Defending Byzantine Attacks in Ensemble Federated Learning: A Reputation-based Phishing Approach

Beibei Li, *Member, IEEE,* Peiran Wang, *Student Member, IEEE,* Lei Zhang,Yuan Zhang, *Member, IEEE,* and Rongxing Lu, *Fellow, IEEE*

***Abstract*—Federated Learning, a new distributed machine learning scheme, is gaining popularity as the demand for personal privacy grows. Nonetheless, the implementation of FL is still vulnerable to Byzantine attacks, which can bring significant hazards to the global model aggregation process. Byzantine attacks are extremely difficult to defend against in the current FL scheme. In this study, we present a novel reputation-based phishing scheme (called FLPhish) in defending against Byzantine attacks in Ensemble FL. First, we design a new Ensemble FL architecture, to enable flexible implementation of various types of deep learning models in different clients. Second, we craft a phishing mechanism for the Ensemble FL architecture to identify Byzantine attacks. Furthermore, a reputation mechanism based on Bayes's theorem is presented to assess each client's level of confidence. Last, we propose two aggregation techniques in FLPhish: FLPhish-threshold and FLPhish-weight. FLPhish is tested with varying proportions of Byzantine clients and varying degrees of distribution imbalance. Extensive experiments under various situations demonstrates the proposed FLPhish's great efficacy in resisting Byzantine attacks in Ensemble FL.**

***Index Terms*—Federated learning, ensemble learning, bayesian inference reputation, phishing.**

I. Introduction

MANY elements of our daily lives and society have benefited from deep learning tasks in natural language processing, computer vision, and anomaly detection. To learn complex rules, such activities necessitate a large dataset. In most cases, these huge datasets are gathered from users, such as the app's users. Developers get information from users and utilize it to create the dataset. Nonetheless, in recent years, there has been an explosion in social concerns about personal privacy protection, making it difficult to get data directly from consumers any more. Under these circumstances, each individual's data is referred to as an 'Isolated Data Island'. The existence of each 'Isolated Data Island' drives the development of privacy-preserving solutions like Federated Learning [1]. FL is a privacy-preserving machine learning solution that allows a central server to train a global model without gaining access to individuals' personal privacy data. Instead of training global model on users' private data, the central server in FL just needs to train its global model using their gradient updates. FL protects an individual's privacy while simultaneously reducing the load on the central server by leveraging the capabilities of distributed computation and storage. FL is soon applied by many companies. 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

TABLE I
SUMMARY OF NOTATIONS

Term	Description
s	central server in FL
$\mathbf{M}$	global model preserved by s
c_i	the ith client in FL, $i = 1, 2, 3, ..., u$
d_i	the local dataset preserved by the ith client
C	the ensemble of all the clients
u	the number of clients
$\mathbf{m_i}$	local models trained by the ith client
D_t	the unlabeled dataset chosen by s in each procedure
D	the unlabeled dataset preserved by s
n	the number of samples in D_t
B_t	the labeled dataset (*'bait'*) chosen by s in each procedure
B	the labeled dataset preserved by s
m	the number of samples in B_t
$\mathbf{k_i^t}$	the predictions (*'knowledge'*) made by the ith client in the tth procedure
a_i^t	the accuracy of predictions of B_t made by c_i in tth procedure
q_i	the label of c_i to judge it is a malicious client or not
r_q	the threhold of malicious clients
x_l^t	the lth data point in D_t
$\hat{\mathbf{y}}_\mathbf{l}^\mathbf{t}$	the ensembled prediction of data point x_l^t
$\hat{\mathbf{y}}_\mathbf{l}^\mathbf{i}$	the prediction of lth data point made by ith client
$\mathbf{K_t}$	the aggregated labels (predictions) of the tth round's unlabeled dataset

This paper is the extended vision of the paper, 'FLPhish: Reputation-Based Phishing Byzantine Defense in Ensemble Federated Learning', which is published in IEEE ISCC 2021, and rated as the best paper in this conference.

B. Li, P. Wang, and L. Zhang are with the School of Cyber Science and Engineering, Sichuan Universsity, Chengdu, Sichuan, China 610065 China. Email: libeibei@scu.edu.cn; wangpeiran@stu.scu.edu.cn; zhanglei2018@scu.edu.cn.

R. Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada E3B 5A3. Email: rlu1@unb.ca.

Y. Zhang is with the Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China 610054. Email: zy_loye@126.com.

[1]https://federated.withgoogle.com/

scheme called FATE[2] for credit risk prediction. And some former researchers have also applied FL in some industrial cyber–physical Systems [2], [3].

