning (BR-FDRL) method for charging strategy of multiple EVCSs, aiming to reduce charging costs while meeting EV charging needs as much as possible. First, the Byzantine attack problem in FDRL training is illustrated, and the EVCS charging strategy is modeled as a Markov decision process (MDP). Then, the policy-based DRL is used to train the economical operation strategy of EVCS. In the FDRL training process, the global server only aggregates the gradient from the local EVCS to preserve the privacy of different EVCSs and prevent user data leakage. Further, a double-loop stochastically controlled stochastic gradient (SCSG) algorithm is introduced to improve the sampling efficiency and reduce the variance of gradients. However, since the local EVCS only transmits gradients, which cannot be verified by the global server, it is vulnerable to malicious attacks. Hence, to defend against Byzantine attacks, the mean of gradient medians is used to design the Byzantine-resilient gradient filter (BRGF), which excludes malicious gradients. The case study demonstrates that the proposed BR-FDRL method is effective in defending against various Byzantine attacks while preserving the privacy of local EVCS through FL by only transmitting gradients. The SCSG algorithm and FL can improve the sampling efficiency and expedite the convergence speed. The test results show that the proposed method can reduce the operation cost of EVCS and ensure the charging demand of EVs.

The main contributions of this paper are listed as follows.

1) Comprehensive modeling of EVCS operation. The economical operation strategy of EVCS is modeled as an MDP considering Byzantine attacks. To fully use the data scattered in each EVCS, FDRL is used to realize the economical operation of EVCS and meet the charging demand.

2) Novelty in Byzantine-resilient mechanism. To the best of our knowledge, the Byzantine attack on the energy management problem during the FDRL training has not been studied. A BRGF is designed to resist the malicious Byzantine gradient attack.

3) Improvement of sampling efficiency through SCSG. The SCSG algorithm is adopted to improve the convergence speed and sampling efficiency of FDRL. In addition, the sampling efficiency can also be effectively improved by FL.

The remainder of this paper is organized as follows. In Section II, the problem formulation is presented, which constructs the Byzantine attack problem and models the economical operation strategy of EVCS as an MDP. Section III elaborates on the proposed BR-FDRL method, including policy-

based DRL, FDRL, SCSG algorithm, and BRGF. Section IV analyzes the convergence curve of the training process and the test results in the case study. Finally, Section V draws the conclusion.

II. Problem Formulation

A. Byzantine Attack Problem

The Byzantine attack problem in FDRL is derived from the Byzantine general problem. In FDRL, Byzantine nodes are attackers that take control of some distributed nodes. These Byzantine nodes send malicious or arbitrary messages to the global server of FDRL, thereby slowing down or even disrupting the training process. We assume that Byzantine nodes have random noise (RN), random action (RA), and sign flipping (SF) attacks, which will be explained in detail in Section IV-A.

B. Economical Operation Strategy of EVCS

The FDRL training process is overseen by the global server. The local client nodes act as controllers and are distributed at local EVCSs. These controllers collect local EV and exogenous environment data and then make economical (dis)charging decisions based on the local DRL model.

We take a single local EVCS as an example. As shown in Fig. 1, each local EVCS has some charging points, and it includes a photovoltaic power generation system to provide renewable energy. When the EVCS has insufficient renewable energy, it must purchase electricity from the grid. EVs arrive and leave EVCSs randomly throughout the day. EVs can provide power to EVCSs, a process known as V2G, which reduces operation costs. The controller in the EVCS determines the charging and discharging rate of each charging point based on current charging demand, renewable energy generation, grid electricity price, etc. The economical operation strategy of EVCS aims to achieve the lowest cost while meeting the charging demands.

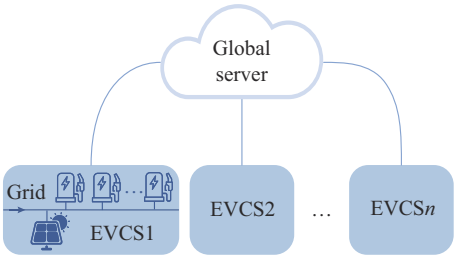

Fig. 1. Illustration of multiple EVCS.

C. MDP

MDP is the mathematical basis of DRL, in which common elements include the agent, environment, state, action, and reward. After executing the action under the current state, the agent receives a certain reward, and the state changes according to the action. The goal of DRL is to maximize the cumulative reward calculated based on the reward in each step.

Action a is determined by the policy function $\pi(a|s)$. After state transition, the next state s' is determined by the state transition function $p(s'|s,a)$. The policy and state transition functions are conditional probability distribution functions $P(\cdot)$. In the process of interaction between the agent and environment, the uncertainty of action a and state s' causes the randomness of MDP.

$$\pi(a|s)=P(A=a|S=s) \tag{1}$$

$$p(s'|s,a)=P(S=s'|S=s,A=a) \tag{2}$$

where A and S are the possible actions that the agent can take and the possible states that the agent can encounter in the environment, respectively.

Starting from an initial state s_0, the agent executes an action a_0 with reward r_0 until the done signal. The agent goes through a complete episode that fits the Markov hypothesis, i.e., $(s_0,a_0,r_0,s_1,a_1,r_1,\ldots,s_t,a_t,r_t,\ldots)$. After period t, the reward R_t with reward decay factor $\gamma\in[0,1]$ is used to express the cumulative reward U_t:

$$U_t=R_t+\gamma R_{t+1}+\ldots=\sum_{k=0}^{\infty}\gamma^k R_{t+k} \tag{3}$$

The action-value function Q_π is the conditional expectation of the reward. For a given policy π, Q_π describes the quality of action a_t in the current state s_t.

$$Q_\pi(s_t,a_t)=E(U_t|S_t=s_t,A_t=a_t) \tag{4}$$

where $E(U_t|S_t=s_t,A_t=a_t)$ is the expected value of the reward, given that the agent is in state s_t and takes action a_t.

The state-value function V_π is the expectation of the action-value function Q_π for the action set A. It is only related to policy π and state s_t, and describes how good or bad the policy function π is in a given state s_t.

$$V_\pi(s_t)=E_A(Q_\pi(s_t,A)) \tag{5}$$

where $E_A(Q_\pi(s_t,A))$ is the expected value of the action-value function Q_π over all possible actions A under a given policy π, while the agent is in state s_t.

Therefore, the state-value function V_π can guide the agent to choose the optimal policy π^* in the current state s_t.

The action-value function and the state-value function obey the Bellman equation, which calculates the previous variable from the subsequent variable. For the current state s_t, the environment receives an action a_t to obtain the next state s_{t+1}, and the return of this action is denoted as $r(s_t,a_t,s_{t+1})$. The Bellman equation for the action value function and the state value function is:

$$Q_\pi(s_t,a_t)=E_{s_{t+1}\sim p(s_{t+1}|s_t,a_t)}\left(r(s_t,a_t,s_{t+1})+\gamma E_{a_{t+1}\sim\pi(a_{t+1}|s_{t+1})}Q_\pi(s_{t+1},a_{t+1})\right) \tag{6}$$

$$V_\pi(s_t)=E_{a_t\sim\pi(a_t|s_t)}E_{s_t\sim p(s_t|s_t,a_t)}\left(r(s_t,a_t,s_{t+1})+\gamma V_\pi(s_{t+1})\right) \tag{7}$$

where $E_{a_t\sim\pi(a_t|s_t)}$ and $E_{s_t\sim p(s_t|s_t,a_t)}$ are the expected value when choosing an action a_t at state s_t according to the probability distribution defined by the policy π and the expected value given the current state s_t and the action a_t taken, respectively.

