FLPhish: Reputation-based Phishing Byzantine Defense in Ensemble Federated Learning

Beibei Li[†], Peiran Wang[†], Hanyuan Huang[†], Shang Ma[†], and Yukun Jiang[†]

[†]School of Cyber Science and Engineering, Sichuan University, China
Email: libeibei@scu.edu.cn;{wangpeiran,huanghanyuan,mashang,jiangyukun}@stu.scu.edu.cn

***Abstract*—Increasing demand for personal privacy facilitates the growing interest in Federated Learning (FL). Nevertheless, the application of FL still suffers from the threat of Byzantine attacks which are incredibly challenging to withstand. In this paper, we propose a novel Byzantine-robust FL scheme called FLPhish. First, we develop a new FL architecture named Ensemble Federated Learning (called Ensemble FL) through making use of the unlabeled dataset in the FL server. Second, we design FLPhish, a Byzantine-robust scheme based on Ensemble FL to confront with Byzantine attacks. A phishing mechanism is crafted in our FL system to preserve security. Further, a Bayes's Theorem-based reputation mechanism is proposed to enhance the performance of our FLPhish. We evaluate FLPhish under different fractions of Byzantine clients and different distribution imbalance degrees q. Extensive experiments under different conditions demonstrate the high effectiveness of the proposed FLPhish in resisting Byzantine attacks in Ensemble FL.**

***Index Terms*—Federated Learning, Ensemble Learning, Reputation, Phishing**

I. Introduction

RECENT years have seen explosive social concerns about personal privacy, which motivates the advance of privacy-preserving technologies such as Federated Learning (called FL). FL is a privacy-preserving machine learning technology that enables central server to train a global model without accessing users' data. In FL, the central server only needs to train its global model using users' gradient updates instead of users' private data. Thus, FL can protect users' privacy. Google built the world's first product-level scalable mobile FL system based on TensorFlow[1]. Its FL system could be operated on thousands of mobile phones. Moreover, a team of WeBank developed an FL framework called FATE[2] for credit risk prediction.

The application of FL also leads the vulnerability to adversarial manipulations. Clients may be manipulated or poisoned by malicious attackers (called Byzantine attacks). The malicious clients can upload the wrong gradient updates to the central server or poison the dataset of the clients. Therefore, they cause the global model preserved by the central server invalid. By the results of the attacks, Byzantine attacks can be categorized into targeted attacks and untargeted attacks. In untargeted attacks [1]–[3], the disturbed global model produces incorrect predictions for the test dataset randomly. While in targeted attacks [4], the global model produces labels for the testing dataset in a definite pattern selected by attackers.

Some Byzantine-robust methods have been proposed by former researchers to address the malicious Byzantine clients in the application scenarios of FL [5]–[9]. The Byzantine-robust methods aim to formulate a global model with high accuracy in the presence of a bounded number of malicious clients. We can categorize the Byzantine-robust methods into two major types according to their different mechanisms. The core of the first one (called Byzantine-Detection) is to set up a Byzantine-robust aggregation rule which can distinguish the suspected clients from benign clients. Then server excludes the suspected clients' gradient updates before applying them for the aggregation. For instance, in DRACO [6], each node evaluates redundant gradients that are used by the parameter server to eliminate the effects of adversarial updates. While the key idea of the another category of Byzantine-robust methods (called Byzantine-Tolerance) is to ensure the aggregation process tolerant against Byzantine clients' poisoned updates without casting aside the Byzantine clients such as Median [8]. The FL server using Median sorts the values of each parameter and picks out the median value of each parameter as the value involving in the updating of global models. However, recent studies [2] implies that existing Byzantine-robust methods are still vulnerable to Byzantine attacks. Our contributions are three-fold:

- First, we design a new FL architecture, Ensemble Federated Learning (called Ensemble FL). Ensemble FL utilizes an unlabeled dataset to replace the gradient updates in traditional FL. This framework supports using different types of deep learning models in each client and makes FL more flexible.
- Second, we craft a robust Byzantine scheme called FLPhish based on our proposed 'phishing' method. FLPhish employs the labeled dataset to detect the potential Byzantine clients in the Ensemble FL system. It can preserve the security of Ensemble FL.
- Third, we propose a reputation mechanism to promote FLPhish's aggregation. Each client is given a reputation to measure the confidence value of it. A client with a low reputation is identified as a Byzantine client and discarded from the aggregation process.

