Privacy-Preserving Byzantine-Robust Federated Learning via Deep Reinforcement Learning in Vehicular Networks

Yanghe Pan, Zhou Su, Yuntao Wang, Jinhao Zhou, and Mohamed M.E.A. Mahmoud

***Abstract*—Federated learning (FL), as a transformative approach in vehicular networks, enables collaborative model training without exposure of vehicle users' local data. However, vehicular FL services are susceptible to Byzantine attacks, since malicious vehicles can upload false local updates to degrade the performance of the global model. Additionally, the honest-but-curious mobile edge computing (MEC) nodes may attempt to extract sensitive information from the shared updates via inference or inversion attacks. In this paper, we propose DRL-PBFL, a privacy-preserving Byzantine-robust FL scheme in vehicular networks that tolerates Byzantine vehicles and the honest-but-curious MEC node during the training process. Specifically, the deep reinforcement learning (DRL) technique is leveraged to maintain the robustness of FL against Byzantine vehicles, and a novel secure aggregation algorithm is designed to prevent curious inferences from the honest-but-curious MEC node. A performance-based weighting aggregation policy optimized by a deep deterministic policy gradients (DDPG) component is also devised to aggregate local updates. As it is challenging to directly integrate weighting aggregation policy with general pseudo-random masking methods for Byzantine resistance, the Lagrange interpolation is utilized to generate local perturbations to defend against the honest-but-curious MEC node and eliminate the weighted perturbations after secure aggregation. Notably, DRL-PBFL effectively achieves Byzantine robustness in FL even though the vehicles' data is not independently and identically distributed (non-IID). Experiments on three benchmark datasets with different adversarial settings show that DRL-PBFL effectively improves the global model accuracy under tailored Byzantine attacks compared with state-of-the-art schemes.**

***Index Terms*—Federated learning, Byzantine robustness, reinforcement learning, privacy preservation.**

I. Introduction

A. Background and The Byzantine Threat

With the increasing desire for privacy protection in vehicular services, federated learning (FL) has emerged as a promising paradigm to efficiently deliver privacy-preserving vehicular services by collaboratively training machine learning (ML) models among multiple vehicles without directly transmitting their local private data [1], [2]. For instance, Intel has developed a practical objective detection framework for autonomous vehicles utilizing the FL paradigm [3]. Coordinated by the MEC node (e.g., base station), during each communication round of FL, vehicles train the received global model on their local sensory data and compute corresponding local updates, which are then uploaded to the MEC node. The MEC node acts as the aggregation server to combine all the vehicles' local updates and produce a new global model for next-round training.

However, vehicles can be vulnerable to various attacks due to their open and distributed characteristics. The compromised/malicious vehicles may launch Byzantine attacks by uploading malicious updates, which prevent the global model from normal convergence and undermine the model performance. Recent works have demonstrated that the classical mean aggregation is vulnerable to Byzantine attacks and the global model is skewed even if there is only a single malicious Byzantine participant [4], [5]. Furthermore, privacy threats, such as the honest-but-curious central server, arise during the model training process [6]–[11]. For instance, the MEC node may execute the steps of the FL process honestly in vehicular networks, while being curious about vehicles' private data and trying to infer sensitive information. Previous works have demonstrated that the original training data can be extracted from the updates or gradients via inversion attacks [8], [9].

B. Related Works and Key Challenges

1) Byzantine-robust FL (BFL): Currently, several BFL schemes have been proposed to defend against Byzantine participants, which can be categorized into two main groups: *statistics-based schemes* and *performance-based schemes*. Statistics-based schemes enhance Byzantine robustness by excluding updates that are geometrically distant from the benign updates. Blanchard *et al.* proposed Krum, where the central server selects the single update closest to its neighbors as the final update [4]. Yin *et al.* proposed Median and Trimmed Mean, two coordinate-wise statistics-based FL schemes that use the median or trimmed mean of all local updates in each dimension as the final global update [12]. Pillutla *et al.* developed RFA, which computes the geometric median of local updates as the aggregation update efficiently [13]. Guerraoui *et al.* proposed Bulyan, introduced Bulyan, leveraging Krum to identify potential malicious participants and employing coordinate-wise aggregation schemes for the remaining updates [14]. Performance-based schemes, on the other hand, reduce the impact of Byzantine participants with poor update performance. Reinforcement learning (RL) and quality assessment are two main techniques for implementing

Y. Pan, Z. Su, and Y. Wang are with the School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, China.(email: pyh1998@stu.xjtu.edu.cn; zhousu@ieee.org;)

J. Zhou is with the Graduate School of Information, Production and Systems, Waseda University, Fukuoka, Japan.

M. Mahmoud is with Electrical and Engineering Department, College of Engineering, Tennessee Technological University, Tennessee, USA.

TABLE I
THE COMPARISON BETWEEN DRL-PBFL AND EXISTING SCHEMES.

Schemes	Privacy Preservation	Defense Against General Byzantine Attacks	Defense Against Tailored Byzantine Attacks	Non-IID Data Distribution	Computation Efficiency
Conventional FL	✗	✗	✗	✗	✓
Statistics-based BFL ([4], [12]–[14])	✗	✓	✗	✗	✓
Performance-based BFL ([15]–[17])	✗	✓	✓	✗	✓
MPC-BPFL ([18]–[20])	✓	✓	✗	✗	✗
HE-BPFL ([21], [22])	✓	✓	✗	✓	✗
TEE-BPFL ([23], [24])	✓	✓	✗	✗	✓
Proposed DRL-PBFL	✓	✓	✓	✓	✓

performance-based BFL schemes. Pan *et al.* proposed GAAvernor, which effectively mitigates Byzantine attacks even when the proportion of malicious participants is 50% or more using a vanilla deep RL technique in distributed learning systems [15]. Xie *et al.* presented Zeno, which uses a small validation dataset on the central server to assess local updates' quality scores, then averages multiple high-quality updates [16]. Cao *et al.* introduced FLtrust, which calculates the cosine similarity between each local update and the server update (obtained from a standard dataset) to determine the quality scores of local updates, using these scores to weight the aggregation process [17].

Despite their effectiveness, these BFL schemes assume a fully trusted central server. However, in practical scenarios, the central server can access participants' original local model updates and potentially extract sensitive information. Furthermore, these schemes suffer performance degradation when participants' data is highly non-IID.

2) Privacy-preserving FL (PFL): To protect local model updates from a curious server, participants typically mask or encrypt their original updates in the FL training process. Bonawitz *et al.* proposed a practical secure aggregation scheme in FL, where each participant masks their local update with perturbations from a pseudo-random generator, preventing the server from accessing individual updates [25]. Aono *et al.* and Phong *et al.* proposed PFL schemes based on additive homomorphic encryption (HE), where the uploaded gradients are encrypted during the FL training process to prevent potential inferences [26], [27]. Unfortunately, masking or encrypting local model updates complicates the detection of Byzantine attackers, as the server cannot compute statistical information or evaluate updates' performance without access to the original updates.

Differential privacy (DP) is another rigorous privacy-preserving technique in FL, offering theoretical privacy guarantees [28]–[31]. Geyer *et al.* proposed the client-level (ε, δ)-DP mechanism, which effectively protects participants' privacy in an FL system by adding the crafted Gaussian noise [28]. Wei *et al.* proposed a DP-based privacy-preserving FL framework with extensive theoretical proofs, demonstrating DP's effectiveness in FL [29]. Truex *et al.* integrated DP and HE to design a hybrid PFL scheme, achieving the balance between privacy protection and computational efficiency [30]. However, DP application often leads to performance deterioration, especially in the presence of Byzantine participants, making it challenging to achieve a trade-off between privacy preservation and performance for practical tasks.

3) Privacy-preserving Byzantine-robust FL (PBFL): Recently, a few works have focused on PBFL, primarily using secure multi-party computing (MPC), HE, and trusted execution environments (TEE). MPC allows participants to compute statistics information such as Euclidean distances between encrypted or masked updates for Byzantine detection and supports Byzantine-robust aggregation of these updates. He *et al.* proposed a secure aggregation framework with the aid of two non-colluding servers utilizing two-party computation (2PC) [18]. So *et al.* further removed the strong assumption of two non-colluding servers by utilizing secret sharing while maintaining the robustness and privacy of local mode updates [19]. Rathee *et al.* also proposed a robust secure aggregation framework ELSA utilizing the 2PC, significantly improving the efficiency [20]. These schemes compute the pairwise distances between updates with MPC and apply Krum aggregation for Byzantine robustness. Evaluating encrypted updates with HE is another direction for achieving PBFL. Ma *et al.* proposed ShieldFL, which presents a secure cosine similarity method to compute distances between gradients encrypted by two-trapdoor HE, achieving privacy-preserving Byzantine-robustness in both IID and non-IID scenarios [21]. Yazdinejad *et al.* facilitated Byzantine-tolerant aggregation by employing the Gaussian Mixture Model and Mahalanobis distance, with additive HE protecting participants' privacy during training [22]. Additionally, TEEs such as Intel SGX are employed to protect participants' updates while enabling Byzantine resilience. Zhao *et al.* and Hashemi *et al.* TEEs to compute pairwise distances without privacy leakage and aggregate updates with Krum [23], [24].

However, most existing privacy-preserving Byzantine-robust FL schemes employ complex cryptographic techniques, raising computational efficiency concerns and hindering deployment in vehicular networks. Furthermore, most existing schemes utilize privacy-preserving Krum aggregation to achieve Byzantine robustness, which has a non-negligible impact on the global model performance, particularly with non-IID data. Meanwhile, these PBFL schemes remain vulnerable to tailored Byzantine attacks [32]–[34]. Moreover, constrained memory in TEE-based schemes poses a significant hardware limitation, complicating computations for large-scale models with complex structures and extensive parameters.

The comparison between the proposed scheme and existing

schemes is illustrated in TABLE I.