For clarity, we first focus on explaining the model for a single EVCS. The FDRL facilitates its extension to a multi-EVCS context. In this framework, each EVCS operates as a distinct agent, optimizing its own state and actions autonomously. Through FDRL, coordinated functionality among these agents is achieved, allowing for a common policy to be learned while upholding data confidentiality, thereby accurately representing the complex multi-EVCS environment.

Figure 2 shows the MDP model of the economical operation strategy for EVCS. The state s_t includes the solar radiation and electricity price for the current moment and the next three hours, the state of charge (SOC) information for the EV at the charging point, and the remaining time the EV stays at the EVCS.

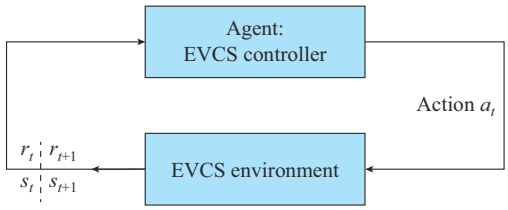

Fig. 2. MDP model of economical operation strategy for EVCS.

Action a_t is the (dis)charging rate at the moment t, which is a continuous variable ranging from −1 to 1. a_t greater than 0 indicates charging the EV and less than 0 indicates discharging the EV or V2G.

$$P_{t,\max}^{i}=\begin{cases}E_{\max}\left(1-SOC_t^i\right) & a_t^i\geq 0\\ E_{\max}\cdot SOC_t^i & a_t^i<0\end{cases} \tag{8}$$

$$P_{t,\max}^{i}\leq \eta P_{\max} \tag{9}$$

$$P_t^i=a_t^i P_{t,\max}^{i} \tag{10}$$

where $P_{t,\max}^{i}$ is the maximum (dis)charging power of the EV at the i^{th} charging point at the moment t; SOC_t^i is the SOC value of the EV at the i^{th} charging point at the moment t; $E_{\max}$ is the maximum capacity of the EV battery; η is the (dis)charging efficiency; $P_{\max}$ is the maximum charging power; P_t^i is the actual (dis)charging power of the i^{th} charging point at the moment t; and a_t^i is the (dis)charging rate at the i^{th} charging point at the moment t.

When action a^i is executed, the main changes in the environment are the SOC of the EV:

$$SOC_{t+1}^{i}=\begin{cases}SOC_t^i+\dfrac{P_t^i}{E_{\max}} & a^i>0\\ SOC_t^i-\dfrac{P_t^i}{E_{\max}} & a^i<0\end{cases} \tag{11}$$

The reward at the moment t is set as:

$$r_t\left(s_t,a_t\right)=-\sum_i c_t P_t^i-\sum_i\left[2\left(1-SOC_t^i\right)\right]^2 \tag{12}$$

where c_t is the electricity price at the moment t. The first item of (12) is the cost of purchasing electricity from the grid by the EVCS. The second item is the penalty for not being fully charged. The weight parameter 2 in the second term serves as a typical setting for our experiments to balance the two objectives effectively. Besides, the central controller and the local agents share the same reward.

III. Proposed BR-FDRL Method

In Section III-A, the policy-based DRL is used to solve MDP. To preserve the privacy of EVCSs, Section III-B elaborates on FL and FDRL. In Section III-C, FDRL requires each agent to train while collecting episodes. However, the interaction between DRL and the real environment is usually slow, expensive, and fragile. Hence, the choice of a sample-efficient gradient update algorithm is important. In Section III-D, the fundamental requirement of FDRL not to transmit sample data poses threats to the credibility of agents, so it is necessary to design an aggregation mechanism for stochastic gradients to eliminate gradients subject to Byzantine attacks. An overview of the proposed BR-FDRL method is given in Section III-E.

A. Policy-based DRL

Policy-based DRL directly optimizes the policy through gradient ascent for the expected reward, also known as the policy gradient method.

The policy network $\pi\left(a\,|\,s;\boldsymbol{w}\right)$ is used to approximate the policy function $\pi\left(a\,|\,s\right)$, where $\boldsymbol{w}$ is the policy network parameter. Policy learning aims at training a policy network with a near-optimal policy to control the agent. The training objective function is $J\left(\boldsymbol{w}\right)=E_S\left(V_\pi\left(S\right)\right)$, where $E_S(\cdot)$ is the expected value in the state S. Clearly, the greater the value of the objective function, the better the policy. To increase the value of the objective function $J\left(\boldsymbol{w}\right)$, the DRL model of the policy gradient, specifically the reinforce model [24], is used to update $\boldsymbol{w}$. After the gradient derivation, the policy gradient $\nabla_{\boldsymbol{w}}J\left(\boldsymbol{w}\right)$ is:

$$\nabla_{\boldsymbol{w}}J\left(\boldsymbol{w}\right)=E_S\left(E_{A\sim\pi\left(\cdot\,|\,S;\boldsymbol{w}\right)}\left(g\left(S,A;\boldsymbol{w}\right)\right)\right) \tag{13}$$

$$g\left(s,a;\boldsymbol{w}\right)\triangleq Q_\pi\left(s,a\right)\nabla_{\boldsymbol{w}}\ln\pi\left(a\,|\,s;\boldsymbol{w}\right) \tag{14}$$

where $E_{A\sim\pi\left(\cdot\,|\,S;\boldsymbol{w}\right)}(\cdot)$ is the the expected value in the action space A, given the state S and policy network parameter $\boldsymbol{w}$.

Given that the action-value function $Q_\pi\left(s,a\right)$ is typically unknown, the reinforce model employs Monte Carlo simulation to approximate $Q_\pi\left(s,a\right)$ through the actual cumulative reward. Consequently, the stochastic gradient $g\left(s,a;\boldsymbol{w}\right)$ serves as an unbiased estimator of $\nabla_{\boldsymbol{w}}J\left(\boldsymbol{w}\right)$.

It can be observed that the policy gradient method learns the parameterized policy $\pi_{\boldsymbol{w}}$ directly. The advantage is that there is no need to solve Q-value-maximizing optimization problems in the action space, which makes it more suitable for solving problems with high-dimensional or continuous action space. And it is more natural for modeling stochastic strategies.

B. FDRL

FDRL combines FL with DRL, which is a distributed DRL method that considers data privacy security [25]. The purpose of FL is to address data silos while preserving privacy. FL establishes a learning model based on multiple device datasets to ensure user privacy and security. However, the

main bottleneck of FL is the unstable communication between the local client and the global server. To this end, federated averaging (FedAvg) is proposed [26]. By increasing the computational load at local clients, the communication frequency is limited, and the communication load is reduced.