[1]https://federated.withgoogle.com/
[2]https://github.com/FederatedAI/FATE

II. Models and Design Goals

In this section, we discuss the system model, threat model and identify our design goal.

A. System Model

We first discuss the design of traditional FL.

1) FL Server: FL server sends a global model to each client at each round. After receiving the gradient updates of all the clients, the FL server utilizes the gradient updates to aggregate a global update. The aggregation process is accomplished based on FedAvg. After the aggregation process, the FL server updates the global model via adding the global updates to the global model.

2) FL Client: Each client uses their local dataset to train the model sent from the FL server. Then it dispatches the gradient updates of the model back to the FL server.

B. Threat Model

However, the current system still suffers from multiple problems, especially Byzantine attacks. Malicious users can initiate untargeted Byzantine attacks towards global model via the label flipping attack in the current system model. Label-flipping requires adversaries to modify the labels of training data and ensure the features of data unchanging. Thus, malicious users can dispatch the false updates of the gradient to the central server. Therefore the false updates of the gradient can cause the central server to learn false distilled knowledge from clients. If the weight of the malicious clients reaches a threshold, the central server can be misguided to produce false predictions.

C. Design Goals

The key objective of the proposed FLPhish scheme is to provide a robust approach to accurately resist opportunistic untargeted attacks in our Ensemble FL system. Our design goals are as follows:

1) Inspired by the idea of ensemble learning, we build a new FL architecture called Ensemble FL. It can reduce the network transfer cost and provide more opportunities for us to counter Byzantine attacks in FL.

2) Our proposed Ensemble FL architecture lacks protection against Byzantine attacks. Since the clients in FL can not be fully trusted, we urgently require an efficient way to tackle malicious Byzantine clients. Thus, we present a phishing-based model to guard against Byzantine attacks in our proposed Ensemble FL system.

3) To accurately assess clients' behaviors, we further propose an effective Bayesian-based reputation scheme based on our phishing-based model to spot Byzantine attacks compromised by malicious users.

III. Our Framework

In this section, we introduce the phishing method and reputation mechanism proposed in detail.

A. Ensemble Federated Learning

Inspired by the idea of ensemble learning, we propose a new FL architecture, Ensemble FL.

Unlike existing FL architecture, which adopts gradient updates for global model updates, we apply an unlabeled dataset preserved by a central server and clients' predictions of it for global aggregation.

1) Client: Each client c_i (i indicates the number of the client) undertakes the task of collecting their local data, and labeling its local data. c_i collects data from the personal computer, smartphone, and smart cars, etc. The collected data will be labeled and preprocessed by c_i. After that, c_i adopts the preprocessed local dataset to train its local model. When receiving a public dataset, c_i utilizes its local model to make predictions for the public dataset and return the predictions to the central server s.

2) Central server: Central server s is responsible for making a public dataset and building a global model. The public dataset consists of a variety of data that is unlabeled. The dataset can be collected by s or produced by it (such as using GAN to generate data). After the construction of the public dataset, s sends the public dataset to the clients. Each client c_i sends its predictions about the public dataset back to s. After receiving all the predictions, s aggregates the predictions. Then s employs the aggregated results and the public dataset to train the global model.

This system model demonstrates a variety of advantages over traditional FL architecture:

- The selection of the global model and each client's local model is restricted to the same type of neural architecture in traditional FL. Nevertheless, in our system, deploying different types of neural architectures is allowed for us to apply the distilled knowledge (the predictions of the unlabeled data produced by the clients). Different selected features in different clients can be permitted as well.
- The overheads and latency of the communication process can be significantly reduced compared to the traditional FL architecture. Transferring data can be much faster than transferring gradient updates.