C. Contributions

In this paper, we propose DRL-PBFL, a novel Privacy-preserving Byzantine-robust Federated Learning scheme via Deep Reinforcement Learning (DRL) in vehicular networks. DRL-PBFL defends against various Byzantine attacks effectively, including tailored attacks, while guaranteeing the vehicles' privacy against the inferences of an honest-but-curious MEC node. The main idea is to use a performance-based weighting aggregation policy to aggregate vehicles' updates masked by eliminable perturbations. There is a root dataset with a limited size on the MEC side, which conforms to standard data distribution and is used to evaluate the performance of updates. Intuitively, the performance-based weighting aggregation policy can identify Byzantine vehicles in FL and assign low weights, even if the vehicles' data distribution is non-IID. The aggregation policy is determined only by the performance of the global model on the root dataset, independent of the distribution of the local data among vehicles. As the masked updates cannot be evaluated on the root dataset directly, we leverage the deep deterministic policy gradient (DDPG) RL technique to learn a Byzantine-robust aggregation policy based on the global model performance on the root dataset during the training process. To handle the perturbation elimination issue caused by weights, we devise a secure aggregation algorithm based on Lagrange interpolation. Vehicles construct perturbations by the Lagrange polynomial and current aggregation policy, the perturbations are eliminated after the aggregation of the updates by the MEC node. Theoretical analysis proves that the perturbations are eliminated correctly, and the RL method improves the Byzantine robustness of the model.

We evaluate DRL-PBFL on three ML benchmark tasks: MNIST, CIFAR10, and Fashion-MNIST, in different adversarial settings. Four types of Byzantine attacks are implemented to evaluate the Byzantine robustness, among which the Gaussian random attack is the general Byzantine attack method, while the local model poison (LMP) [32], the little is enough (LIE) [33], and the optimization model poisoning (OMP) attacks [34] are tailored Byzantine attacks. The experimental results demonstrate that DRL-PBFL can effectively defend against these Byzantine attacks with desirable model performance, and the results also show that DRL-PBFL can achieve better performance than existing Byzantine-robust schemes in non-IID scenarios. In particular, compared with the existing state-of-the-art secure Byzantine FL schemes, our DRL-PBFL can significantly improve the model performance under tailored Byzantine attacks, i.e., it improves the global model accuracy by 80%+ on MNIST, 30%+ on CIFAR10, and 15% on Fashion-MNIST.

Our main contributions are summarized as follows.

- DRL-PBFL is the first work that addresses the challenge of the aggregation and evaluation of masked local model updates based on performance-based schemes utilizing the DRL in vehicular networks.
- DRL-PBFL can effectively defend against various Byzantine attacks with diverse settings, including the Gaussian random, LMP, LIE, OMP, and adaptive Byzantine attacks, even if the vehicles' data distribution is non-IID.
- An innovative privacy-preserving aggregation algorithm is designed based on Lagrange interpolation to solve the perturbations elimination problem caused by weights. The perturbations are added on the local model updates and eliminated after weighted aggregation so that the privacy of vehicles is protected against the honest-but-curious MEC node.
- The theoretical analyses are carried out to prove the correctness and privacy of DRL-PBFL, and extensive experiments are conducted to demonstrate that DRL-PBFL outperforms existing secure Byzantine-robust FL schemes in both model performance and model robustness against various Byzantine attacks.

Outline. In Section II, we present the preliminary knowledge. Then, we formalize the problem that we aim to address in Section III. After that, we introduce DRL-PBFL in detail and provide its theoretical analysis in Section IV, followed by the performance evaluation and experimental results over three datasets in Section V. Finally, we draw our conclusions in Section VI.

II. Background

In this section, we introduce some prior background knowledge of FL, Byzantine-robust aggregation policy, secure aggregation, and secret sharing.

A. Federated Learning

FL is a novel distributed ML paradigm, in which participants train a global model collaboratively without sharing their original local data with the aid of a central server. In a general FL workflow, the central server starts with the initialization of a global model w_g and communicates with the participants multiple rounds to produce the global model. Specifically, each communication round of the FL can be divided into three steps:

- **Broadcasting global model**. The central server broadcasts the global model w_g to all or sampled participants.
- **Training local models**. After receiving the global model w_g, participants use the local dataset to train a local model w_l. Generally, each participant trains the local model using the stochastic gradient descent (SGD) method for one or more local epochs and then computes the differences between the original global model and the trained local model (or gradients) as the update, e.g., $\Delta w = w_l - w_g$. Finally, the update Δw is uploaded to the central server.
- **Aggregating local updates and updating the global model**. The server aggregates the local updates $\{\Delta w\}$ uploaded by participants via a special aggregation policy $\mathcal{A}$. Then, the aggregation is used to update the global model with a learning rate α.

However, the naive mean aggregation policy in the FL training process can be skewed by a single malicious Byzantine participant [12]. At the same time, participants may worry about the privacy leakage of the local model updates acquired by the honest-but-curious server [8], [9].

B. Byzantine-Robust Aggregation Policy

To tolerate Byzantine failures in the FL, many Byzantine-robust aggregation schemes have been proposed, including both statistics-based Byzantine-robust aggregation policies and performance-based Byzantine-robust aggregation policies.

- **Krum [4].** Krum selects an update as the global update based on the pairwise Euclidean distances between the updates. Specifically, the server computes a score $s(i)$ for each participant i in a communication round via the following equation:

$$s(i) = \sum_{i \to j} \|\Delta\theta_i - \Delta\theta_j\|^2, \tag{1}$$

  there are n participants and f Byzantine participants, and for any $i \neq j$, $i \to j$ denotes that $\Delta\theta_j$ belongs to the $n-f-2$ closest updates to $\Delta\theta_i$. The selected update w^* is the one that has the smallest score.
- **Trimmed Mean [12].** Trimmed mean is a coordinate-wise aggregation policy which means that each model parameter is considered individually. For parameters of the updates and a trim fraction β, the largest β parameters, and the smallest β parameters are removed by the server. Then, the server computes the mean of the remaining $n-2\beta$ parameters as the corresponding parameter of the updated global model.
- **Median [12].** Median is another coordinate-wise aggregation policy. The median of the updated parameters is used as the corresponding parameter of the updated global model.
- **Bulyan [14].** Bulyan is a composite Byzantine aggregation method. Specifically, Bulyan first runs Krum multiple times to form a selection gradient set and then uses the coordinate-wise aggregation policy like Median to aggregate the selected gradients.
- **FLTrust [17].** FLTrust computes the trust score of each participant based on the cosine similarity between the gradients of the global root model trained by a root dataset which conforms to a standard data distribution on the server side and the local model.
- **GAAvernor [15].** The robust gradient aggregation agent utilizes the vanilla RL technique to allocate the corresponding weights, and a weight aggregation strategy to degrade the impacts of Byzantine attackers.

C. Secure Aggregation

The conventional secure aggregation scheme is implemented by a double-masking structure while the masks are generated by a pseudo-random generator (PRG). The pair of the participants i, j negotiates a pairwise secret s_{ij} and generates the perturbation $\mathrm{PRG}(s_{ij})$ by PRG. Meanwhile, each participant i creates a private random seed b_i to add a secondary perturbation $\mathrm{PRG}(b_i)$, which guarantees privacy in case the participant is identified as a dropped participant due to communication delays [25]. The final masked update $\Delta w_i'$ of a participant i is uploaded to the server.

$$\Delta w_i' = \Delta w_i + \mathrm{PRG}(b_i) + \sum_{j:i<j} \mathrm{PRG}(s_{ij}) - \sum_{j:i>j} \mathrm{PRG}(s_{ji}). \tag{2}$$

The participant i utilizes Shamir's secret sharing [35] to share b_i and the secrets $\{s_{ij}\}$ with other participants. To aggregate the participants' updates, the server collects the secret shares of pairwise secrets $\{s_{ij}\}$ to remove the masks of dropped participants and collects the secret shares of the private seeds $\{b_i\}$ to remove the masks of the surviving participants. The server aggregates the masked updates by:

$$\begin{aligned} \Delta w = & \sum_{i \in \mathcal{U}_1} (\Delta w_i' - \mathrm{PRG}(b_i)) \\ & - \sum_{i \in \mathcal{U}_2} \Big(\sum_{j:i<j} \mathrm{PRG}(s_{ij}) - \sum_{j:i>j} \mathrm{PRG}(s_{ji}) \Big) \\ = & \sum_{i \in \mathcal{U}_1} \Delta w_i', \end{aligned} \tag{3}$$

where $\mathcal{U}_1$ and $\mathcal{U}_2$ denote the group of surviving and dropped participants, respectively. In this paper, the dropped participants are not considered, and we focus on reducing the effects of the Byzantine participants. If we apply the conventional secure aggregation directly, the perturbations cannot be eliminated because the weights change the scale of the perturbations. Hence, a secure aggregation algorithm based on Lagrange interpolation is devised to handle the elimination issue.

D. Secret Sharing

In this paper, we leverage Shamir's secret sharing [35] and the naive multiplicative secret sharing mechanisms to compute the local perturbations privately. Especially, Shamir's secret sharing has the additive homomorphic property while the naive multiplicative secret sharing has the multiplicative homomorphic property.

Shamir's secret sharing is implemented based on the polynomial interpolation over finite fields. Specifically, a secret $\mathbb{S}$ is divided into n shares of secret $\{\mathbb{S}_1, \cdots, \mathbb{S}_n\}$. Any t shares can be used to reconstruct $\mathbb{S}$, while any set of at most $t-1$ shares cannot provide any information about $\mathbb{S}$. The principle of Shamir's secret sharing can be presented as follows: $\mathbb{S}$ can be represented as a specific element e_0 of finite field $GF(q)$ where q is greater than n, $t-1$ elements $e_1, \cdots, e_{t-1}$ can be randomly chosen to construct a polynomial $f(x) = e_0 + e_1 x + e_2 x^2 + \cdots + e_{t-1} x^{t-1}$. n points can be calculated from the curve of $f(x)$, and each party i is given a point $(x_i, f(x_i))$ as the secret share $\mathbb{S}_i$. Given any subset of t of the shares, the secret e_0 ($\mathbb{S}$) can be obtained using interpolation such as $e_0 = f(0) = \sum_{i=0}^{t-1} (f(x_i) \prod_{i \neq j}^{t-1} \frac{x_j}{x_j - x_i})$.