As shown in Fig. 3, policy gradient combined with FedAvg can be formulated as a distributed learning method that allows multiple agents to train a DRL model simultaneously. The global server does not require any private episode. During the training process, the central global server distributes an initial parameter of the policy function. The local agents interact with the local environment and train the local agent models simultaneously according to the collected episodes. The central global server aggregates the local policy network gradients or parameters to obtain the global model. After multiple iterations, a model similar to the centralized DRL model is well-trained. Policy gradient with FedAvg can effectively mitigate many privacy risks of traditional distributed DRL. With the use of FedAvg, FDRL performs multiple iterations of the local policy network and updates the global model once, which requires less communication frequency and solves the communication problem of FL.

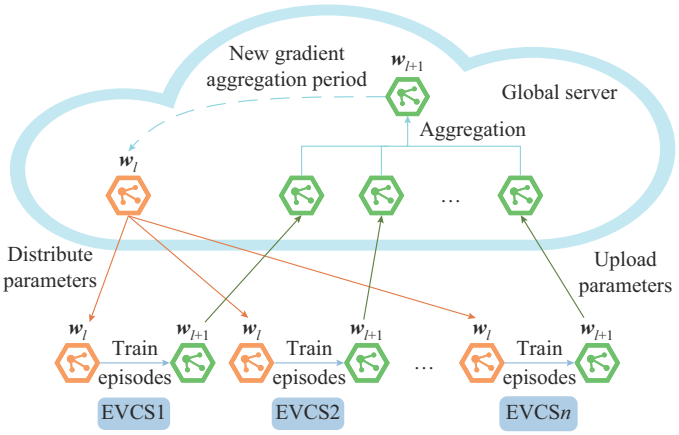

Fig. 3. Policy gradient combined with FedAvg.

C. SCSG Algorithm

Machine learning trains the optimal model by minimizing the expected risk. However, the expected risk cannot be calculated, so it is often modeled as an empirical risk minimization problem [27]:

$$\min_{\boldsymbol{w}} F(\boldsymbol{w}) = E\left(f(\boldsymbol{w},\boldsymbol{\xi})\right) = \frac{1}{n}\sum_{i=1}^{n} f\left(\boldsymbol{w},\xi_i\right) \tag{15}$$

where $\boldsymbol{w}\in\mathbf{R}^d$; $\boldsymbol{\xi}$ is the dataset; n is the the number of input-output pairs in the dataset; $E(\cdot)$ is the expectation function; $\xi_i=\left(x_i,y_i\right)$ denote the i^{th} input-output pair; $F(\cdot):\mathbf{R}^d\to\mathbf{R}$ is a function that is strongly convex, and its gradient is L-Lipschitz continuous; and $f(\cdot)$ is the loss function. In the control problem of EVCS, $f\left(\boldsymbol{w},\boldsymbol{\xi}\right)$ can be expressed as $f\left(\boldsymbol{w},\boldsymbol{\xi}\right)=\left(r_t\left(s_t,a_t\right)-Q\left(s_t,a_t;\boldsymbol{w}\right)\right)^2$.

Traditional gradient descent methods include full gradient descent and stochastic gradient descent (SGD) [28]. The gradient in SGD is an unbiased estimation of the gradient for the entire sample. However, the gradient variance increases with iteration, so SGD can only guarantee the sub-linear convergence speed, not the linear convergence speed [29]. A class of stochastic variance reduced gradient (SVRG) [30] is proposed to deal with the problem of excessive variance of SGD, which accelerates the convergence speed and reduces the computational burden.

The SVRG is a double-loop iteration method. The underlying idea is to calculate a batch gradient $\nabla F\left(\boldsymbol{w}_l\right)=\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(\boldsymbol{w}_l\right)$ in each outer loop, and the single iteration in the inner loop adopts $\boldsymbol{g}_j\leftarrow\nabla f_{i_k}\left(\tilde{\boldsymbol{w}}_k\right)-\left(\nabla f_{i_k}\left(\boldsymbol{w}_k\right)-\nabla F\left(\boldsymbol{w}_k\right)\right)$ to update the current parameters. $\boldsymbol{w}_l$ is the parameter at the l^{th} iteration of the outer loop; $\boldsymbol{w}_k$ is the parameter at the k^{th} iteration of the inner loop; $\tilde{\boldsymbol{w}}_k$ is the snapshot of the parameters taken periodically during the optimization process at the k^{th} iteration, used as a reference point for variance reduction; $i_k\in\{1,2,\ldots,n\}$; and $\nabla f_i(\cdot)$ is the the gradient of the loss function. In the convergence analysis of SGD, the variance of the sample gradient is assumed to have a constant upper bound, which results in non-linear convergence. Instead of simply selecting $\boldsymbol{g}_k=\nabla f_{i_k}\left(\tilde{\boldsymbol{w}}_k\right)$, SVRG uses the special update term $\nabla f_{i_k}\left(\tilde{\boldsymbol{w}}_k\right)-\left(\nabla f_{i_k}\left(\boldsymbol{w}_k\right)-\nabla F\left(\boldsymbol{w}_k\right)\right)$ to make the variance have a continuously reduced upper bound, so that the variance is much smaller than the expected value. Therefore, the linear convergence can be achieved [30].

The procedure for solving (15) with the SVRG are as follows.

Step 1: input the start point $\boldsymbol{w}_1\in\mathbf{R}^d$, learning rate $\alpha>0$, loop period L, and step number of SGD m.

Step 2: initialize $\tilde{\boldsymbol{w}}_1=\boldsymbol{w}_l$ and calculate the batch gradient $\nabla F\left(\boldsymbol{w}_l\right)=\frac{1}{n}\sum_{i=1}^{n}\nabla f_i\left(\boldsymbol{w}_l\right)$.

Step 3: randomly select $i_k\in\{1,2,\ldots,n\}$, and let $\tilde{\boldsymbol{g}}_k=\nabla f_{i_k}\left(\tilde{\boldsymbol{w}}_k\right)-\left(\nabla f_{i_k}\left(\boldsymbol{w}_k\right)-\nabla F\left(\boldsymbol{w}_k\right)\right)$ and $\tilde{\boldsymbol{w}}_{k+1}=\tilde{\boldsymbol{w}}_k-\alpha\tilde{\boldsymbol{g}}_k$. Let $k=k+1$. Loop until $k=m+1$ is reached, and then turn to *Step 2*.

Step 4: let $l=l+1$, and loop until $l=L+1$ is reached.

Step 5: output $\boldsymbol{w}_{l+1}=\tilde{\boldsymbol{w}}_{L+1}$ or $\boldsymbol{w}_{l+1}=\frac{1}{L}\sum_{i=1}^{L}\tilde{\boldsymbol{w}}_{i+1}$.

In the above procedure, $\nabla f_{i_k}\left(\boldsymbol{w}_k\right)-\nabla F\left(\boldsymbol{w}_k\right)$ is the biased estimation of gradient $\nabla f_{i_k}\left(\boldsymbol{w}_k\right)$.