B. Phishing Mechanism

The proposed Ensemble FL still confronts with the threat of Byzantine attacks as shown in Fig. 1. Malicious clients can manipulate their local model via label flipping. They can mislabel the local dataset to build a 'poisoned' local model. When malicious clients receive unlabeled data from the central server, they manufacture false predictions (called poisoned knowledge) and send these false predictions to the central server. Subsequently, the central server aggregates the false predictions as the labels of the unlabeled dataset. Then central server trains the global model using these unlabeled datasets with the false aggregation predictions. Therefore, a flawed global model is manufactured. Inspired by the idea of ensemble learning, we consider utilizing the labeled data in the architecture of Ensemble FL to cope with Byzantine attacks. We called labeled data 'bait'.

Algorithm 1 Ensemble FL

```
Input: the ensemble of clients $C$ with local dataset $d_i$, $i = 1, 2, 3, ..., u$; a central server $s$ with unlabeled dataset $D$; number of training iterations $T$; unlabeled batch size $n$;
Output:
1:  $m_i \leftarrow$ each client $c_i$ train a local model using its own local dataset $d_i$;
2:  for $t$=1,2,3,...,$T$ do
3:      $s$ selects $D_t$ (containing $n$ samples) from $D$;
4:      for $i$=1,2,3,...,$u$ do
5:          $s$ sends $D_t$ to $c_i$;
6:          $c_i$ makes predictions $k_i^t$ of the $D_t$;
7:          $c_i$ sends $k_i^t$ to $s$;
8:      end for
9:      $Y_t$ = $KnowledgeEnsemble(k_1^t, k_2^t, k_3^t, ..., k_u^t)$;
10:     $M$ = $ModelUpdate(Y_t, D_t, M)$;
11: end for
12: return $M$.
```

Algorithm 2 KnowledgeEnsemble

```
Input: the ensemble of $k_{i\{i=1,2,3,...,u\}}^t$; size of each client's local dataset $e_{i\{i=1,2,3,...,u\}}$; the unlabeled dataset $D_t$ used in $t$th procedure; $\hat{y}^t$ is the ensembled prediction of dataset $D_t$; $\hat{y}_i^t$ denotes the prediction of the dataset $D_t$ made by $i$th client;
Output:
1: for $l = 1, 2, 3, ..., n$ ($l = 1, 2, 3, ..., n$, denotes the data point in the unlabeled dataset) do
2:     $\hat{k}^t \leftarrow \sum_{i=1}^{u} \frac{e_i}{\sum_{i=1}^{u} e_i} \hat{k}_i^t$;
3:     $\hat{y}^t \leftarrow argmax(\hat{k}^t)$;
4: end for
5: return $\hat{y}^t$.
```

1) Local Model Training: Each client c_i uses its own local dataset d_i to train a local model m_i as

$$m_i = Train(d_i). \tag{1}$$

2) Data Transfer: Central server s selects n samples of data D_t from unlabeled dataset D and m samples of data B_t from labeled dataset B. Then s sends D_t and B_t to each client c_i.

3) Predictions: Each client c_i predicts the labels of the unlabeled data D_t and the labeled data B_t:

$$k_i^t = Predict(D_t, B_t, m_i) \tag{2}$$

(c_i can not distinguish between D_t and B_t). via the local model trained by itself in Step 1 and sends its prediction back to the central server as the distilled knowledge k_i^t:

$$k_i^t = \begin{vmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,g-1} & p_{1,g} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,g-1} & p_{n,g} \end{vmatrix}. \tag{3}$$

Unlike benign clients, malicious clients return the false prediction as the distilled knowledge to central server s.