In FL, thousands of clients from different sources participate in the training process. Such a large-scale distributed system also brings security issues to FL. Former researchers have already studied the privacy problems of FL and offered their schemes to enhance privacy protection in FL [4] [5]. Furthermore, FL clients may be manipulated or poisoned by malicious attackers. Researchers call such attacks Byzantine attacks referring to the same types of attack in wireless communication network [6]–[10]. By poisoning the clients' datasets or directly changing the gradient updates, the malicious clients send the incorrect gradient updates to the central server. The incorrect gradient updates cause the central server's global updates to learn incorrect knowledge from the clients. As a result, they render the central server's global model obsolete. Furthermore, Byzantine attacks are divided into two types based on the consequences of the attacks: targeted attacks and untargeted attacks. The disturbed global model randomly delivers inaccurate predictions for the test dataset in untargeted attacks [11]–[13]. In targeted attacks, the global model generates labels for the testing dataset in a predetermined pattern chosen by the attackers [14]–[18].

Former researchers have offered certain Byzantine-robust techniques to deal with malevolent Byzantine clients in FL application settings [19]–[28]. In the presence of a bounded number of malicious clients, Byzantine-robust approaches try to develop a global model with high accuracy. According to their different mechanisms, we divide Byzantine-robust approaches into two major kinds. The first (named Byzantine-Detection) is based on the creation of a Byzantine-robust aggregation rule that distinguishes questionable customers from benign clients. The server then eliminates the suspected clients' gradient updates from the aggregate process. For instance, in DRACO, each node analyzes duplicate gradients that the parameter server uses to mitigate the effects of adversarial updates [20]. Another group of Byzantine-robust approaches (dubbed Byzantine-Tolerance) aims to ensure that the aggregation process is tolerant of Byzantine clients' poisoned updates without excluding Byzantine clients like Median [22]. The FL server sorts the values of each parameter based on Median and selects the median value of each parameter as the value to be used in global model updates. Recent research, however, suggest that current approaches are still vulnerable to Byzantine attacks [12]. We present a novel reputation-based phishing scheme (called FLPhish) in defending against Byzantine attacks in Ensemble FL. Our contributions are four-folds:

- 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 architecture supports different types of deep learning models in each client and makes FL more flexible.
- Second, we craft a 'phishing' method based on Ensemble FL to detect Byzantine attacks. The 'phishing' method employs the labeled dataset to detect the potential Byzantine clients in the Ensemble FL system. It preserves the security of Ensemble FL.
- Third, we present a bayesian inference-based reputation mechanism to promote FLPhish's aggregation. The reputation mechanism gives each client a reputation to measure its confidence value and identifies the clients with low reputation values as Byzantine clients. The reputation mechanism helps FLPhish identify the Byzantine clients more accurately.
- Last, we propose two aggregation algorithms FLPhish-threshold and FLPhish-weight to aggregate the predictions of the FL clients, based on reputation mechanism. FLPhish using FLPhish-threshold identifies the clients with low reputation values as Byzantine clients, while FLPhish with FLPhish-weight utilizes the reputation value of each client as its aggregation weight to participate aggregation process. These two algorithms contribute to the robustness of FLPhish.

[2]https://github.com/FederatedAI/FATE

II. Related Work

A. Byzantine Defense Methods in Federated Learning

Recent years have witnessed the increasing interest in the research of Byzantine-robust schemes in the context of FL. Most of the current Byzantine-robust FL methods tend to make a more robust aggregation rule which aims to tolerate the presence of Byzantine clients. For example, in 2017, Chen *et al*. Krum developed an approach [19]. Krum selects one client's update as a global model based on a square-distance score in each iteration. In the same year, Blanchard *et al*. proposed two Byzantine-tolerant FL aggregation algorithms Trimmed mean and Median [22]. Trimmed Mean considers each parameter of the model update individually. Trimmed Mean sorts the parameter of the model updates collected. Median sorts the values of each parameter of all local model updates as well. And it considers the median value of each parameter as the value of the parameter in the global model update. In 2018, Chen *et al.* designed an approach Draco to evaluate redundant gradients that are used by the parameter server to eliminate the effects of adversarial updates. In 2019, Zeno uses a ranking-based preference mechanism [21]. The server computes a score for each client by using the stochastic zero-order oracle. Then Zeno presents a ranking list of clients based on the estimated descent of the loss function and the magnitudes. At last, Zeno computes the global model update by aggregating the clients with the highest scores. In 2020, SLSGD developed by Xie *et al.* also uses trimmed mean as the robust aggregation rules for Byzantine-robust FL [29]. In the same year, Cao *et al.* proposed a Byzantine-tolerant scheme: FLTrust to introduce the use of trust [23]. In each iteration, the server calculates a trust score for each client at first, and lowers the trust score if the client's local model update's direction deviates more from the direction