To further reduce the computational burden, [31] proposed the SCSG algorithm. SCSG algorithm improves upon SVRG by utilizing variance reduction methods for enhanced learning and offers superior computational efficiency, making it better suited for large-scale and complex learning environments. SCSG algorithm calculates stochastical batch gradients instead of full gradients in each outer loop, while the number of inner loops follows a geometric distribution P_g. The procedure of SCSG algorithm is as follows.

Step 1: input the initial point $\boldsymbol{w}_1\in\mathbf{R}^d$, learning rate $\alpha>0$, loop period L, and batch size B.

Step 2: sample a batch $I_l\in\{1,2,\ldots,n\}$ evenly.

Step 3: calculate the batch gradient $\nabla F\left(\boldsymbol{w}_l\right)=\frac{1}{B}\sum_{i\in I_l}\nabla f_i\left(\boldsymbol{w}_l\right)$. Let $\tilde{\boldsymbol{w}}_1=\boldsymbol{w}_l$. Generate inner-loop count $N_l\propto P_g$.

Step 4: randomly select $i_k \in I_l$. Let $\tilde{\boldsymbol{g}}_k \leftarrow \nabla f_{i_k}(\tilde{\boldsymbol{w}}_k) - \left(\nabla f_{i_k}(\boldsymbol{w}_k) - \nabla F(\boldsymbol{w}_k)\right)$ and $\tilde{\boldsymbol{w}}_{k+1} = \tilde{\boldsymbol{w}}_k - \alpha \tilde{\boldsymbol{g}}_k$. Let $k = k+1$. Loop until $k = N_l$ is reached, output $\boldsymbol{w}_{l+1} = \tilde{\boldsymbol{w}}_{N_l}$, and turn to *Step 2*.

Step 5: let $l = l+1$, and loop until $l = L+1$ is reached.

Step 6: output $\boldsymbol{w}_{l+1} = \tilde{\boldsymbol{w}}_{L+1}$ or $\boldsymbol{w}_{l+1} = \dfrac{1}{L}\sum_{i=1}^{L} \tilde{\boldsymbol{w}}_{i+1}$.

SCSG algorithm calculates a random batch gradient in each outer loop, where a single iteration in the inner loop takes $\nabla f_{i_k}(\tilde{\boldsymbol{w}}_k) - \left(\nabla f_{i_k}(\boldsymbol{w}_k) - \nabla F(\boldsymbol{w}_k)\right)$ to update the current parameters, and the number of inner-loop iterations N_l obeys a geometric distribution P, which effectively reduces the computational burden. Note that, $\nabla f_{i_k}(\tilde{\boldsymbol{w}}_k) - \left(\nabla f_{i_k}(\boldsymbol{w}_k) - \nabla F(\boldsymbol{w}_k)\right)$ is an unbiased estimation of gradient $\nabla f_{i_k}(\boldsymbol{w}_k)$, whose variance is smaller than that of $\nabla f_i(\boldsymbol{w}_l)$ (the proof can be found in [32]). Therefore, SCSG algorithm has a smaller gradient update variance than SGD and has better convergence, which means that the number of samples required for convergence is less.

D. BRGF

If the training process of FDRL contains Byzantine nodes, random failures or malicious gradient attacks will cause the system to be difficult to converge. If these Byzantine nodes send incorrect gradients during aggregation, it will slow down the training convergence speed and even lead to divergence, further increasing the sampling and computational burden of DRL.

As shown in Fig. 4, to filter out Byzantine gradients and only aggregate good gradients, the BRGF is added for gradients calculated by each local agent. The gradient difference of each episode can be bounded in stochastic non-convex optimization [33]. Besides, suppose that the Byzantine node rate does not exceed β at any time, where β is less than 50% due to the settings of the Byzantine general problem.

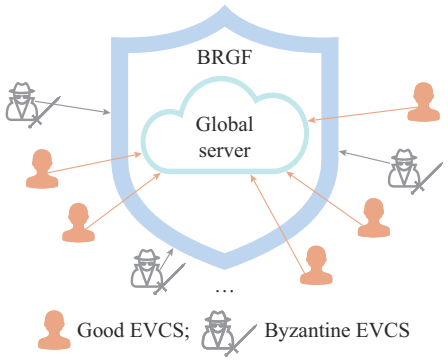

Fig. 4. Byzantine gradient identification using BRGF.

The Byzantine gradients are filtered out according to the following rules.

BRGF rule 1: according to Pinelis' inequality [34], all good gradients have a high probability of occurrence in a small region:

$$P\left(\left\| \mu_t^{(m')} - \mu_t^{(m)} \right\| \le T_\mu\right) \ge 1-\delta \tag{16}$$

where $\mu_t^{(m)}$ is the estimated gradient from the m^{th} agent in outer loop round t; $T_\mu = 2\sigma\sqrt{V/B_t}$ is the filtering threshold, σ is the variance bound, B_t is the batch size, $V = 2\lg(2M/\delta)$, and M is number of agents; and $\delta \in (0, 1)$.

The following three points outline the steps involved in BRGF.

1) Construct multiple sets of gradient samples S_1, where the distance between any two gradients in one set should be less than T_μ. The number of gradients in these sets should be more than half of M.

2) Calculate the median of each gradient set and then obtain the mean of the medians to obtain μ_t^{mom}.

3) Select all agent gradients whose distances from μ_t^{mom} are less than or equal to T_μ into a good gradient set.

If the number of gradients filtered by rule 1 is less than $M(1-\beta)$, rule 1 is too conservative and filters out the good gradients. Hence, rule 2 is required to ensure that all good gradients are included.

BRGF rule 2 is similar to rule 1, which only needs to replace T_μ with 2σ.

The two rules guarantee that all non-attacked gradients are retained. Even if Byzantine gradients are not filtered out, they have limited convergence effects on the FDRL, since their distance from $\nabla J(\boldsymbol{w}_0^l)$ is limited to 3σ. $\boldsymbol{w}_0^l$ is the snapshot of the parameters taken periodically at the l^{th} iteration.

After good gradients are filtered out by rules 1 and 2, their average value is the stochastic batch gradient needed for the outer loop in SCGC.

E. Overview of Proposed BR-FDRL Method

The algorithm pseudocode of the proposed BR-FDRL method is shown in Algorithm 1. The algorithm using SCSG can effectively reduce the variance of gradient updates, improving its convergence speed and sampling efficiency. The details of Algorithm 1 are as follows.