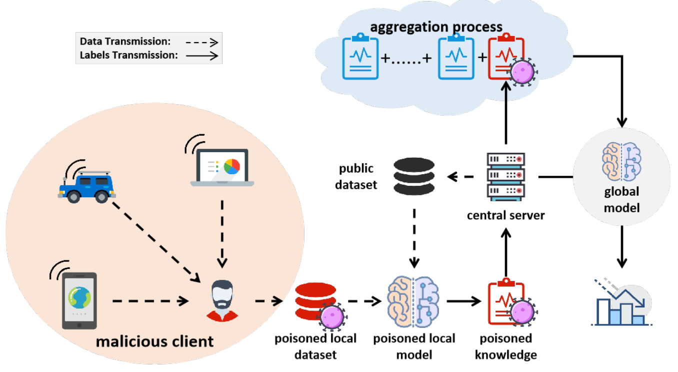


Fig. 1. Byzantine attacks in Ensemble FL.

4) Byzantine Identification: Accepting the distilled knowledge from each client c_i, s extracts the predictions of the *'bait'* from k_i^t, and calculates the accuracy of the predictions via the true label of the *'bait'*:

$$a_i^t = AccuracyCal(k_i^t, B_t). \tag{4}$$

Then s identifies those clients who hold a low value of accuracy and distinguishes as malicious clients.

5) Global Model Update: After identifying the malicious clients within all clients, s aggregate the knowledge $\hat{k}^t$ from all clients as

$$\hat{k}^t = \sum_{i=1}^{u} \frac{e_i}{\sum_{i=1}^{u} e_i} \hat{k}_i^t. \tag{5}$$

Then the server s use the aggregated knowledge $\hat{k}^t$ to get the labels

$$\hat{y}^t \leftarrow argmax(\hat{k}^t). \tag{6}$$

C. Reputation Mechanism: Bayesian Inference

Server s maintains a reputation list which records the reputation of all the clients C in the model. Let X_i be the reputation of the c_i client which represents s's belief that how likely client c_i is a Byzantine client. The computation of X_i is based on the accuracy a_i^t of client c_i from the 1st round to the tth round. Every time a new update of client c_i comes to server s, s uses the accuracy a_i^t to update the X_i.

Initially, the reputation is neutral. Each client c_i is considered a benign client by server s with a probability of 50%. When a new update k_i^t comes to the server s, the reputation is updated by the s. When the reputation is lower than the threshold r, s considers the client c_i as a Byzantine client officially and discards the update coming from the client c_i. The updates of the client c_i are reconsidered in the aggregation when the X_i exceeds the threshold r.

We employ Bayesian inference to construct our reputation mechanism. For each data prediction made by client c_i, there are two situations: wrong predictions or correct predictions. Thus, the use of binomial parameter distributions becomes a natural choice for our reputation mechanism. Let Y_i be the event that the number of wrong predictions and correct

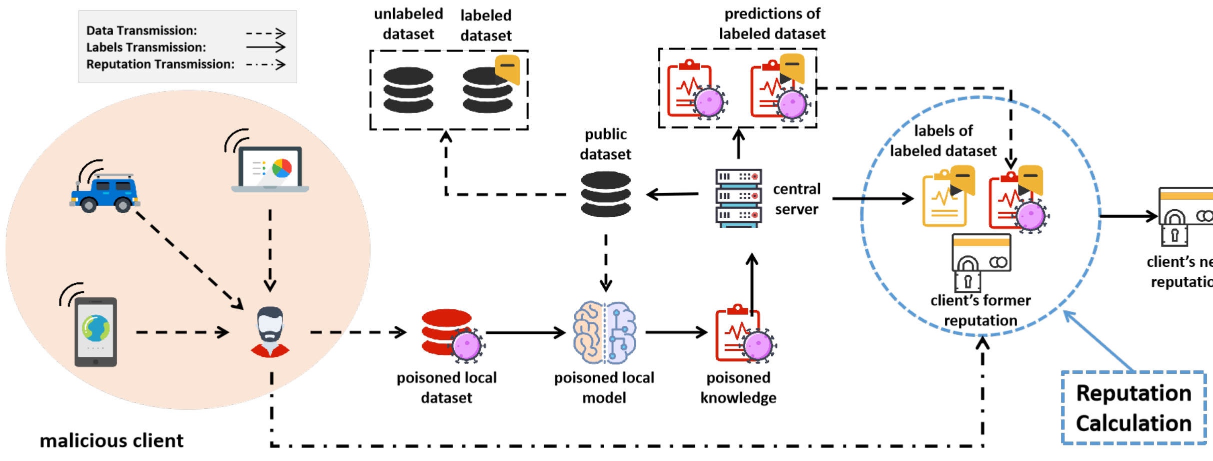


Fig. 2. Phishing Method & Reputation Mechanism.