The naive multiplicative secret sharing can be designed with basic idea of (n, n)-secret sharing. Formally, we assume there is a prime p and multiplicative group $\mathbb{Z}_p^*$. The secret is denoted as $\mathbb{S}$. To share $\mathbb{S}$ with n parties and all parties are required to reconstruct $\mathbb{S}$, $n-1$ secret shares $\mathbb{S}_1, \mathbb{S}_2 \cdots, \mathbb{S}_{n-1} \in \mathbb{Z}_p^*$ are chosen at random, and the n^{th} secret share $\mathbb{S}_n = \mathbb{S} \times$

$\prod_{i=1}^{n-1} \mathbb{S}_i^{-1}$. Hence, the secret $\mathbb{S}$ can be reconstructed by $\mathbb{S} = \prod_{i=1}^{n} \mathbb{S}_i$.

The additive homomorphic property of Shamir's secret sharing means that the sum of two secrets can be obtained by adding the corresponding shares of each secret. For example, if $\mathbb{S}1$ and $\mathbb{S}2$ are two secrets shared utilizing Shamir's secret sharing, then $\mathbb{S}1+\mathbb{S}2$ can be reconstructed by adding $\mathbb{S}1_i+\mathbb{S}2_i$ for each party i. Similarly, the multiplicative homomorphic property of naive multiplicative secret sharing means that the product of multiple secrets can be obtained by multiplying the corresponding shares of each secret. In this paper, we set the threshold t to n of Shamir's secret in order to integrate naive multiplicative secret sharing.

III. Problem Statement

A. System Model

We consider a typical vehicular network, which incorporates multiple MEC nodes and distributed vehicles. Each MEC node can communicate with Vehicles in its coverage area through wireless infrastructure-to-vehicle (I2V) links. Here, each vehicle is assumed to be assigned with a dedicated subchannel with orthogonal resource blocks during communication with its corresponding MEC node [36], [37], thereby there exists no interference.

1) MEC nodes: Each MEC node serves as a central server within the FL training process in its coverage, which schedules nearby vehicles, allocates specific training tasks, and aggregates local updates from these vehicles. Additionally, MEC nodes are responsible for initializing the FL process and detecting Byzantine vehicles. Each MEC node maintains a reliable backhaul connection to the Internet and manages a standard root dataset [17].

2) Vehicles: Vehicles are equipped with built-in sensors for local data collection and possess certain computational resources to train local models. As FL participants, vehicles assist MEC nodes by performing allocated training tasks. Upon receiving training requests from MEC nodes, vehicles train the global model on their local data and subsequently upload the corresponding updates to the MEC nodes during each training round.

B. Security Model & Assumptions

1) Honest-but-curious MEC nodes: In the whole FL process in vehicular networks, MEC nodes are assumed to be honest-but-curious, which means that MEC nodes execute each step of the training process honestly, but they might be curious to extract private information of vehicles. According to [8], [9], MEC nodes can extract private original training data by exploiting the gradients uploaded by vehicles in FL.

2) Byzantine vehicles: An adversary can compromise a proportion ε of all vehicles, where $\varepsilon \in (0, 1)$. The objective of Byzantine vehicles is to produce malicious updates which can reduce the performance of the global model, such that the global model has a high test error rate for the test dataset. The compromised Byzantine vehicles can arbitrarily manipulate the updates to execute different Byzantine attacks, such as Gaussian random, LIE [33], LMP [32], OMP [34], and adaptive Byzantine attacks.

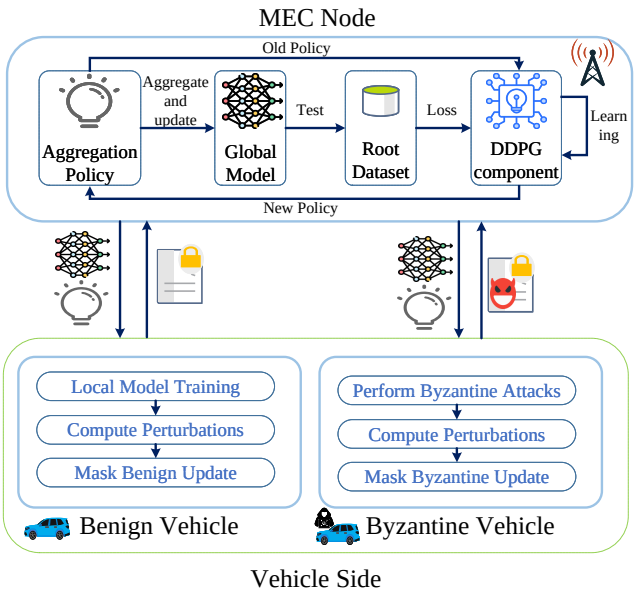


Fig. 1. The workflow of DRL-PBFL in vehicular networks.

3) Security assumption: We assume that Byzantine vehicles adhere to the training protocol established by MEC nodes; while any deviation from this protocol would disrupt the training process and render Byzantine vehicles susceptible to detection.

C. Defense Goals

The defense goals of DRL-PBFL are to prevent privacy inferences from the MECs and performance degradation caused by the Byzantine vehicle. We can formalize the defense goals as:

$$\begin{aligned} &\min \sum_{x \in \mathcal{D}_r} \mathcal{L}(\hat{\theta}_g(x)), \\ &s.t. \quad \hat{\theta}_g = \theta_g + \mathcal{A}(f(\Delta\theta'_1), ..., f(\Delta\theta'_c), f(\Delta\theta_{c+1}), ..., f(\Delta\theta_m)), \end{aligned} \tag{4}$$

where θ_g is the global model in each communication round, $\mathcal{D}_r$ is the root data set, $\mathcal{L}$ is the loss function, and $\mathcal{A}$ is the Byzantine-robust aggregation mechanism while f is the mask method applied by the vehicles to perturb the local updates. The first c updates are from the Byzantine vehicles. The MEC node can obtain an effective model for a specific task without being able to learn original updates of the vehicles. In this way, the profits of the MEC node and the privacy of vehicles are guaranteed simultaneously.

IV. Design of DRL-PBFL

In this section, we introduce DRL-PBFL which is designed from the weighting secure aggregation and DRL technique DDPG to achieve the objectives outlined in the section III-C.

A. Overview

Fig. 1 illustrates the workflow of DRL-PBFL in each communication round and Algorithm 1 gives the detailed steps.

Algorithm 1: DRL-PBFL.

Input: The number of communication rounds T, the number of local epochs E, the number of local batch size B, root dataset $\mathcal{D}_r$, the number of vehicles n.

Initialize: Aggregation policy $\mathcal{P} = \{\frac{1}{n}, ..., \frac{1}{n}\}$, global model θ_g, *DDPG* internal neural networks $\{\mu, \mu', Q, Q'\}$, t = 0.
while $t < T$ **do**
  MEC node:
  Broadcast the global model θ_g and the aggregation policy $\mathcal{P}$ to vehicles and await vehicles to upload local updates;
  Collect all the updates and aggregate them with $\mathcal{P}$ according to Equ. (8);
  Update the global model according to Equ. (9);
  Test the updated global model on root dataset and get the loss l_r according to Equ. (10);
  Get the new aggregation policy $\mathcal{P}'$ by inputting the state S to *DDPG* component according to Equ. (11);
  Update *DDPG*'s internal neural networks $\{\mu, \mu', Q, Q'\}$;
  Vehicles:
  for *each vehicle* **do**
    if *benign vehicle* **then**
      Optimize the local model according to Equ.(5);
      Compute the benign update $\Delta\theta_b$ according to Equ.(6);
    end
    else if *Byzantine vehicle* **then**
      Use Byzantine attacks to generate the Byzantine update $\Delta\theta_{Byzantine}$;
    end
    Generate a perturbation p_i and mask the update according to Equ.(7);
    Upload the final masked update $\Delta\theta_i'$;
  end
  $t = t + 1$;
end

Output: Trained global model $\theta_g^{(T)}$ and trained *DDPG* internal neural networks $\{\mu, \mu', Q, Q'\}$.

DRL-PBFL follows the common FL framework, and integrates a performance-based weighting secure aggregation algorithm and a DDPG component as the RL optimizer to update the aggregation policy. The aggregation policy is defined as the weights of local model updates corresponding to the vehicles, reflecting model quality and data quality. The performance of the global model on the root dataset $\mathcal{D}_r$ determines how to update the aggregation policy. As the training progress goes on, the benign vehicles are allocated higher weights, while the Byzantine vehicles are allocated lower weights. The weights of vehicles are optimized by RL based on the global model's performance on $\mathcal{D}_r$, so that all Byzantine attacks that aim at degrading the model performance fail to succeed. However, it is impossible to aggregate the local model updates if they have been perturbed by the local vehicles without considering the weights. The weights change the scale of the original perturbations at the aggregation step, and the simple addition cannot execute the elimination. Therefore, the conventional secure aggregation scheme [25] cannot work effectively. To meet the perturbation elimination challenge, a novel secure aggregation algorithm based on Lagrange interpolation is proposed. The algorithm generates erasable perturbations based on the current aggregation policy $\mathcal{P} = \{w_1, ...w_i, ..., w_n\}$.