of the global model update. The client with a trust score lower than the threshold is considered a malicious client. In 2021, a privacy-enhanced FL (PEFL) framework is presented by Liu *et al.* [30]. PEFL takes advantages of homomorphic encryption to protect the privacy of the clients. Furthermore, a channel using the effective gradient data extraction is provided for the server to punish poisoners.

B. Reputation Mechanism in Information Security

The reputation mechanism is valued as a way to measure an entity's performance in a long term, such as in an online social network [31], and in a smart grid system [32], [33]. In 2012, Das *et al.* first presented a dynamic trust computation model called SecuredTrust. This framework is used to distribute the workload and deal with the altering behavior of malicious clients [34]. In 2015, Zhu *et al.* proposed an authenticated trust and reputation calculation and management system in wireless sensor network and cloud computing to calculate and manage trust and reputation of the service of CSP and SNP [35]. In 2018, Lei *et al.* presented a Reputation-based Byzantine Fault Tolerance algorithm that incorporates a reputation model to evaluate the performance of each node in the blockchain system [36]. The nodes get lower discourse rights and reputation in the voting process if any malicious behavior is detected by the system. Furthermore, they presented a reputation-based primary change scheme. The node with a higher reputation would get greater opportunities to generate new valid blocks, which reduces the security risk of the system. In 2020, Chouikhi *et al.* built a model for reputation computing and a credibility model to enhance network efficiency [37]. They used the reputation score or value to measure the behavior of a vehicle towards other vehicles and network services. And the credibility of vehicles is used to determine the accuracy of a reputation score offered by a vehicle. In the same year, Wen *et al.* proposed a Dirichlet reputation-based scheme and adopt the reputation score to select a trustworthy Helper as a friendly jammer in a wireless cooperative system (WCS) [38]. Furthermore, they developed an artificial noise detection method with multiple thresholds. They provided ratings with multiple graded levels. In the Dirichlet reputation-based scheme, the graded ratings were directly expressed and reflected in the derived reputation scores. In 2021, Liang *et al.* presented a markov-based reputation scheme in an intrusion detection system. The Hidden Generalized Mixture Transition Distribution (HgMTD) model, namely RS-HgMTD, is developed to assist each vehicle in the VANET to measure the creditworthiness of its neighbor vehicles [39].

III. Models and Design Goals

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

A. System Model

We first discuss the design of traditional FL consisting of two entities, FL server, and FL client.

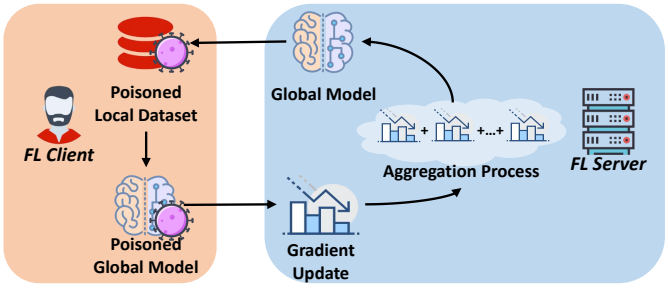


Fig. 1. System Model&Threat Model.

1) FL Server: FL server s 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 FL client c_i (c_i indicates the ith client in FL) preserves a local dataset d_i collected by themselves. FL client c_i uses its local dataset d_i to train the model sent from the FL server. Then it dispatches the gradient updates of the model back to the FL server. It repeats the above actions in the whole process of FL until the FL server s stops sending the new model.