predictions made by client c_i is α_i and β_i. Given $X_i = \gamma_i$, then the conditional probability is

$$Pr(Y_i|X_i=\gamma_i) = \left(\frac{\alpha_i+\beta_i}{\alpha_i}\right)\gamma_i{}^{\alpha_i}\left(1-\gamma_i\right)^{\beta_i}. \quad (7)$$

α_i and β_i is the available evidence for the estimation of the X. γ_i is unknown. Equation 7 indicates the likelihood function for X. According to Bayes' theory, we can compute the posterior probability as

$$Pr(X_i=\gamma_i|Y_i) = \frac{Pr\left(Y_i|X_i=\gamma_i\right)Pr(X_i=\gamma_i)}{\int_0^1 Pr\left(Y_i|X_i=x\right)Pr(X_i=x)dx}. \quad (8)$$

As the posterior probability function is given by our analysis, the final value for the expectation value of reputation X can be computed as

$$E(X) = \int_0^1 \frac{Pr\left(Y_i|X_i=\gamma_i\right)Pr(X_i=\gamma_i)}{\int_0^1 Pr\left(Y_i|X_i=x\right)Pr(X_i=x)dx}\gamma_i d\gamma_i. \quad (9)$$

Furthermore, we decide to use the binomial parameter beta distribution to describe the distribution of X. Assume X is a random variable of the beta distribution with the parameters (α, β). Therefore the density function is

$$f(x,\alpha,\beta) = \frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}. \quad (10)$$

B function is the beta function. Therefore we can compute the expectation of X as

$$E(x) = \frac{\alpha}{\alpha+\beta}. \quad (11)$$

Therefore server s can easily compute the reputation of the client c_i as $E(x_i) = \frac{\alpha_i}{\alpha_i+\beta_i}$. We can set the initial value of the reputation to 50% by set the value of α and β to 1. This indicates that the probability of being benign client and malicious client are equal for client c_i. When a new round of FL is accomplished, $\alpha_i{}'$ and $\beta_i{}'$ are provided as the new evidence for the computation of the reputation. The parameters are updated as $(\alpha_i+\alpha_i{}', \beta_i+\beta_i{}')$. Then the reputation is given by

$$E(x_i)' = \frac{\alpha_i+\alpha_i{}'}{\alpha_i+\alpha_i{}'+\beta_i+\beta_i{}'}. \quad (12)$$

Then we identify the client c_i whose reputation is lower than the threshold τ as Byzantine clients, and given them a aggregation weight as

$$\omega_i = \begin{cases}1 & \text{if } x \geq \tau\\ 0 & \text{if } x < \tau\end{cases}. \quad (13)$$

The aggregated knowledge will be given by

$$\hat{k}^t = \sum_{i=1}^{u}\frac{e_i}{\sum_{i=1}^{u}e_i}\hat{k}_i^t \times \omega_i \quad (14)$$

$$\hat{y}^t \leftarrow argmax(\hat{k}^t). \quad (15)$$

IV. Performance Evaluation

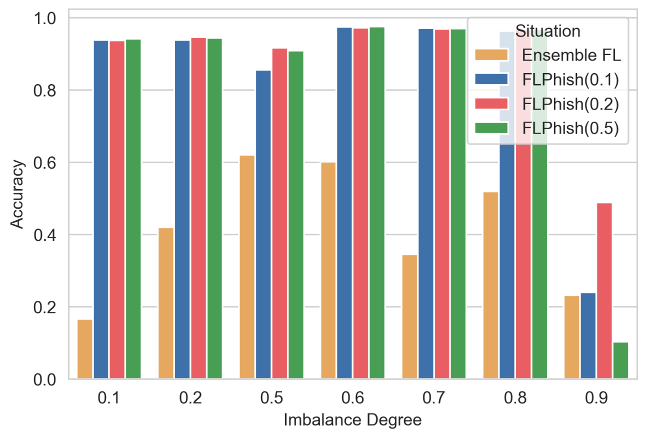
In this section, we experimentally evaluate our FLPhish against the untargeted attacks under different conditions.