The MEC node first initializes the aggregation policy $\mathcal{P}$, the global model θ_g, the root dataset $\mathcal{D}_r$, and the DDPG component. There are multiple communication interactions between the MEC node and n vehicles in the training process. Specifically, each training round mainly consists of the following steps:

- **Step 1:** The MEC node broadcasts the global model θ_g and the aggregation policy $\mathcal{P}$ to vehicles.
- **Step 2:** For the benign vehicles, they train the local model with their local dataset and upload the updates masked by adding perturbations. For the Byzantine vehicles, they generate Byzantine updates with the various Byzantine attacks and upload masked Byzantine updates. The perturbations are computed by Lagrange interpolation according to the current aggregation policy described in the section IV-B.
- **Step 3:** The MEC node receives all the masked updates and aggregates them with the aggregation policy $\mathcal{P}$. Then, the aggregated update is used to update the global model θ_g.
- **Step 4:** The MEC node evaluates the new global model θ_g' on the root dataset $\mathcal{D}_r$ and gets the corresponding loss l_r. The loss l_r, the old policy $\mathcal{P}$, and the auxiliary information $\mathcal{H}$ are input into the DDPG component, and the DDPG component outputs the new policy $\mathcal{P}'$.
- **Step 5:** The DDPG component updates its internal neural networks $\{\mu, \mu', Q, Q'\}$ through the underlying RL process.

For the local training, a benign vehicle first creates a local model θ_l that is identical to the original global model θ_g and uses the stochastic gradient descent (SGD) algorithm to optimize the θ_l as follows:

$$\theta_l' = \theta_l - \alpha \bigtriangledown_{\theta_l} J(\theta_l; x(i), y(i)), \tag{5}$$

where J is the loss function and α is the learning rate. Then, the benign update $\Delta\theta_b$ is computed as:

$$\Delta\theta_b = \theta_l' - \theta_g. \tag{6}$$

The Byzantine vehicle uses the Byzantine attacks to generate the malicious update $\{\Delta\theta_{Byzantine}\}$. Each vehicle then generates a perturbation p_i according to the aggregation policy through Lagrange interpolation to achieve secure aggregation as described in section IV-B. Each vehicle masks the update $\Delta\theta_i$ by adding the perturbation p_i and uploads the masked update $\Delta\theta_i'$:

$$\Delta\theta_i' = \Delta\theta_i + p_i. \tag{7}$$

The MEC node uses a linear weighting method and the current round aggregation policy $\mathcal{P}$ to compute the aggregated update $\Delta\theta$:

$$\Delta\theta = \sum_{i=1}^{n} w_i \Delta\theta_i', \tag{8}$$

$\Delta\theta$ is used to update the global model θ_g with a global learning rate α_g:

$$\theta_g' = \theta_g - \alpha_g \Delta\theta. \tag{9}$$

The following equation computes the loss l_r of θ_g' on $\mathcal{D}_r$:

$$\begin{aligned} l_r &= \frac{1}{N} \sum_{i=1}^{N} L(y_i, \theta_g'(x_i)), \\ &(x_i, y_i) \in \mathcal{D}_r. \end{aligned} \tag{10}$$

The loss l_r, the old policy $\mathcal{P}$, and the auxiliary information $\mathcal{H}$ are concatenated to the state S, which is the input of the DDPG component. The auxiliary information $\mathcal{H}$ is used to assist the DDPG component in generating reasonable aggregation policies at the beginning of each RL round, $\mathcal{H}$ can be the vehicles' historical scores or the MEC node's subjective assessment. Then, the DDPG component outputs a new aggregation policy $\mathcal{P}'$.

$$\begin{aligned} \mathcal{P}' &= DDPG(S), \\ S &= (l_r, \mathcal{P}, \mathcal{H}). \end{aligned} \tag{11}$$

As the training process goes on, the DDPG component updates the aggregation policy $\mathcal{P}$ that degrades the influences of Byzantine vehicles and enhances the influences of benign vehicles in the aggregation step. The underlying RL process is described in section IV-C.

B. Secure Aggregation Algorithm based on Lagrange Interpolation

We propose a novel secure aggregation algorithm based on Lagrange interpolation which generates erasable perturbations according to the current aggregation policy $\mathcal{P} = \{w_i\}$. Specifically, we assume there are n vehicles, and each vehicle receives the aggregation policy $\mathcal{P}$. These vehicles first agree on an $(n-1)$-degree polynomial $f(x)$ with the constant term of 0:

$$f(x) = c_1 x^{n-1} + c_2 x^{n-2} + \cdots + c_{n-2} x^2 + c_{n-1} x, \tag{12}$$

where $c_1, \ldots, c_{n-1}$ are the parameters of $f(x)$. Then, each vehicle i chooses a secret parameter s_i which satisfies:

$$w_i = c_1 s_i^{n-1} + c_2 s_i^{n-2} + \cdots + c_{n-2} s_i^2 + c_{n-1} s_i. \tag{13}$$

According to the Lagrange interpolation, the Lagrange basis polynomial $p_i(x)$ can be written as:

$$\begin{aligned} p_i(x) &= \prod_{j \in \mathcal{B}_i} \frac{x - s_j}{s_i - s_j}, \\ \mathcal{B}_i &= \{j | j \neq i, j \in \mathcal{D}_n\}, \end{aligned} \tag{14}$$

where $\mathcal{D}_n$ is the set of the vehicles' serial numbers. When x is set to 0, $p_i(0)$ is the target perturbation.

$$p_i = p_i(0) = \prod_{j \in \mathcal{B}_i} \frac{-s_j}{s_i - s_j}. \tag{15}$$

To further prevent potential collusion between the MEC node and the vehicles, the secret parameters $\{s_i\}$ cannot be directly transmitted between vehicles. We utilize the Shamir secret sharing (which is additive homomorphic) and the naive multiplicative secret sharing (which is multiplicative homomorphic) to compute p_i of each vehicle i privately. Specifically, for each vehicle i, it first selects a local mask scale parameter η_i to mask s_i. Then, vehicle i generates the Shamir secret shares $\{j, s_{i,j}\}$ of s_i and the multiplicative secret shares $\{j, \eta_{i,j}\}$ of η_i for other vehicles. Next, the secret shares and the masked $\eta_i s_i$ are transmitted to the corresponding vehicles. Vehicle k utilizes the additive homomorphic property of Shamir secret sharing and computes $g_{i,j}^k$ by:

$$g_{i,j}^k = s_{i,k} - s_{j,k}. \tag{16}$$

The secret shares $\{g\}$ are sent to a trusted vehicle denoted as l, which is considered not to collude with the MEC node by other vehicles. l receives enough secret shares about $s_i - s_j$ and can reconstruct the original secret difference $s_i - s_j$ as a relay node. Then l computes the products of all secret differences as $\prod_{j \in \mathcal{B}_i} s_i - s_j$ and sends it to the corresponding vehicle i. Therefore, the masked $\hat{p}_i$ can be computed by vehicle i:

$$\hat{p}_i = \prod_{j \in \mathcal{B}_i} \frac{-\eta_j s_j}{s_i - s_j}, \tag{17}$$

To obtain the original perturbation p_i, the product of scale parameters $\prod_{j \in \mathcal{B}_i} \eta_j$ needs to be removed. Each vehicle i leverages the multiplicative homomorphic property of the naive multiplicative secret sharing to compute $\tau_{i,j}$:

$$\tau_{i,j} = \prod_{k \in \mathcal{B}_j} \eta_{k,i}. \tag{18}$$

$\tau_{i,j}$ are sent to the corresponding vehicles. In this way, vehicle i can reconstruct the original value of $\prod_{j \in \mathcal{B}_i} \eta_j$. Finally, each vehicle i can compute the perturbation $p_i = \hat{p}_i / \prod_{j \in \mathcal{B}_i} \eta_j$ without revealing the secret parameter s_i, thereby preventing the collusion between the MEC node and the vehicles.

Then, p_i is added to the update to mask the original update in Equ. (7). These perturbations are erased when the MEC node aggregates the masked updates.

$$\begin{aligned} \Delta\theta &= \sum_{i=1}^{n} w_i \Delta\theta_i' \\ &= \sum_{i=1}^{n} w_i (\Delta\theta_i + p_i) \\ &= \sum_{i=1}^{n} w_i \Delta\theta_i + \sum_{i=1}^{n} w_i p_i \\ &= \sum_{i=1}^{n} w_i \Delta\theta_i. \end{aligned} \tag{19}$$

Since the constant term of $f(x)$ is 0, the term $\sum_{i=1}^{n} w_i p_i$ is 0 according to the properties of the Lagrange interpolation.

Theorem 1: The perturbations $\{p_i\}$ added by vehicles can be eliminated by the aggregation policy $\mathcal{P} = \{w_i\}$ correctly.

Proof of Theorem 1: Given a data point set $\{(x_j, y_j)\}$, where any two x_j are not the same, the Lagrange interpolation polynomial $L_n(x)$ is a linear combination as

$$L_n(x) = \sum_{j=1}^{n} y_j l_j(x). \tag{20}$$

Here, $l_j(x)$ is the Lagrange basis polynomial.

$$\begin{aligned} l_j(x) &= \prod_{k \in \mathcal{B}_j} \frac{x - x_k}{x_j - x_k}, \\ \mathcal{B}_j &= \{k | k \neq j, k \in \mathcal{D}_n\}. \end{aligned} \tag{21}$$

For each x_i, $l_j(x_i)$ satisfies:

$$l_j(x_i) = \begin{cases} 1, \text{ if } i = j \\ 0, \text{ if } i \neq j. \end{cases} \tag{22}$$

Then, we can have:

$$L(x_i) = \sum_{j=1}^{n} y_j l_j(x_i) = y_i. \tag{23}$$

If we view a data pair set $\{(w_j, s_j)\}$ of weights and secrets as the points in the Lagrange interpolation formula, then, for each secret parameter s_i, we have

$$L_n(s_i) = \sum_{j=1}^{n} w_j l_j(s_i) = w_i. \tag{24}$$

The equation means L_n interpolates the $(n-1)$-degree polynomial function $f(x)$ in Equ. (12) exactly. According to the geometric meaning of Lagrange interpolation formula and the constant term of the consensus-based $(n-1)-$degree polynomial $f(x)$ equals 0, we have:

$$L_n(0) = \sum_{j=1}^{n} w_j l_j(0) = \sum_{i=1}^{n} w_i p_i = 0. \tag{25}$$

$l_j(0)$ equals to p_i in Equ. (15) and the above equation means that the interpolation polynomial goes through the point $(0, 0)$. In this way, the perturbations $\{p_i\}$ in Equ. (19) are eliminated during the aggregation process.