Algorithm 1: BR-FDRL

Input: $\tilde{\boldsymbol{w}}_0$, outer-loop count T and its batch size B, and inner-loop count N_l and its batch size b

1. **for** $l = 1, 2, \ldots, L$ **do**
2. $\boldsymbol{w}_0^l \leftarrow \tilde{\boldsymbol{w}}_{l-1}$
3. **for** $k = 1, 2, \ldots, K$ **do**
4. Calculate $\boldsymbol{w}_l^{(k)} = \dfrac{1}{B}\sum_{i=1}^{B} g(s_i, a_i; \boldsymbol{w}_0^l)$ for good EVCS, where $\boldsymbol{w}_l^{(k)}$ is the average gradient of good gradients at the k^{th} iteration
5. Obtain $\boldsymbol{w}_l$
6. **for** $h = 1, 2, \ldots, N_l - 1$ **do**
7. Calculate $\boldsymbol{u}_h^l = \dfrac{1}{b}\sum_{j=1}^{b}\left(g(s_j, a_j; \boldsymbol{w}_h^l) - \varphi g(s_j, a_j; \boldsymbol{w}_0^l)\right) + \boldsymbol{w}_l$, where $\boldsymbol{u}_h^l$ is the gradient used for updating, and $\boldsymbol{w}_h^l$ is the parameter at the h^{th} iteration of the inner loop
8. Calculate $\boldsymbol{w}_{h+1}^l = \boldsymbol{w}_h^l + \alpha \boldsymbol{u}_h^l$
9. $\tilde{\boldsymbol{w}}_l \leftarrow \boldsymbol{w}_{N_l}^l$

Output: $\tilde{\boldsymbol{w}}_L$

First, initialize the policy network parameter of global server $\tilde{\boldsymbol{w}}_0$. In each outer loop (lines 1-9), the neural network

parameters $\boldsymbol{w}_{t-1}$ in the global server are sent to the local EVCS (line 2). If it is a good EVCS, the gradient will be calculated based on (14); otherwise, a malicious gradient is emitted, which is also picked up by the global server (line 4). Malicious gradients are excluded by BRGF, and then the average of the filtered gradients is obtained as the parameter $\boldsymbol{w}_l$ required for the inner-loop update (line 5). In the inner loop (lines 6-8), the server independently samples b samples to update the gradient of the policy function according to the SCSG algorithm. φ is an importance weight used for the unbiased gradient estimation (line 7). Finally, the output of the inner loop $\boldsymbol{w}_{N_l}^l$ is sent to $\tilde{\boldsymbol{w}}_l$ (line 9). The output of the outer loop is well-trained policy network parameter $\tilde{\boldsymbol{w}}_L$.

IV. Case Study

A. Environment Settings

1) Byzantine Attack Settings

According to the assumption of Byzantine attack problem, Byzantine nodes generally make up less than half of the total nodes. We assume that Byzantine nodes have the following three model types.

1) RN model. In the RN model, each Byzantine node introduces a stochastic component by transmitting a vector comprised of RN to the global server. The vector simulates the gradient data, representing the inaccuracies that commonly arise during data transmission in distributed networks. This random perturbation may lead to non-convergent behavior in the optimization algorithm, affecting the training stability of the model.

2) RA model. In the RA model, each Byzantine node ignores the prescribed method of the global server and instead takes actions randomly selected from a predefined action space. It simulates instances of system failures such as hardware malfunction, resulting in miscalculations in the gradient update steps. The model under this failure can exhibit diminished performance and decreased convergence rate.

3) SF model. In this adversarial failure model, Byzantine nodes compute the gradient accurately, but intentionally invert the sign and apply a scaling factor before transmitting it to the global server. This perturbation distorts the direction and magnitude of the gradient, adversely influencing the gradient descent optimization. This can result in a deviation of model parameters from their optimal values, thereby impeding learning efficacy.

2) EV Charging Environment Settings

We assume that each EVCS contains ten charging points and a group of photovoltaic power generation systems. The SOC of EVs arriving at the EVCS is ranged from 10% to 80%. The arrival time is distributed from 0 to 22 hours, the minimum departure time is 2 hours after arrival, and the maximum duration of stay can be until the following day. The battery characteristics of all EVs are consistent. The charging/discharging efficiency η of all EV batteries is 91%, and battery capacity $E_{\max}$ is 30 kWh. The maximum charging power of the charging point $P_{\max}$ is 11 kW. Each charging and discharging decision determines the power of the next hour. The cost of EVCS purchasing electricity from the grid is determined by the fixed peak-valley time-of-use electricity price, and the cost of V2G is also determined by the purchase price. The peak-valley electricity price is set to be 0.05 €/kWh during 0-7 hours, 0.1 €/kWh during 8-20 hours, and 0.05 €/kWh during 21-24 hours. The environment simulation platform is modified from ChargGym [35] using Pytorch on the Linux server.

B. Experiment Settings

The training hyperparameter settings are as follows. The episode of each training session is set to be 24, i.e., the training is performed in a day. The reward decay factor γ is 0.99. The learning rate of the neural network is 2×10^{-5}, the numbers of hidden layer units of the actor and critic neural networks are 400 and 300, respectively, the inner activation function is ReLU, and the activation function of the output layer is tanh. The chosen variance bound σ is determined based on initial experiments and statistical evaluation of gradient dispersion, which is set to be 3.

C. Training Phase

To achieve more convincing convergence curves, each experiment is repeated 10 times, and the average reward in the validation dataset is displayed.

1) Effectiveness of Byzantine Resilience

To verify the effectiveness of the proposed method against Byzantine attacks, the training convergence results of the proposed method, stochastic variance reduced policy gradient (SVRPG), and simple policy gradient (SimplePG) in the scenario of ten EVCSs (three of which are Byzantine nodes that emit SF/RA/RN attacks maliciously) are compared, which are denoted as BR-W10B3-SF/RA/RN, SVRPG-W10B3-SF/RA/RN, and SimplePG-W10B3-SF/RA/RN, respectively. To establish control groups for comparison, the training convergence outcomes of the proposed method, SVRPG, and SimplePG are also evaluated in a scenario involving ten EVCSs without any Byzantine nodes. These control groups are labeled as BR-W10, SVRPG-W10, and SimplePG-W10, respectively.

As shown in Fig. 5, the proposed method can effectively defend against various types of Byzantine attacks with the help of BRGF. The final training reward of the proposed method in the W10B3 setting containing three Byzantine nodes is the same as that in the W10 setting without Byzantine nodes, indicating that the BRGF can defend against various attacks without affecting the convergence performance.

All three types of Byzantine attacks have a severe impact on SVRPG and SimplePG. RN usually slows down the convergence speed of SVRPG and SimplePG. SF causes the agent to send large negative gradients to the global server, so that the training turns to an undesirable direction, leading to a gradual decrease in reward and ultimately causing the training to fail. RA attacks have the severest impact, resulting in a serious unlearnable training process and the lowest reward. This is because, in the actual economical operation of EVCS, the trained agent should be charged most of the time. However, the Byzantine node ignores the policy trained by the global server and randomly executes the action. This results in more discharging actions, which further

causes lots of invalid learning gradients to return to the global server, seriously hindering the training convergence.