A. Experiment Setup

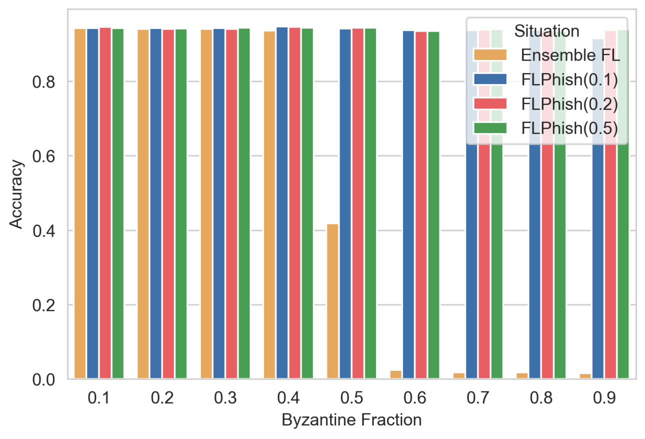
1) The fractions of Byzantine clients: We evaluate our FLPhish under the circumstances of different fractions of Byzantine clients: 0.0 (no Byzantine clients), 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9.

2) The imbalance degree q of the data: According to the previous research [2], we distribute the data in a dataset among all the clients. Giving M classes of data in a dataset, we split the clients into M groups. A client c in group m is provided with data where data m accounts for over q percent. Within the same group, data are uniformly distributed among all the clients. The parameter q controls the distribution inference of clients' local training data. If $q = \frac{1}{M}$, the clients' local training data are independent and identically distributed. We evaluate our FLPhish on three different q: 0.1 (IID), 0.2, 0.5, 0.6, 0.7, 0.8 and 0.9 (extremely non-IID).

3) The number of the clients: The number of clients is set to 50 in our experiment.

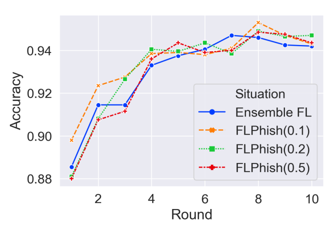


(a) Experiments on different imbalances degrees

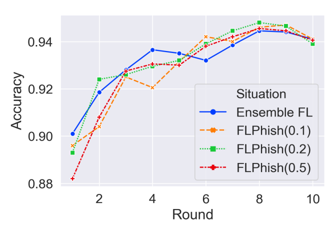


(b) Experiments on different fractions of byzatnine clients

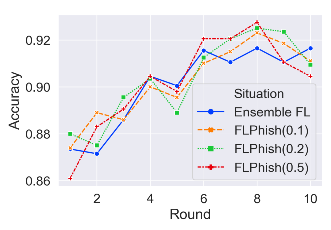
Fig. 3. Performance comparison on different fractions of byzatnine clients and different imbalances degrees.