C. Aggregation Policy Update via DDPG

One of the key issues to be solved in DRL-PBFL is how to optimize the aggregation policy $\mathcal{P}$, so that each vehicle's weight w_i effectively reflects its model and data quality, while mitigating the effect of Byzantine vehicles and increasing the effect of benign vehicles on the aggregated update $\Delta\theta$. In this way, the defense goals in Equ. (4) are achieved. RL is an effective general artificial intelligence method to output an optimized action according to the current environment states, or called policy here. Deep Q Network (DQN) is one of the most commonly used algorithms in RL, it can solve problems with high-dimensional observation spaces [38]. However, DQN cannot be applied in DRL-PBFL because it only can handle discrete action spaces, while the aggregation policy $\mathcal{P}$ is continuous. In this paper, we use the deep deterministic policy gradient (DDPG) [39] algorithm to update the aggregation policy. DDPG is an actor-critic algorithm that can handle high-dimensional, continuous action spaces. DDPG can effectively update the aggregation policy according to the performance of the global model. Particularly, the presence of non-IID data distribution among vehicles and the effects of Byzantine vehicles ultimately deteriorate the performance of the global model on the root dataset, with performance being the most critical factor in the optimization process of the aggregation policy. Therefore, DRL-PBFL achieves more satisfactory performances in the non-IID scenarios than other schemes.

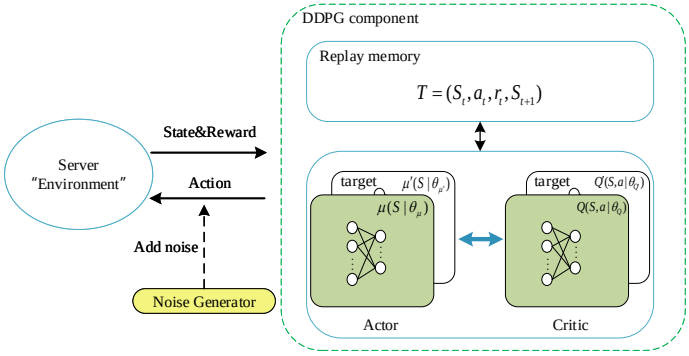


Fig. 2. The diagram of the DDPG component.

The DDPG component uses the general DDPG architecture, including the actor network $\mu(S|\theta_\mu)$, the target actor network $\mu'(S|\theta_{\mu'})$, the critic network (Q network) $Q(S, a|\theta_Q)$, the target critic network $Q'(S, a|\theta_{Q'})$, a random process noise generator $\mathcal{N}_t$, and the experience replay buffer $\mathcal{R}$. Fig. 2 illustrates the architecture of the DDPG component. S and a denote the state and the action, respectively, while θ denotes the parameters of networks in the DDPG component. We model the update process of the aggregation policy as a Markov decision process with a state space $\mathbb{S}$, an action space $\mathbb{A}$, a reward function R, an initial state distribution $P(S_1)$, and transition dynamics $P(S_{t+1}|S_t, a_t)$, the subscript t denotes the communication round t. The state S_t at a communication round t refers to a tuple $S_t = (l_r, \mathcal{P}, \mathcal{H})$ as Equ. (11) and (10). The action a is an n-dimension vector, while n is the number of vehicles. The output action a can be converted to the aggregation policy $\mathcal{P}$ easily via normalization. The reward function R reflects the changes of states and provides heuristics for the aggregation policy update. The reward r_t for the round t is determined by the loss of the global model $l_r(\theta'_g)$ in this round, the loss of the global model $l_r(\theta_g)$ in the last round and the loss of the current best model $l_r(\theta_g^{best})$.

$$r_t = (l_r(\theta_g) - l_r(\theta'_g)) - (l_r(\theta'_g) - l_r(\theta_g^{best})), \tag{26}$$

the loss l_r is computed according to Equ. (10), the best model θ_g^{best} is recorded during the training process, and θ_g^{best} denotes the current global model with the lowest test loss on the root dataset $\mathcal{D}_r$. For the initial state distribution $P(S_1)$, the term $\mathcal{P}$ is initialized as $w_1 = w_2 = \cdots = w_n$ and $w_1 + w_2 + \cdots + w_n = 1$, while the term l_r depends on the random initialization of model parameters. For $P(s_{t+1}|S_t, a_t)$, the term $\mathcal{P}$ of S_{t+1} is

the normalized a_t while the term l_r depends on how the global model updates.

The whole DDPG policy update process is divided into two phases. The first phase is to fill the experience replay buffer R. There are two possible cases for the filling process: if there is history information of the same task and the same vehicles, the information can be processed directly into the data format required by the buffer and filled into the buffer; or there is a warm-up period to fill the replay buffer. For the warm-up period, the action a_t is generated by the actor network $\mu(S|\theta_\mu)$ according to the current state S_t and exploration noise $\mathcal{N}_t$.

$$a_t = \mu(S_t|\theta_\mu) + \mathcal{N}_t, \tag{27}$$

The action a_t is used to update the aggregation policy $\mathcal{P}$. Then, the reward r_t and the next state S_{t+1} are observed and computed as Equ. (26) and Equ. (10), respectively. In this way, we have a transition tuple T as

$$T = (S_t, a_t, r_t, S_{t+1}), \tag{28}$$

T is stored in the experience replay buffer $\mathcal{R}$. The exploration noises $\mathcal{N}_t$ can generate random policies, policies in which assigning high weights to benign vehicles and low weights to Byzantine vehicles would result in experiences with high rewards, and vice versa. These experiences in $\mathcal{R}$ help the DDPG component to train the neural networks and identify Byzantine vehicles among the vehicles. When the size of $\mathcal{R}$ reaches a certain threshold, the warm-up period is over, and the action a_t selection no longer involves exploration noise.

$$a_t = \mu(S_t|\theta_\mu). \tag{29}$$

During each communication round, a mini-batch with N transition tuples $\{(S_i, a_i, r_i, S_{i+1})\}$ is randomly sampled from $\mathcal{R}$ to update the actor network and the critic network. The next Q value v_i for the tuple (S_i, a_i, r_i, S_{i+1}) is output by the target critic network θ'_Q as:

$$v_i = r_i + \gamma Q'(S_{i+1}, \mu'(S_{i+1}|\theta_{\mu'})|\theta_{Q'}), \tag{30}$$

where $\{v_i\}$ is called the next Q values batch. We can update the critic network θ_Q by minimizing the loss between the Q values batch and the next Q values batch as:

$$L = \frac{1}{N}\sum_i (v_i - Q(S_i, a_i|\theta_Q)). \tag{31}$$

The policy gradient algorithm [40] is used to update the actor network μ. According to [39], the policy gradient of μ can be estimated as:

$$\bigtriangledown_{\theta_\mu} J \approx \frac{1}{N}\sum_i \bigtriangledown_a Q(S_i, a_i|\theta_Q) \bigtriangledown_{\theta_\mu} \mu(S_i|\theta_\mu). \tag{32}$$

The Adam optimizer [41] is used to update the actor network μ and the critic network Q. Finally, the target actor network μ' and the target critic network Q' are soft updated:

$$\begin{aligned} \theta_{Q'} &= \tau\theta_Q + (1-\tau)\theta_{Q'}, \\ \theta_{\mu'} &= \tau\theta_\mu + (1-\tau)\theta_\mu, \end{aligned} \tag{33}$$

where τ is set to 0.001.

In general, multiple episodes are required to complete the convergence of the internal neural network of DDPG and the global model θ_g of FL. Once θ_g converges, we get the final aggregation policy $\mathcal{P}_{final}$ as:

$$\mathcal{P}_{final} = \psi(a_{final}) = \psi(\mu((l_r(\theta_g), \mathcal{P}_{final-1}))), \tag{34}$$

where ψ denotes the normalization function.

We prove the Byzantine robustness of the FL scheme theoretically when the Byzantine ratio is fixed as ε, which means that the loss differences between the converged global model θ_g and the global optimal model θ^* are bounded. We assume the loss function $\mathcal{L}(\cdot)$ is convex and $\beta-$smooth. Then, $\mathcal{L}(\cdot)$ is $\beta-$smooth if $|\mathcal{L}(\theta_1) - \mathcal{L}(\theta_2)| \leq \beta\|\theta_1 - \theta_2\|_2$ for $\forall\theta_1, \theta_2 \in \Theta$, where Θ denotes the parameter space. Besides, we use U to denote the upper bound of gradient norm, that is, $\| \bigtriangledown_\theta \mathcal{L}(\theta)\|_2 \leq U$.

First, we consider the situation when $\mathbb{D}_r = \mathbb{D}$, where $\mathbb{D}_r$ denotes the distribution of $\mathcal{D}_r$ and $\mathbb{D}$ denotes the distribution of local datasets. We can have Lemma 1:

Lemma 1: For a fixed Byzantine ratio ε and the root dataset $\mathcal{D}_r$ that satisfies $\mathbb{D}_r = \mathbb{D}$, the loss difference upper bound between the global model $\theta_g^{(t_s)}$ with t_s steps gradient descent and the optimal global model $\theta_g^{(*)}$ satisfies:

$$\mathcal{L}(\theta_g^{(t_s)}) - \mathcal{L}(\theta_g^{(*)}) \leq \frac{2RU}{\sqrt{(1-\varepsilon)nt_s}} + \frac{R^2B\beta}{t_s}, \tag{35}$$

where R is the diameter of parameter space, B is the local batch size for local model training, and t_s is the steps of gradient descent.

Proof of Lemma 1: For a special RL episode, we assume that the DDPG neural networks are converged and the DDPG component gives an optimal aggregation policy $\mathcal{P}^*$ at the beginning, where the benign vehicles are allocated high weights and the Byzantine vehicles are allocated low weights. Then, for gradient descent steps t_s, following the [42], we can have the gradient variation as:

$$\mathbb{E}\| \sum_i w_i \bigtriangledown_\theta \mathcal{L}(\theta_i^{(t_s)}) - \bigtriangledown_\theta \mathcal{L}(\theta_g^{(t_s)})\|_2^2 \leq \frac{2U^2}{(1-\varepsilon)nB}. \tag{36}$$

$(1-\varepsilon)nB$ denotes all samples trained in the form of mini-batch in a gradient descent step. According to the Theorem 6.3 in [42], we have the Equ. (35) and prove the Lemma. 1.