B. Threat Model

However, the current system still suffers from multiple problems, especially Byzantine attacks. Malicious Byzantine client b initiates untargeted Byzantine attacks towards the global model via the label flipping attack in the current system model. Label-flipping requires b to modify the labels of training data and ensure the features of data unchanging [40]. Byzantine client b's local model is trained with false labels, thus constructing a 'poisoned' model with low accuracy. Then Byzantine client b dispatches the false updates of the gradient to the central server. Therefore the false updates of the gradient cause the central server to learn false distilled knowledge from clients. The server s's aggregation process is played on FedAvg which takes each client c_i's dataset d_i's size as the aggregation weight for c_i. This means that a client c_i with a larger size of d_i gets a larger aggregation weight. Meanwhile FedAvg takes the size of d_i declared by c_i as d_i's real size which means c_i can declare a fake size number larger than d_i's real size number. If the weight of the malicious clients reaches a threshold, the central server is 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 given as follows:

1) Inspired by the idea of ensemble learning, we build a new FL architecture called Ensemble FL. It reduces the

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:

$\mathbf{m_i} \leftarrow$ each client c_i train a local model using its own local dataset d_i;
for *t=1,2,3,..,,T* **do**
  s selects D_t (containing n samples) from D;
  for *i=1,2,3,...,u* **do**
    $s \xrightarrow{D_t} c_i$;
    c_i makes predictions $\mathbf{k_i^t}$ of the D_t;
    $c_i \xrightarrow{\mathbf{k_i^t}} s$;
  end for
  $Y_t = KnowledgeEnsemble(\mathbf{k_1^t}, \mathbf{k_2^t}, \mathbf{k_3^t}, ..., \mathbf{k_u^t})$;
  $\mathbf{M} = ModelUpdate(Y_t, D_t, \mathbf{M})$;
end for
return $\mathbf{M}$.

Algorithm 2 KnowledgeEnsemble

Input: the ensemble of $\mathbf{k_{i\{i=1,2,3,...,u\}}^t}$; size of each client's local dataset $e_{i\{i=1,2,3,...,u\}}$; the unlabeled dataset D_t used in tth procedure; $\hat{\mathbf{y}}^\mathbf{t}$ is the ensembled prediction of dataset D_t; $\hat{\mathbf{y}}_\mathbf{l}^\mathbf{t}$ denotes the prediction of the dataset D_t made by ith client;
Output:

for $l = 1, 2, 3, ..., n$ ($l = 1, 2, 3, ..., n$, denotes the data point in the unlabeled dataset) **do**
  $\hat{\mathbf{k}}^\mathbf{t} \leftarrow \sum_{i=1}^{u} \frac{e_i}{\sum_{i=1}^{u} e_i} \hat{\mathbf{k}}_i^\mathbf{t}$;
  $\hat{\mathbf{y}}^\mathbf{t} \leftarrow argmax(\hat{\mathbf{k}}^\mathbf{t})$;
end for
return $\hat{\mathbf{y}}^\mathbf{t}$.

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.

4) Furthermore, we propose two aggregation algorithms in FLPhish, FLPhish-threshold and FLPhish-weight to aggregate the predictions of the FL clients, which can further enhance FLPhish's defending ability against Byzantine attacks.

IV. Our Proposed FLPhish Scheme

In this section, we elaborate on the proposed FLPhish scheme including the Ensemble Federated Learning, the phishing mechanism, the reputation mechanism, and the aggregation algorithms.

A. Designed 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 is 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 is 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 architectures:

- 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 by applying the distilled knowledge (the predictions of the unlabeled data produced by the clients). Different selected features in different clients are permitted as well.
- The overheads and latency of the communication process are significantly reduced compared to the traditional FL architecture. Transferring data is much faster than transferring gradient updates.

B. Phishing Mechanism-based Detection

The proposed Ensemble FL still confronts the threat of Byzantine attacks. Malicious clients can manipulate their local model via label flipping. They 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