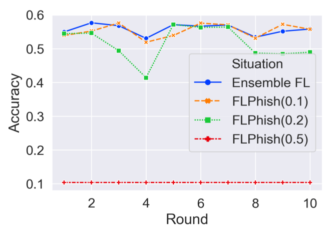
(a) q=0.1 & Byzantine fractions=0.2



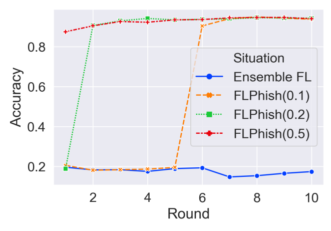
(b) q=0.2 & Byzantine fractions=0.2



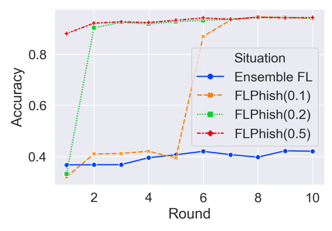
(c) q=0.5 & Byzantine fractions=0.2



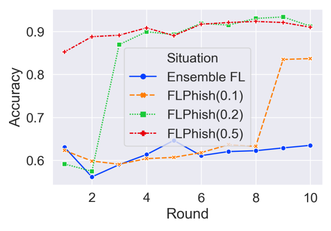
(d) q=0.9 & Byzantine fractions=0.2



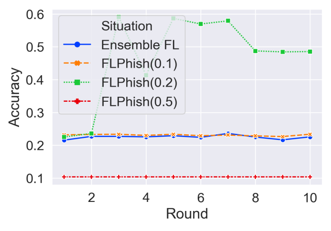
(e) q=0.1 & Byzantine fractions=0.5



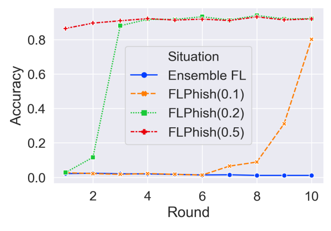
(f) q=0.2 & Byzantine fractions=0.5



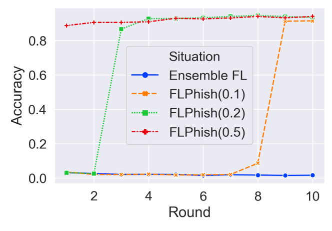
(g) q=0.5 & Byzantine fractions=0.5



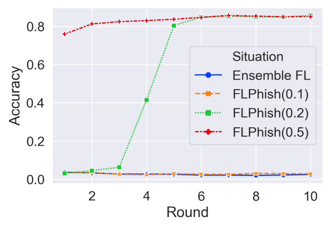
(h) q=0.9 & Byzantine fractions=0.5



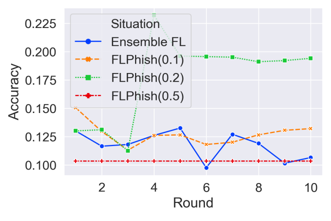
(i) q=0.1 & Byzantine fractions=0.9



(j) q=0.2 & Byzantine fractions=0.9



(k) q=0.5 & Byzantine fractions=0.9



(l) q=0.9 & Byzantine fractions=0.9

Fig. 4. Performance comparison on MNIST untargeted attack.

4) The local CNN model used by clients: We use ResNet to perform deep learning tasks in our local client.

5) The dataset: We take MNIST as our experiment dataset. Each client has 1000 samples. Among the samples, 800 are used as training datasets, while another 200 are treated as test datasets.

B. Experiment Results

1) Performance Comparison on Different Distributions: We evaluate our FLPhish under the condition where Byzantine client portion is a fixed value 0.5 and distribution imbalance value q is different across the Experiments. The experiment results are shown in Fig. 3. From Fig. 3, we can observe that the FLPhish outperforms baseline until the

TABLE I
PERFORMANCE COMPARISON ON DIFFERENT FRACTIONS OF BYZANTINE AND FIXED IMBALANCE DEGREE $q = 0.2$.

Byzantine clients fraction	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Baseline	0.9430	0.9405	0.9410	0.9365	0.4185	0.0240	0.0180	0.0180	0.0150
FLPhish(threshold=0.1)	0.9425	**0.9425**	0.9430	**0.9470**	0.9415	**0.9380**	0.9380	**0.9385**	0.9160
FLPhish(threshold=0.2)	**0.9465**	0.9405	0.9410	0.9460	0.9435	0.9355	0.9385	0.9380	0.9370
FLPhish(threshold=0.5)	0.9430	0.9415	**0.9440**	0.9440	**0.9440**	0.9355	**0.9395**	0.9355	**0.9395**

TABLE II
PERFORMANCE COMPARISON ON DIFFERENT q AND FIXED FRACTIONS OF BYZANTINE.