Then, we extend the situation when $\mathbb{D}_r \neq \mathbb{D}$ and there exists a Kullback-Leibler (KL) divergence $D_{KL}(\mathbb{D}_r\|\mathbb{D})$, and we give the following theorem:

Theorem 2: For a fixed Byzantine ratio ε and KL divergence $D_{KL}(\mathbb{D}_r\|\mathbb{D})$, the loss difference upper bound between the global model $\theta_g^{(t_s)}$ with t_s steps gradient descent and the optimal global model $\theta_g^{(*)}$ satisfies:

$$\begin{aligned} &\mathcal{L}(\theta_g^{(t_s)}) - \mathcal{L}(\theta_g^{(*)}) \leq \\ &\frac{2RU}{\sqrt{(1-\varepsilon)nt_s}} + \frac{R^2B\beta}{t_s} + \sqrt{2}\|\mathcal{L}\|_\infty \sqrt{D_{KL}(\mathbb{D}_r\|\mathbb{D})}. \end{aligned} \tag{37}$$

Proof of Theorem 2: We compute the loss difference between the $\mathcal{D}_r$ and $\mathcal{D}$ in the form of expectation for $\theta_g^{(t_s)}$ as:

$$|\mathbb{E}_{x\sim\mathcal{D}}[\mathcal{L}(\theta_g^{(t_s)})] - \mathbb{E}_{x\sim\mathcal{D}_r}[\mathcal{L}(\theta_g^{(t_s)})]| \leq \|\mathcal{L}\|_\infty \int_x d|\mathcal{D} - \mathcal{D}_r|(x). \tag{38}$$

Apply the Pinsker's inequality [43]:

$$\|\mathcal{L}\|_\infty \int_x d|\mathcal{D} - \mathcal{D}_r|(x) \leq \|\mathcal{L}\|_\infty \frac{\sqrt{2}}{2}\sqrt{D_{KL}(\mathcal{D}_r\|\mathcal{D})}. \quad (39)$$

For the $\theta_g^{(*)}$ we can have a similar inequality:

$$|\mathbb{E}_{x\sim\mathcal{D}}[\mathcal{L}(\theta_g^{(*)})] - \mathbb{E}_{x\sim\mathcal{D}_r}[\mathcal{L}(\theta_g^{(*)})]| \leq \|\mathcal{L}\|_\infty \frac{\sqrt{2}}{2}\sqrt{D_{KL}(\mathcal{D}_r\|\mathcal{D})}. \quad (40)$$

Combining the Lemma 1 and Equ. (39-40), we prove the Theorem 2.

The theoretical analysis suggests that a smaller Byzantine ratio β and a lower KL-divergence $D_{KL}(\mathbb{D}_r\|\mathbb{D})$ lead to a better optimum.

V. Performance Evaluation

A. Experiments Setup

Datasets. Three ML benchmark datasets are used to evaluate the performance of DRL-PBFL.

- *CIFAR10 [44]* is a benchmark image recognition dataset in the ML area. CIFAR10 contains 10 categories including cat, bird, airplane, etc. There are 60,000 triple-channel RGB 32×32 images, 10,000 of which are test examples. Each category contains 6,000 images, and the data distribution is balanced.
- *MNIST [45]* is the most general handwritten digits recognition dataset used in ML research. MNIST contains gray-scale images of digits in 10 categories, ranging from 0 to 9. The dataset includes 60,000 grey-scale training images and 10,000 test images, all sized 28×28 pixels. The digits are located in the center of the image.
- *Fashion-MNIST [46]* is a novel 10-class classification dataset of fashion products such as dresses, shirts, and coats. Fashion-MNIST consists of 70,000 28×28 grayscale images, 10,000 of which are test images. The data distribution of this dataset is balanced, each category contains 6,000 training images and 1,000 test images.

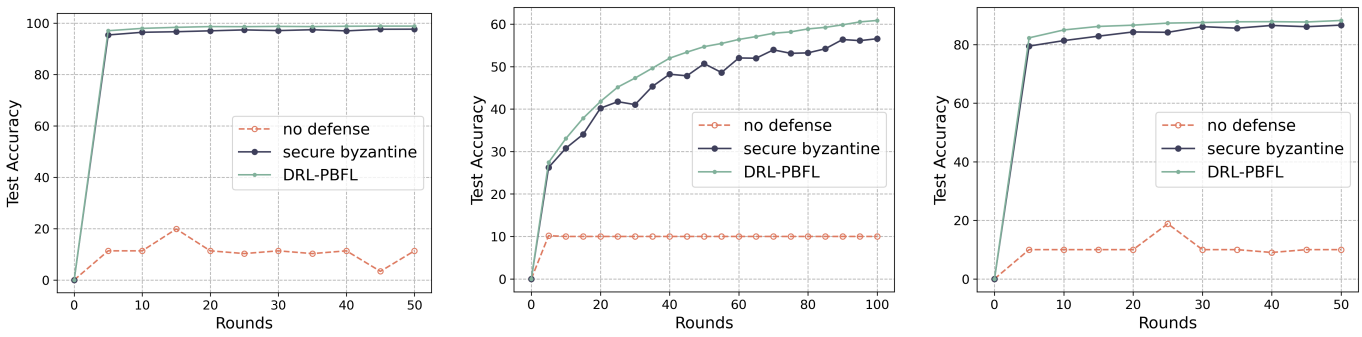
Environment Simulation. In this paper, we mainly consider an environment with a fixed range of 5 kilometers by 5 kilometers. An MEC node serves as the aggregation server, and 20 vehicles participate in the federated learning process. The number of Byzantine vehicles ranges from 0 to 5, while the others are benign vehicles.

Model architectures and hyperparameters. In the experiments, the models with different architectures are implemented to meet the needs of different classification tasks. For the classification models, CNN models with different layers are implemented for MNIST, Fashion-MNIST, and CIFAR10 tasks, respectively. The number of layers is 4 for MNIST and Fashion-MNIST, and 5 for CIFAR10. For the DDPG component, the actor model and the critic model are DNN models with multiple linear layers. We train the MNIST and the Fashion-MNIST model using a training batch size of 128, an SGD optimizer with a decay learning rate which is set to 0.05 initially, the global rounds number is set to 50, and the local epochs number is set to 1. Similarly, we train the CIFAR10 model using a training batch size of 128, an SGD optimizer with a decay learning rate which is set to 0.01 initially, the local epochs number is in $[1, 3, 5]$ for different settings, and the global rounds number is 100.

Attacks. In the experiments, we mainly focus on five types of Byzantine attacks:

- *Gaussian random attack.* This attack randomly crafts the updates of the malicious vehicles to random noises. Specifically, a Gaussian distribution with a mean value μ and a variance σ is estimated according to the target task, and malicious vehicles randomly sample numbers from the Gaussian distribution as their updates. Gaussian random attack degrades the global model effectively, but it can be well defended by the general Byzantine aggregation mechanisms, such as Krum. In the experiments, the mean is determined by the benign updates, so that the magnitude of the mean does not differ much from benign updates. For instance, the mean is set to 0.1 and the variance is set to 2×10^{-6} for MNIST.
- *LMP attack.* [32] The LMP attack is an optimization-based model poisoning attack that degrades the model's performance. The LMP attack can be tailored to different Byzantine aggregation mechanisms. For secure aggregation based on PRG in [25], which is the most general secure aggregation mechanism in federated learning, the full-knowledge LMP attack can be implemented. Even if it is difficult to implement a full knowledge attack, the partial knowledge attack can be implemented without benign vehicles' updates. In this paper, we focus on the LMP attack tailored to Krum, because Krum is the common aggregation mechanism in Byzantine-robust privacy-preserving federated learning schemes. Specifically, the adversary computes the mean μ_b of the owned benign updates and the sign of the malicious update $-sign(\mu_b)$ which is opposite to the sign of μ_b. Then, the adversary computes the final malicious update by $\Delta\theta_m = -\lambda sign(\mu_b)$. For the Krum-tailored LMP attack, the attack optimizes the value of λ until the Krum chooses the malicious update $\Delta\theta_m$.
- *LIE attack.* [33] The LIE attack prevents the global model from convergence by adding small amounts of noises on each dimension of the average of benign updates from other vehicles. The LIE attack degrades the performance of the final global model and evades the detection of the Krum aggregation policy, which means that the malicious updates are always chosen by the Krum policy in each communication round. Specifically, the adversary computes the mean value μ and the standard variation σ of owned benign vehicles' updates and computes the coefficient z based on the cumulative standard normal function $\phi(z)$, the number of all vehicles, and the number of malicious vehicles. The final malicious update is $\mu + z\sigma$.
- *OMP attack.* [34] The OMP attack is a modified version of the LMP attack which optimizes its core coefficient λ. The OMP attack gives a more effective λ than the LMP attack. In this paper, we focus on attacking the Krum aggregation policy and implementing the Krum-tailored

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2024.3524834



(a) MNIST (b) CIFAR10 (c) Fashion-MNIST

Fig. 3. Convergence results in the Gaussian attack settings for three datasets.

OMP attack.

- *Adaptive Byzantine attack.* The aggregation policies are transmitted to the vehicles by the MEC node in each communication round and the malicious vehicle can adjust the poisoning update according to his/her corresponding weight. As with the above attacks, we assume that the attacker can implement a full-knowledge attack or a partial-knowledge attack. Firstly, the attacker computes the mean of owned benign update μ_b. The malicious update can be computed by $\Delta\theta_m = max(-sign(\mu_b) * \mu_b * \frac{\gamma}{w_m}, -sign(\mu_b) * \mu_t)$. γ is the parameter determined by the vehicles' number, w_m is the weight of the attacker in dispatched policy. μ_t is a threshold value to prevent the MEC node from detecting and excluding the malicious update, here, it's set to 10 times the average update μ_b.