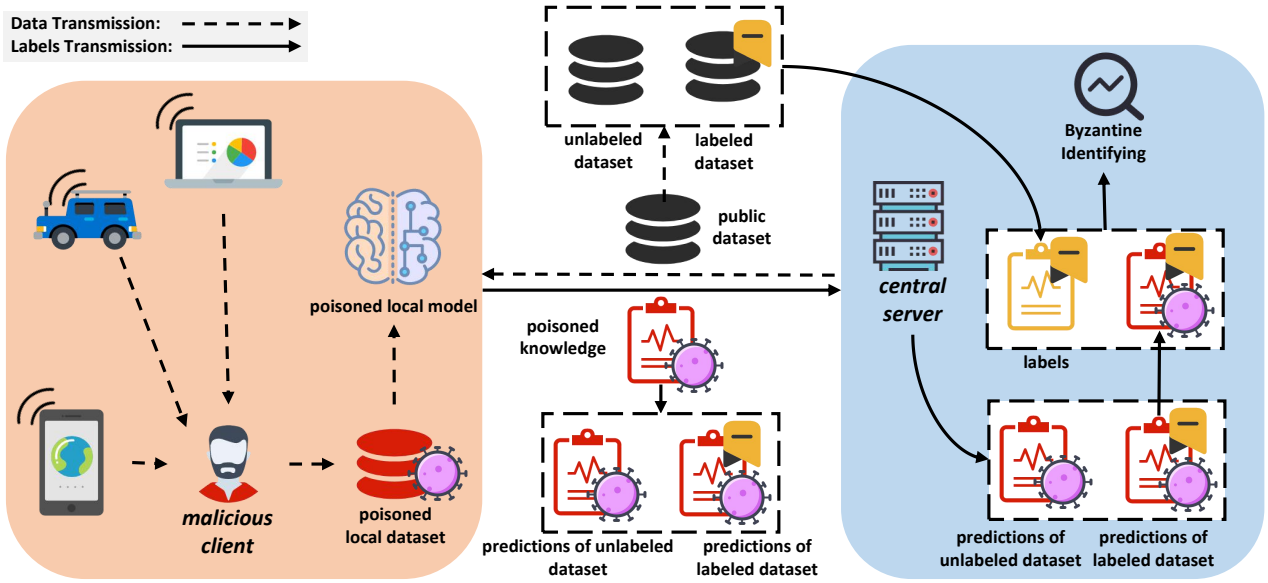


Fig. 2. Our Proposed FLPhish Scheme.

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'.

1) Local Model Training: The ensemble of clients $C = \{c_1, c_2, ..., c_{n-1}, c_n\}$. Each client c_i possesses a local dataset d_i. At the beginning of the Ensemble FL, c_i utilizes its local dataset d_i to train a local model $\mathbf{m_i}$ as

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

2) Dataset Transferring: 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 randomly. Then s sends D_t and B_t to each client c_i as

$$s \overset{D_t, B_t}{\rightarrow} c_i. \tag{2}$$

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

$$\mathbf{k_i^t} = Predict(D_t, B_t, \mathbf{m_i}) \tag{3}$$

(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 $\mathbf{k_i^t}$:

$$\mathbf{k_i^t} = \begin{vmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,g-1} & p_{1,g} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,g-1} & p_{n,g} \end{vmatrix}. \tag{4}$$

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

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

$$a_i^t = AccuracyCal(\mathbf{k_i^t}, B_t). \tag{5}$$

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

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

$$\hat{\mathbf{k}}^\mathbf{t} = \sum_{i=1}^{u} \frac{e_i}{\sum_{i=1}^{u} e_i} \hat{\mathbf{k}}_\mathbf{i}^\mathbf{t}. \tag{6}$$

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

$$\hat{\mathbf{y}}^\mathbf{t} \leftarrow argmax(\hat{\mathbf{k}}^\mathbf{t}). \tag{7}$$

C. Bayesian Inference-based Reputation Mechanism

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 $\mathbf{k_i^t}$ 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.

Algorithm 3 Phishing Mechanism

Input: the ensemble of clients C with local dataset d_i, $i = 1, 2, 3, ..., u$; a central server s with unlabeled dataset D and labeled dataset B; number of training iterations T; unlabeled batch size n;labeled batch size m.

Output: output result

$\mathbf{m_i} \leftarrow$ each client c_i train a local model using its own local dataset d_i.
for *t=1,2,3,..,T* **do**
  s selects D_t (containing n samples) from D and B_t (containing m samples) from B.
  for *i=1,2,3,...,u* **do**
    s sends D_t and B_t to c_i.
    c_i makes predictions $\mathbf{k_i^t}$ of the D_t and B_t.
    c_i sends $\mathbf{k_i^t}$ to s.
    s calculates the accuracy a_i^t of the predictions of B_t made by c_i in tth procedure.
    if $a_i^t > r_q$ **then**
      $q_i = 1$.
      for *j=i,i+1,i+2,...,u-1* **do**
        $d_j \leftarrow d_{j-1}$.
        $c_j \leftarrow c_{j+1}$.
        $\mathbf{k_j^t} \leftarrow \mathbf{k_{j+1}^t}$.
        $q_j \leftarrow q_{j+1}$.
        $a_j^t \leftarrow a_{j+1}^t$.
      end for
    end if
  end for
  $\mathbf{K_t} = KnowledgeEnsemble(\mathbf{k_1^t}, \mathbf{k_2^t}, \mathbf{k_3^t}, ..., \mathbf{k_u^t})$.
  $\mathbf{M} = ModelUpdate(\mathbf{K_t}, D_t, \mathbf{M})$.
end for
return $\mathbf{M}$