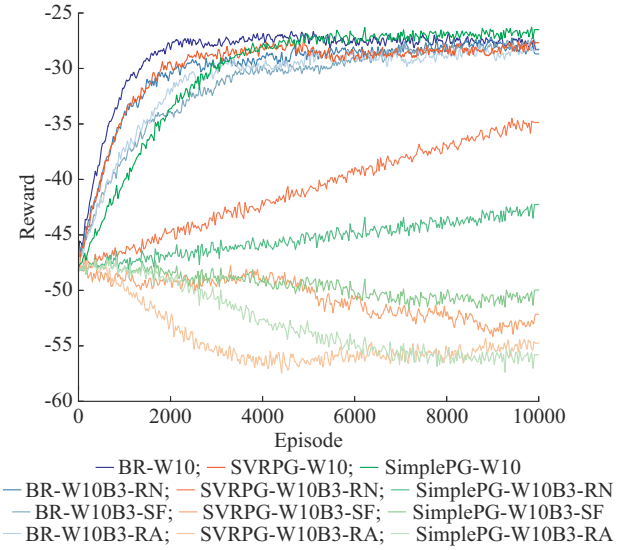

Fig. 5. Training convergence results of proposed method, SVRPG, and SimplePG in different scenarios.

As shown in Fig. 6, the number of filtered gradients selected by the proposed method is plotted to verify the effectiveness of BRGF in filtering Byzantine gradients. Based on experiment settings, BRGF needs to filter out seven good gradients as there are only three malicious Byzantine gradients.

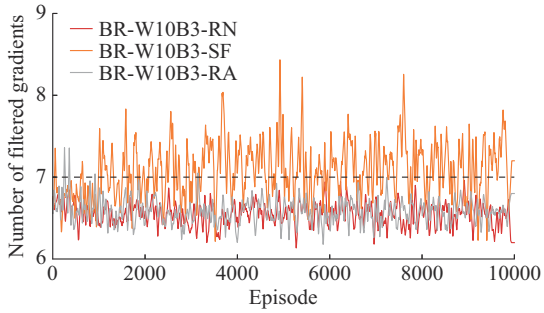

Fig. 6. Number of filtered gradients selected by proposed method.

The proposed method can filter the malicious Byzantine gradients well for all three attacks. Under SF attacks, there may be instances where the number of filtered gradients exceeds seven, meaning that the Byzantine gradients are included. As shown in Fig. 5, the convergence speed of BR-W10B3-SF is relatively slow among all the BR-W10B3 settings, and the BRGF and proposed method can still ensure the convergence of the final training. Even if Byzantine gradients are included, the distance between the filtered gradients is less than 3σ. Hence, the impact of Byzantine gradients on training is small and the training convergence can still be guaranteed.

The filter accuracy metric quantifies the ability of the model to correctly filter out Byzantine gradients while retaining good ones. The value of this metric falls between 0 and 1, with values closer to 1 indicating higher filter accuracy. The filter accuracy is the sum of gradients that are correctly filtered out and retained, divided by the total number of gradients.

The filter accuracies achieved under RN, SF, and RA attacks are 94.8%, 92.5%, and 96.1%, respectively, which means that the BRGF exhibits a robust accuracy of 94.8% for the RN attack, 92.5% for the SF attack, and a commendable 96.1% for the RA attack. The results underscore the robustness of BRGF against Byzantine attacks.

2) Effectiveness of FL and SCSG Algorithm

To further validate the role of FL, comparisons between W10 setting and W1 setting (single EVCS) are conducted. During the training process, each local EVCS agent only sends gradients to the global server to ensure that the private data of each EVCS are not leaked, which preserves privacy.

As shown in Fig. 7, regardless of the method used, the fast convergence speed of W10 setting proves that FL can improve the training sampling efficiency, and multiple agents sharing the training experience obtained from different environments can accelerate the convergence speed. The rewards of FL training are higher than those of the single training, proving that FL can fully explore various environments and increase the reward in low-probability unseen scenarios.

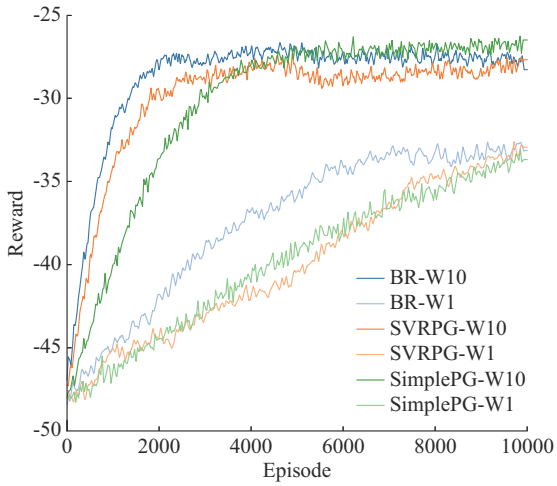

Fig. 7. Effectiveness of FL.

By comparing BR-W10, SVRPG-W10, and SimplePG-W10, it can be observed that the proposed method has the fastest convergence speed, followed by SVRPG, while SimplePG has the slowest convergence speed. This verifies the superiority of the proposed method in sampling efficiency. In the non-FL training of W1 setting, the advantages of the high sampling efficiency of the proposed method can also be reflected.

D. Testing Phase

The effectiveness of FDRL under Byzantine attacks in the EVCS environment is empirically validated. Different methods are tested on ten different days, and their average test results are shown in Table I. At the charging station, a centralized controller verifies each charging point and records the departure time of every connected EV. If an EV departs with-

in the next t hours, then the power station will fully charge that specific EV; otherwise, the power station uses the current solar power to charge the EV, where $t = E_{max}/(P_{max}\eta)$. The electricity price and solar power in one test day are shown in Fig. 8.

TABLE I
AVERAGE TEST RESULTS OF DIFFERENT METHODS

Method	Reward	Cost	SOC deviation (%)
Rule-based	–40.2	39.8	1.02
SimplePG-W1	–33.3	28.8	11.23
SVRPG-W1	–31.8	29.1	9.32
BR-W1	–33.8	30.0	9.34
SimplePG-W10	–27.9	24.9	7.48
SVRPG-W10	–27.3	24.9	6.07
BR-W10	–27.2	25.2	5.10
SimplePG-W10B3-SF	–51.5	19.3	80.38
SVRPG-W10B3-SF	–52.2	23.4	72.70
BR-W10B3-SF	–27.9	25.6	5.73
SimplePG-W10B3-RA	–56.5	20.2	90.79
SVRPG-W10B3-RA	–55.4	26.0	73.66
BR-W10B3-RA	–27.5	25.9	4.44
SimplePG-W10B3-RN	–42.3	19.9	56.02
SVRPG-W10B3-RN	–36.9	16.6	50.98
BR-W10B3-RN	–27.4	25.5	4.72

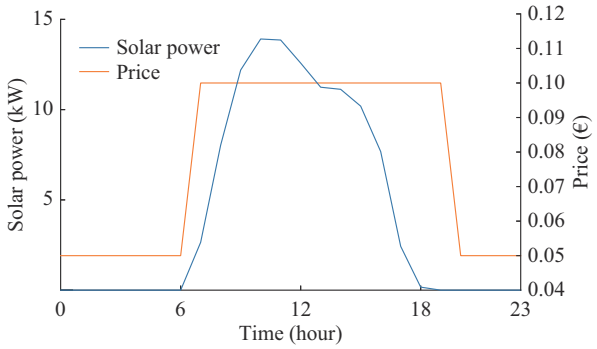

Fig. 8. Electricity price and solar power in one test day.