Non-IID setting. In this paper, in addition to experiments under the IID settings, experiments are also conducted under the non-IID settings to further demonstrate the superior performance of DRL-PBFL. The Dirichlet non-IID is implemented across vehicles following [47]. The parameter β determines the degree of non-IID. The lower β, the higher degree of non-IID. Generally, β is set to 0.1 to show the behaviors of schemes in the highly non-IID case.

Comparison schemes. The FedAvg baseline [1], the statistics-based Byzantine-robust schemes (e.g., Krum [4], Median [12], etc.), FLtrust [17], and the secure Byzantine-robust schemes in [23], [24] are compared to DRL-PBFL to demonstrate that the proposed scheme outperforms existing Byzantine-robust FL schemes.

B. Defense Results against General Byzantine Attacks

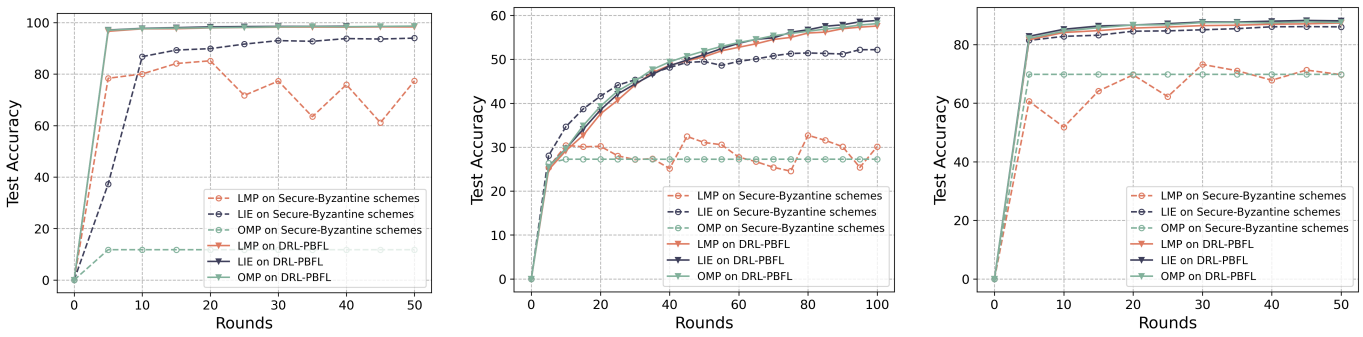
In this subsection, we implement the Gaussian attack, which is the most general Byzantine attack, to evaluate the defense capability of DRL-PBFL. As a comparison, we implement the secure Byzantine-robust FL scheme based on Krum [23], [24]. This scheme and DRL-PBFL can achieve Byzantine aggregation and secure aggregation at the same time. Besides, FL without defense mechanisms is used as a baseline scheme for comparison. For Gaussian attack settings, there are 15 benign vehicles and 5 malicious vehicles. General ML parameters in Gaussian attack settings are the same as the non-adversary settings. For MNIST and Fashion-MNIST, the malicious vehicles choose the Gaussian noise whose mean equals 0.1 and variation equals 2×10^{-6}; for CIFAR10, the mean is set to 0.01, and the variation is 2×10^{-6}.

The Gaussian attacks prevent the baseline scheme from convergence, while the secure byzantine scheme based on Krum and DRL-PBFL defends against Gaussian attacks well. As shown in Fig. 3(a), the no-defense baseline scheme's test accuracy on MNIST fluctuates at 10%, which is the random guess accuracy for a 10-class classification task. The Krum-based secure Byzantine FL scheme, represented by the blue curve in the figure, is close to convergence within 10 communication rounds and only results in a negligible performance loss. For comparison, the convergence rate and model performance of DRL-PBFL are higher than that of the secure Byzantine scheme. The results demonstrate that DRL-PBFL can effectively defend against general Byzantine attacks, such as Gaussian attacks, and has a lower performance loss than the existing state-of-the-art secure Byzantine FL schemes. Fig. 3(b) shows the convergence results on CIFAR10 in Gaussian attack settings. Compared to MNIST, the no-defense scheme has smaller fluctuations around the random guess accuracy (10%). The blue line represents the relationship between test accuracy and communication rounds of Krum-based secure Byzantine-robust scheme. We observe that the model performance loss of the secure Byzantine scheme is relatively large on CIFAR10 and the convergence curve is very unstable. By contrast, the green line which represents our proposed scheme is smooth, which means that the updates from most benign vehicles are involved, and the effects of updates from Byzantine vehicles are reduced in DRL-PBFL. Fig. 3(c) plots the convergence curves on Fashion-MNIST in Gaussian attack settings. In general, the performance of different schemes on Fashion-MNIST is similar to that on MNIST, and DRL-PBFL has the best defense capability and classification performance. The results demonstrate that DRL-PBFL not only achieves secure aggregation to guarantee privacy from the honest-but-curious MEC node but also defends against general Byzantine attacks effectively with a better model performance.

C. Defense Results against State-of-the-art Tailored Byzantine Attacks

In this subsection, we implement several state-of-the-art tailored byzantine attacks targeting on Krum aggregation

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2024.3524834



(a) MNIST (b) CIFAR10 (c) Fashion-MNIST

Fig. 4. Convergence results in the tailored Byzantine attack settings for three datasets.

mechanism, such as LMP attack, LIE attack, and OMP attack. Similar to the previous subsection, we implement the secure Byzantine-robust FL scheme based on Krum for comparison. We use the three tailored Byzantine attacks to attack the secure Byzantine-robust scheme and DRL-PBFL, respectively. The experimental results show that DRL-PBFL has a better performance than the state-of-the-art scheme while defending against Krum-tailored Byzantine attacks. The tailored attack settings are mostly the same as the previous non-adversary and Gaussian attack settings.

Fig. 4(a) shows the attack results on MNIST, the dotted lines refer to the convergence curves of the Krum-based secure Byzantine FL scheme under tailored attacks, while the solid lines refer to the convergence curves of DRL-PBFL under tailored attacks. We observe that the three tailored attacks degrade the performance of the Krum-based secure-Byzantine schemes obviously: the LIE attack degrades the accuracy by about 10%, the LMP attack degrades the accuracy by about 20% to 40%, and for OMP attack, the accuracy degrades to the random guess. However, for all three attacks, there are nearly no attack impacts on DRL-PBFL, the three convergence curves under different attacks almost coincide. Fig. 4(b) and Fig. 4(c) illustrate the attack results on CIFAR10 and Fashion-MNIST respectively. For CIFAR10, the impacts of the LMP and the OMP attacks on the Krum-based schemes are about 30% while the impact of the LIE attack is about 10%. By contrast, the impacts of three attacks on DRL-PBFL are less than 5%. For Fashion-MNIST, the impact of the LMP attack and OMP attackon the Krum-based scheme is about 15%, and the LIE attack has a negligible impact on the Krum-based schemes. However, similar to MNIST, the convergence curves of DRL-PBFL under the three attacks almost coincide and these attacks have no significant impact on DRL-PBFL. In summary, these results demonstrate our proposed scheme can effectively defend against tailored attacks, while existing Krum-based secure Byzantine schemes cannot defend.

At the same time, we implement the LMP attacks on different schemes in the case of the non-IID Fashion-MNIST dataset. Fig. 5 shows the attack results. The experimental results demonstrate that all existing Byzantine-robust schemes except DRL-PBFL suffer from performance degradation due to non-IID. Specially, the existing secure Byzantine schemes

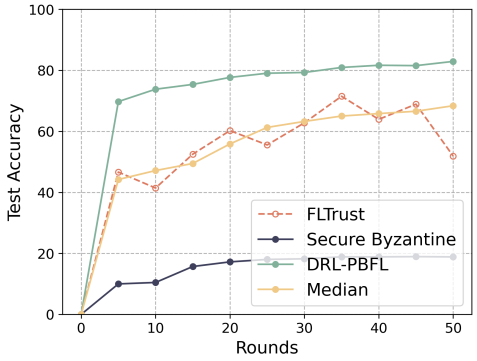


Fig. 5. Convergence results under LMP attacks for non-IID Fashion-MNIST.

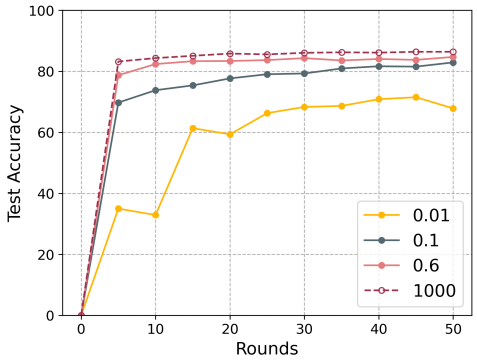


Fig. 6. Convergence results of different β (Fashion-MNIST).

are based on the Krum scheme [23], [24], however, the Krum scheme is significantly affected in the non-IID scenario, which means that the secure Byzantine schemes cannot complete the convergence task in this case. It is worth noting that FLTrust, which can effectively defend against tailored attacks, is also significantly influenced by non-IID. The superiority of DRL-PBFL is demonstrated by its ability to defend against tailored Byzantine attacks effectively in a non-IID setting while maintaining vehicles' privacy.

D. Defense Results against Adaptive Byzantine Attacks

In this subsection, we implement the adaptive Byzantine attacks and evaluate the attack impacts on the model trained by DRL-PBFL. Meanwhile, the accuracy results of non-adversary settings and Gaussian attack settings are compared with the

adaptive attack results to prove the defense performance of DRL-PBFL. The experimental results demonstrate that DRL-PBFL can still effectively defend against Byzantine attacks and complete target task training under the adaptive attack settings. The parameter settings are the same as the previous settings.