Imbalance Degree	q=0.1	q=0.2	q=0.5	q=0.6	q=0.7	q=0.8	q=0.9
Baseline	0.1660	0.4195	0.6205	0.6005	0.3450	0.5185	0.2315
FLPhish(threshold=0.1)	0.9375	0.9380	0.8555	0.9735	**0.9700**	0.9625	0.2395
FLPhish(threshold=0.2)	0.9365	**0.9460**	**0.9165**	0.9715	0.9685	**0.9670**	**0.4880**
FLPhish(threshold=0.5)	**0.9405**	0.9430	0.9080	**0.9745**	0.9690	**0.9670**	0.1035

imbalance degree q reaches 0.9. The imbalance degree q of 0.9 means that the distribution of data becomes extremely non-IID. The performance of FLPhish with a threshold of 0.5 rapidly falls in this case. When facing the distribution of imbalance degree $q = 0.9$, each client performs badly, bringing a decline to its reputation. Figure. 4demonstrates that the rise of imbalance degree will bring a explosive decline to the global model's accuracy. The accuracy of the global model under the FLPhish of threshold 0.5 stays 0.1 in the whole learning process. It means that the FLPhish of threshold 0.5 identifies all the clients as Byzantine clients, making the aggregation process invalid. We can see that FLPhish of threshold 0.2 outperforms others. When the distribution becomes non-IID, the threshold of FLPhish should be set to a lower value to avoid a high false-negative rate. In the experiment of MNIST, it should be set to 0.2.

2) Performance Comparison on Different Fractions of Byzantine Clients: Different fractions of Byzantine clients are taken into account by us as well. Figure. 3 shows that the accuracy for Ensemble FL without any defense mechanism begins to fall rapidly when the Byzantine portion reaches nearly 50%. In comparison, the performance of FLPhish under different thresholds keeps good performance when the portion increases. It means that FLPhish detects Byzantine clients effectively and discards them from the aggregation process accurately. With no Byzantine clients involving in aggregation processes, the global model can be trained successfully. Besides, we can infer from the results that FLPhish with threshold of 0.1 performs worse than the other two. This is because the threshold of 0.1 is a excessively low value to detect Byzantine clients effectively. Specifically, there are some Byzantine clients' reputation can go over 0.1. It means Byzantine clients can bypass FLPhish's detection if the threshold is set to a overly low value.

V. CONCLUSION

In this paper, we develop an FL architecture called Ensemble Federated Learning. Ensemble FL enable us to take advantage of unlabeled dataset to transfer knowledge between the FL server and the FL clients. Specifically, we propose a FLPhish mechanism to enable Ensemble FL robust against Byzantine attacks using labeled dataset as 'bait' to detect malicious Byzantine clients in Ensemble FL. Moreover, we present a reputation mechanism to measure the confidence level of the clients. Furthermore, we evaluate our proposed FLPhish in different situations. The experiment results demonstrate that FLPhish shows outstanding performance to confront with Byzantine attacks. Even encountering conditions with extremely non-IID distribution and high fractions of Byzantine clients, FLPhish still shows much better performance than our Ensemble FL.

REFERENCES

[1] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Model poisoning attacks in federated learning," in *Proc. Workshop Secur. Mach. Learn.(SecML) 32nd Conf. Neural Inf. Process. Syst.(NeurIPS)*, Dec 2018.

[2] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020.

[3] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ser. ICML '12, July 2012.

[4] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, 26–28 Aug 2020.

[5] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, Dec 2017.

[6] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "DRACO: Byzantine-resilient distributed training via redundant gradients," in *Proceedings of the 35th International Conference on Machine Learning*, 10–15 Jul 2018.

[7] 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*, 09–15 Jun 2019.

[8] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*, 10–15 Jul 2018.

[9] X. Cao, M. Fang, J. Liu, and N. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *2021 Network and Distributed System Security Symposium (NDSS)*, Feb 2021.