In Table I, the reward r_t is calculated based on (12) and the cost is the first item of (12), which measures the average cost of purchasing electricity for EVCS from the grid per day. The SOC deviation is calculated by the average of $\left(1 - SOC_t^i\right)^2$.

The rule-based method adjusts the charging and discharging only based on the departure time, without considering the electricity price. Although rule-based method can effectively ensure that the EV is charged as much as possible, it is the most expensive. The reward of the DRL (SimplePG-W1, SVRPG-W1, BR-W1) is better than the rule-based method. Although there is a small amount of dissatisfaction with the SOC in the DRL, the charging cost of EVCS is significantly reduced. Moreover, the overall test reward of W10 is better than W1, which also verifies the fact that FL can improve the reliability of the model and avoid the problem of insufficient training of a single agent in low-probability scenarios.

Further comparing the impact of the Byzantine attack, RA and SF attacks can seriously affect the training process of SimplePG, causing a serious lack of SOC with a deviation of 80.38% and 90.79%, respectively. Meanwhile, the serious lack of SOC results in low costs for purchasing power from the grid. Other non-Byzantine-resilient methods also cause a large amount of dissatisfaction with the SOC when exposed to Byzantine attacks, which makes it challenging to meet the primary charging demand. Byzantine attacks in FDRL can lead to a worse method than rule-based method. The proposed method uses BRGF to effectively defend against various Byzantine attacks, ensuring that the unsatisfied SOC level is around 5% and the power purchasing cost is reduced by about 35% compared with that of the rule-based method.

As shown in Figs. 9 and 10, BR-W10B3-RN (the strongest method against Byzantine attacks), SimplePG-W10B3-RN, and rule-based method are selected for comparison.

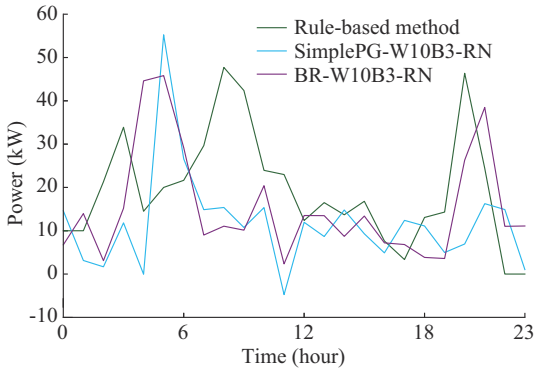

Fig. 9. Comparison of charging power in one EVCS with different methods.

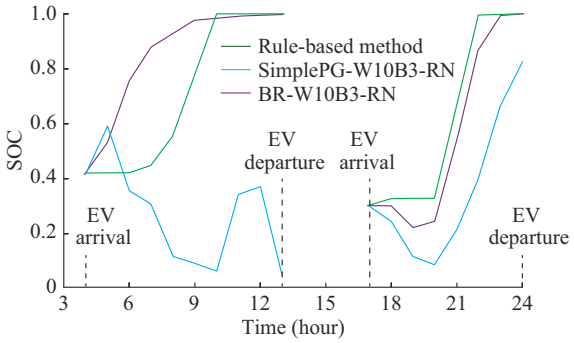

Fig. 10. Comparison of SOC in one charging point of EVCS with different methods.

Figure 9 shows the charging power in one EVCS with different methods. The BR-W10B3-RN can transfer the charging demand more effectively, which can make the EV charge less or even make the EV discharge during the period with high electricity prices (8^{th}-20^{th} hours) and charge more during the period with low electricity prices. The rule-based method only judges the charging or discharging action based on the rules. Although it can make EVs charge more, it can neither make decisions based on electricity prices, nor make EV discharge during the period with peak electricity prices. Therefore, the rule-based method has the highest cost. The

SimplePG-W10B3-RN shows a more disordered charging result, indicating that Byzantine attacks severely affect the FDRL.

Further in Fig. 10, one charging point of EVCS is selected to analyze the impact of the three methods on EV SOC. The BR-W10B3-RN and rule-based method can successfully meet the charging demand. The SimplePG-W10B3-RN makes illogical decisions at some moments, resulting in poor SOC satisfaction. The analysis of SOC between the 17th and 20th hour shows that the BR-W10B3-RN can effectively acquire the discharging strategy when the EV has just arrived and is not in a rush to depart, or during periods with high electricity prices, to reduce the cost. From the 1st to 7th hour, the BR-W10B3-RN can make the EV charge as much as possible to avoid the period with high electricity price from the 7th hour. This intelligent charging-discharging decision can effectively reduce charging costs while meeting SOC demands.

V. Conclusion

This paper proposes a Byzantine-resilient economical operation strategy based on FDRL for multiple EVCSs.

1) The proposed BR-FDRL method fulfils SOC demand and minimizes charging costs by shifting the charging demand from high-cost to low-cost intervals, even when facing severe Byzantine attacks.

2) The proposed BR-FDRL method improves the sampling efficiency via the SCSG algorithm.

3) Specialized gradient filtering rules are utilized by BRGF to distinguish between malicious and benign gradients, and distance metrics are utilized as the distinguishing criterion.

4) FDRL augments data privacy across individual EVCS through FL.

In summary, the proposed BR-FDRL method provides a robust, efficient, and privacy-preserving solution for managing multi-EVCS systems even under unfavorable conditions. Additionally, it should be noted that like any other privacy-preserving technique, FL is not completely foolproof in preserving privacy. Privacy could still be revealed by re-engineering the original data from the gradient or other attack methods. Further enhancing the privacy-preserving capabilities of the proposed method could be investigated in future work.

References

[1] S. Li, W. Hu, D. Cao *et al*., "Electric vehicle charging management based on deep reinforcement learning," *Journal of Modern Power Systems and Clean Energy*, vol. 10, no. 3, pp. 719-730, May 2022.

[2] International Energy Agency. (2022, May). Global EV outlook 2022 – analysis. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2022/

[3] M. Zhang and J. Chen, "The energy management and optimized operation of electric vehicles based on microgrid," *IEEE Transactions on Power Delivery*, vol. 29, no. 3, pp. 1427-1435, Jun. 2014.

[4] C. D. Korkas, S. Baldi, P. Michailidis *et al*., "A cognitive stochastic approximation approach to optimal charging schedule in electric vehicle stations," in *Proceedings of 2017 25th Mediterranean Conference on Control and Automation*, Valletta, Malta, Jul. 2017, pp. 484-489.

[5] W. Yin and X. Qin, "Cooperative optimization strategy for large-scale electric vehicle charging and discharging," *Energy*, vol. 258, p. 124969, Nov. 2022.

[6] B. Hu, L. Zhan, S. Sahoo *et al*., "Synchronization stability analysis under ultra-weak grid considering reactive current dynamics," *IEEE Transactions on Industrial Electronics*, doi:10.1109/TIE.2024.3370947

[7] Z. Yang, R. Zhu, and W. Liao, "Minkowski distance based pilot protection for tie lines between offshore wind farms and MMC," *IEEE Transactions on Industrial Informatics*, vol. 71, no. 17, pp. 15220-15223, Nov. 2024.