Fig. 8 illustrates the accuracy results in the adaptive Byzantine attack settings for MNIST, Fashion-MNIST, and CIFAR10. The final accuracy of the proposed DRL-PBFL is close to each other under different experimental settings for all three datasets. For MNIST and Fashion-MNIST, the models' accuracy under adaptive attacks is slightly lower than that under Gaussian attacks; and both models are less accurate than the baseline model trained in the non-adversaries settings. For CIFAR10, the model accuracy under adaptive attacks exceeds the model accuracy under Gaussian attacks and approaches the baseline. In summary, DRL-PBFL can effectively defend against adaptive Byzantine attacks in various experimental settings, further demonstrating the Byzantine robustness of the proposed scheme.

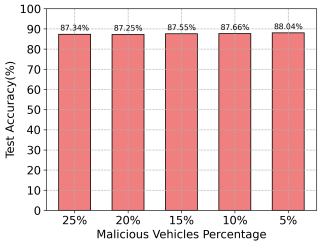


Fig. 7. Impacts of number of malicious vehicles.

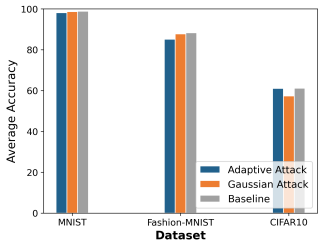


Fig. 8. Adaptive attack results.

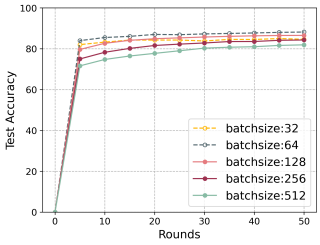


Fig. 9. Convergence curves of different local batch sizes (Fashion-MNIST).

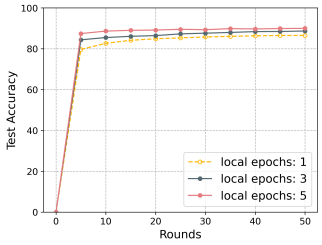


Fig. 10. Convergence curves of different local epochs (Fashion-MNIST).

E. Effect of Different Parameters

In this subsection, we evaluate the effects of different parameters in the DRL-PBFL training process by adjusting some important learning parameters and adversarial parameters. All experiments in this subsection are performed on the Fashion-MNIST classification task. The adversary launches the LMP attack. Except for the changed parameters, the remaining parameters are the same as the adversary settings in Sect. V-C.

Effect of the number of malicious vehicles. We change the malicious vehicles number in $[1, 2, 3, 4, 5]$, which refers to percentages $[5\%, 10\%, 15\%, 20\%, 25\%]$. Fig. 7 illustrates the effect of the number of malicious vehicles on Fashion-MNIST. From Fig. 7, we can observe that the model performance is hardly affected by different percentages of malicious vehicles. The results in Fig. 7 demonstrate that within a specific range, the proposed DRL-PBFL approach effectively mitigates the impact of malicious vehicles, regardless of their quantity.

Effect of the non-IID parameter β. We evaluate different β in $[0.01, 0.1, 0.6, 1000]$, $\beta = 0.01$ means the extremely non-IID scenario while $\beta = 1000$ means the nearly IID scenario. Fig. 6 illustrates the experimental results of the convergence curves on Fashion-MNIST with 5 Byzantine vehicles using LMP attacks. The results show that the convergence performance is guaranteed even with an extremely β value.

Effect of the local epochs. The local epochs mainly affect the convergence rate of the global model. Fig. 10 illustrates the convergence curves of the global model with different local epochs in $[1, 3, 5]$. The results demonstrate that the larger the local epochs are, the higher the final test accuracy can be. Increasing the local epochs improves the performance of DRL-PBFL, but it also increases the training time correspondingly. A suitable number of the local epochs is a trade-off between the training time cost and the model performance.

Effect of the batch sizes. Fig. 9 shows the convergence curves of the global model with different local batch sizes. The local batch sizes are set to $[32, 64, 128, 256, 512]$ respectively. We observe that the global model has the best performance while the batch size equals 128. Too high or too low local batch sizes result in different degrees of performance loss. Therefore, for the experiments in this paper, the local batch size of the federated learning is set to 128.

VI. Conclusions

In this paper, we have proposed a novel PBFL scheme in vehicular networks based on DDPG that prevents the Byzantine vehicles from degrading the model performance, while protecting the privacy of vehicles against the curious MEC node. To the best of our knowledge, it is the first work to achieve privacy-preserving model aggregation and effective evaluation of local model updates simultaneously in vehicular networks. By harnessing the distributed vehicular networking paradigm, the proposed scheme can utilize the distributed computing power of vehicles to collaborative train an FL model. In addition, an innovative secure aggregation mechanism based on Lagrange interpolation enables the weighted aggregation over the protected data, thus supporting the privacy-preserving linear computation of the sensitive data. Finally, we have demonstrated that the proposed scheme outperforms the state-of-the-art schemes in terms of security levels and global model accuracy on three ML benchmark tasks.

Acknowledgement

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3104500, NSFC (Nos. U22A2029, U20A20175, 62302387), Postdoctoral Innovative Talent Support Program of China under Grant BX20230282 and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.

[2] J. Posner, L. Tseng, M. Aloqaily, and Y. Jararweh, "Federated learning in vehicular networks: Opportunities and solutions," *IEEE Network*, vol. 35, no. 2, pp. 152–159, 2021. DOI: 10.1109/MNET.011.2000430.

[3] S. Dai, S. M. Iftekharul Alam, R. Balakrishnan, K. Lee, S. Banerjee, and N. Himayat, "Online federated learning based object detection across autonomous vehicles in a virtual world," in *2023 IEEE 20th Consumer Communications and Networking Conference (CCNC)*, 2023, pp. 919–920. DOI: 10.1109/CCNC51644.2023.10060782.

[4] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. NeurIPS*, 2017, pp. 118–128.

[5] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. ICML*, 2019, pp. 634–643.

[6] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE S&P*, 2019, pp. 739–753.

[7] L. Melis, C. Song, E. De Cristaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE S&P*, 2019, pp. 691–706.

[8] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. NeurIPS*, 2019, pp. 14 747–14 756.

[9] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" In *Proc. NeurIPS*, 2020, pp. 16 937–16 947.

[10] S. Yeom, M. Fredrikson, and S. Jha, "The unintended consequences overfitting: Training data inference attacks," *arXiv preprint arXiv:1709.01604*, 2017.

[11] Y. Wang, Y. Pan, M. Yan, Z. Su, and T. H. Luan, "A survey on ChatGPT: AI-generated contents, challenges, and solutions," *IEEE Open Journal of the Computer Society*, vol. 4, pp. 280–302, 2023.

[12] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. ICML*, 2018, pp. 5650–5659.

[13] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.

[14] R. Guerraoui, S. Rouault, *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *Proc. ICML*, 2018, pp. 3521–3530.

[15] X. Pan, M. Zhang, D. Wu, Q. Xiao, S. Ji, and M. Yang, "Justinian's gaavernor: Robust distributed learning with gradient aggregation agent," in *Proc. USENIX*, 2020, pp. 1641–1658.

[16] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 6893–6901.

[17] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *Proc. NDSS*, 2021, pp. 1–18.

[18] L. He, S. P. Karimireddy, and M. Jaggi, "Secure byzantine-robust machine learning," *arXiv preprint arXiv:2006.04747*, 2020.

[19] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.

[20] M. Rathee, C. Shen, S. Wagh, and R. A. Popa, "Elsa: Secure aggregation for federated learning with malicious actors," in *2023 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2023, pp. 1961–1979.

[21] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1639–1654, 2022.

[22] A. Yazdinejad, A. Dehghantanha, H. Karimipour, G. Srivastava, and R. M. Parizi, "A robust privacy-preserving federated learning model against model poisoning attacks," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 6693–6708, 2024.

[23] L. Zhao, J. Jiang, B. Feng, Q. Wang, C. Shen, and Q. Li, "Sear: Secure and efficient aggregation for byzantine-robust federated learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 1–14, 2021.

[24] H. Hashemi, Y. Wang, C. Guo, and M. Annavaram, "Byzantine-robust and privacy-preserving framework for fedml," *arXiv preprint arXiv:2105.02295*, 2021.

[25] K. Bonawitz, V. Ivanov, B. Kreuter, *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM CCS*, 2017, pp. 1175–1191.

[26] Y. Aono, T. Hayashi, L. Wang, S. Moriai, *et al.*, "Privacy-preserving deep learning: Revisited and enhanced," in *Proc. ATIS*, 2017, pp. 100–110.

[27] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2018.

[28] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[29] K. Wei, J. Li, M. Ding, *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[30] S. Truex, N. Baracaldo, A. Anwar, *et al.*, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM workshop on artificial intelligence and security*, 2019, pp. 1–11.

[31] Y. Wang, Z. Su, Y. Pan, T. H. Luan, R. Li, and S. Yu, "Social-aware clustered federated learning with customized privacy preservation," *IEEE/ACM Transactions on Networking*, pp. 1–13, 2024.

[32] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proc. USENIX*, 2020, pp. 1605–1622.

[33] M. Baruch, G. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *Proc. NeurIPS*, 2019, pp. 8635–8645.

[34] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *Proc. NDSS*, 2021, pp. 1–19.

[35] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[36] X. Li, H. Yao, J. Wang, X. Xu, C. Jiang, and L. Hanzo, "A near-optimal uav-aided radio coverage strategy for dense urban areas," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 9098–9109, 2019.

[37] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing uav," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[39] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," in *Proc. ICLR*, 2016, pp. 1–14.

[40] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. NeurIPS*, 1999, pp. 1057–1063.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–15.

[42] S. Bubeck *et al.*, "Convex optimization: Algorithms and complexity," *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015.

[43] I. Csiszár and J. Körner, *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press, 2011.

[44] A. Krizhevsky and G. Hinton, "Learning multiple layers features from tiny images," *Tech Report*, 2009.

[45] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. the IEEE*, 1998, pp. 2278–2324.

[46] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[47] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.