[8] G. Huang, F. Wu, and C. Guo, "Smart grid dispatch powered by deep learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 23, no. 5, pp. 763-776, May 2022.

[9] B. Feng, Z. Liu, G. Huang *et al*., "Robust federated deep reinforcement learning for optimal control in multiple virtual power plants with electric vehicles," *Applied Energy*, vol. 349, p. 121615, Nov. 2023.

[10] L. Wang, T. Wang, G. Huang *et al*., "Softly collaborated voltage control in PV rich distribution systems with heterogeneous devices," *IEEE Transactions on Power Systems*, vol. 39, no. 4, pp. 5991-6003, Jul. 2024.

[11] Z. Wan, H. Li, H. He *et al*., "Model-free real-time EV charging scheduling based on deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5246-5257, Sept. 2019.

[12] H. Li, Z. Wan, and H. He, "Constrained EV charging scheduling based on safe deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2427-2439, May 2020.

[13] T. Qian, C. Shao, X. Wang *et al*., "Deep reinforcement learning for EV charging navigation by coordinating smart grid and intelligent transportation system," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1714-1723, Mar. 2020.

[14] J. Verbraeken, M. Wolting, J. Katzy *et al*. (2019, Dec.). A survey on distributed machine learning. [Online]. Available: https://doi. org/10.48550/arXiv.1912.09789

[15] Y. Yang, Q. Jia, G. Deconinck *et al*., "Distributed coordination of EV charging with renewable energy in a microgrid of buildings," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 6253-6264, Nov. 2018.

[16] J. Konecny, H. B. McMahan, D. Ramage *et al*. (2016, Oct.). Federated optimization: distributed machine learning for on-device intelligence. [Online]. Available: https://arxiv.org/abs/1610.02527

[17] B. Li, Y. Guo, Q. Du *et al*., "P3: privacy-preserving prediction of real-time energy demands in EV charging networks," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 3, pp. 3029-3038, Mar. 2023.

[18] J. Qi, Q. Zhou, L. Lei *et al*. (2021, Aug.). Federated reinforcement learning: techniques, applications, and open challenges. [Online]. Available: https://doi.org/10.48550/arXiv.2108.11887

[19] S. Lee and D. H. Choi, "Federated reinforcement learning for energy management of multiple smart homes with distributed energy resources," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 488-497, Jan. 2022.

[20] B. Liu, L. Wang, and M. Liu, "Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems," *IEEE Robotics and Automation Letters*, vol. 4, pp. 4555-4562, Oct. 2019.

[21] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382-401, Jul. 1982.

[22] Y. Jiang, K. Zhou, X. Lu *et al*., "Electricity trading pricing among prosumers with game theory-based model in energy blockchain environment," *Applied Energy*, vol. 271, p. 115239, Aug. 2020.

[23] Z. Su, Y. Wang, Q. Xu *et al*., "A secure charging scheme for electric vehicles with smart communities in energy blockchain," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4601-4613, Jun. 2019.

[24] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229-256, May 1992.

[25] H. Zhuo, W. Feng, Y. Lin *et al*. (2020, Feb.). Federated reinforcement learning. [Online]. Available: https://doi.org/10.48550/arXiv.1901.08277

[26] H. B. McMahan, E. Moore, D. Ramage *et al*. (2017, Feb.). Communication-efficient learning of deep networks from decentralized data. [Online]. Available: https://doi.org/10.48550/arXiv.1602.05629

[27] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223-311, Jan. 2018.

[28] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125-161, Aug. 2013.

[29] C. Blair, "Problem complexity and method efficiency in optimization," *SIAM Review*, vol. 27, pp. 264-265, Jun. 1985.

[30] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proceedings of the 26th Interna-*

tional Conference on Neural Information Processing Systems, Lake Tahoe, USA, Dec. 2013, pp. 315-323.
[31] L. Lei and M. I. Jordan. (2019, May). Less than a single pass: stochastically controlled stochastic gradient method. [Online]. Available: https://doi.org/10.48550/arXiv.1609.03261
[32] P. Xu, F. Gao, and Q. Gu, "An improved convergence analysis of stochastic variance-reduced policy gradient," in *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference*, Tel Aviv, Israel, Aug. 2020, pp. 541-551.
[33] D. Alistarh, Z. Allen-Zhu, and J. Li. (2018, Mar.). Byzantine stochastic gradient descent. [Online]. Available: https://doi.org/10.48550/arXiv.1803.08917
[34] I. Pinelis, "Optimum bounds for the distributions of martingales in Banach spaces," *The Annals of Probability*, vol. 22, no. 4, pp. 1679-1706, Oct. 1994.
[35] G. Karatzinis, C. Korkas, M. Terzopoulos *et al*., "Chargym: an EV charging station model for controller benchmarking," in *Proceedings of Artificial Intelligence Applications and Innovations*, León, Spain, Jun. 2022, pp. 241-252.

Bin Feng received the B.S. degree in electrical engineering and its automation from North China Electric Power University, Beijing, China, in 2019. He is presently working towards the Ph. D. degree in Zhejiang University, Hangzhou, China. His current research interests include deep reinforcement learning, federated learning, energy management, and forecasting.

Huating Xu received the B. S. degree from Sichuan University, Chengdu, China, in 2012, and the M. S. degree from China Electric Power Research Institute, Beijing, China, in 2020. Presently, he is working toward the Ph. D. degree with the College of Electrical Engineering, Zhejiang University, Hangzhou, China. He was an Electrical Engineer with the China General Nuclear Power Group, Dalian, China, from 2012 to 2017. His current research interests include deep reinforcement learning, reactive power and tie-line power adjustment, optimal power flow, and economic dispatching region.

Gang Huang received the Ph.D. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2018. During 2015-2016, he was with Argonne National Laboratory, Argonne, USA. Presently, he is an Assistant Professor with the College of Electrical Engineering, Zhejiang University. Prior to this role, he served as a Senior Researcher at Zhejiang Lab, Hangzhou, China. His current research interests include artificial intelligence for power and energy systems.

Zhuping Liu received the bachelor's degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2022. She is currently working towards the M.S. degree at Zhejiang University. Her current research interests include new-type power system risk assessment and management based on artificial intelligence methods.

Chuangxin Guo received the B.S., M.S., and Ph.D. degrees in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 1992, 1994, and 1997, respectively. He is currently a Professor with the College of Electrical Engineering, Zhejiang University, Hangzhou, China. Prior to joining Zhejiang University, he was the Director of Beijing Dongfang Electronics Research Institute, Beijing, China. His research interests include power system operation and planning, and power systems information and communication technologies.

Zhe Chen received the B.Eng. and M.Sc. degrees from the Northeast China Institute of Electric Power Engineering, Jilin, China, and the Ph. D. degree from the University of Durham, Durham, UK. He is currently a Full Professor with the Department of Energy Technology, Aalborg University, Aalborg, Denmark. His research interests include power systems, power electronics, electric machines, and wind energy and modern power systems.