We employ bayesian inference to construct our reputation mechanism. Each data prediction made by client c_i faces 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 predictions made by client c_i is α_i and β_i. Given $X_i = \gamma_i$, then the conditional probability is

$$Pr(Y_i|X_i = \gamma_i) = \left(\frac{\alpha_i + \beta_i}{\alpha_i}\right) {\gamma_i}^{\alpha_i} (1 - \gamma_i)^{\beta_i}. \quad (8)$$

α_i and β_i is the available evidence for the estimation of the X. γ_i is unknown. Equation 8 indicates the likelihood function for X. According to bayes' theory, we compute the posterior probability as

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

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

$$E(X) = \int_0^1 \frac{Pr\left(Y_i|X_i = \gamma_i\right) Pr(X_i = \gamma_i)}{\int_0^1 Pr\left(Y_i|X_i = x\right) Pr(X_i = x)dx} \gamma_i d\gamma_i. \quad (10)$$

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

$$f(x, \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1 - x)^{\beta-1}. \quad (11)$$

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

$$E(x) = \frac{\alpha}{\alpha + \beta}. \quad (12)$$

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 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

$$E(x_i)' = \frac{\alpha_i + {\alpha_i}'}{\alpha_i + {\alpha_i}' + \beta_i + {\beta_i}'}. \quad (13)$$

D. Aggregation Algorithms

Based on our reputation mechanism, we propose our Byzantine-tolerant aggregation algorithms: FLPhish-threshold and FLPhish-weight.

1) FLPhish-threshold: FLPhish-threshold as the aggregation rules identify the clients whose reputation is lower than the reputation threshold. Then the server discards the Byzantine clients and aggregate the global model using the survived clients' updates. Server identifies the client c_i whose reputation is lower than the threshold τ as Byzantine clients, and given them a aggregation weight as

$$\omega_i = \begin{cases} 1 & \text{if } x \geq \tau \\ 0 & \text{if } x < \tau \end{cases}. \quad (14)$$

The aggregated knowledge is given by

$$\hat{\mathbf{k}}^{\mathbf{t}} = \sum_{i=1}^{u} \frac{e_i}{\sum_{i=1}^{u} e_i} \hat{\mathbf{k}}_i^{\mathbf{t}} \times \omega_i, \quad (15)$$

$$\hat{\mathbf{y}}^{\mathbf{t}} \leftarrow argmax(\hat{\mathbf{k}}^{\mathbf{t}}). \quad (16)$$

2) FLPhish-weight: Unlike FLPhish-threshold, FLPhish-weight does not discard the potential Byzantine clients' updates. On the contrary, it enables the Byzantine clients to participate the aggregation using its reputation value as the aggregation weight. Due to our reputation mechanism, the Byzantine clients are offered a low value of reputation, so it

TABLE II
PARAMETER SETTINGS

Parameter	Parameter Value
Client Number	50 clients
Sample Total Number	60000 samples
Client Sample Number	800 training samples 200 test samples
Server Sample Number	8000 unlabeled samples 2000 labeled samples
Round Number	10 rounds
Round Sample Number	800 unlabeled samples 200 labeled samples
Deep Learning Model	Residual Networks
Byzantine Fractions p	0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9
Imbalance Degree q	0.1,0.2,0.5,0.6,0.7,0.8,0.9
Dataset d	MNIST,Fashion-MNIST,CIFAR-10

has a lower influence on aggregation. Give a reputation list $R = |x_1, x_1, x_1, \cdots, x_{n-1}, x_n|$ and the prediction results

$$\mathbf{k_i^t} = \begin{vmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,g-1} & p_{1,g} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,g-1} & p_{n,g} \end{vmatrix}, \tag{17}$$

FLPhish-weight computes the aggregated knowledge as

$$\hat{\mathbf{k}}^{\mathbf{t}} = \sum_{i=1}^{u} \frac{e_i}{\sum_{i=1}^{u} e_i} \hat{\mathbf{k}}_i^{\mathbf{t}} \times x_i. \tag{18}$$

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

$$\hat{\mathbf{y}}^{\mathbf{t}} \leftarrow argmax(\hat{\mathbf{k}}^{\mathbf{t}}). \tag{19}$$

In a word, FLPhish-weight does not identify the Byzantine clients but treats the clients with a low reputation as 'bad' clients and give them lower weight.

V. PERFORMANCE EVALUATION

In this section, we experimentally evaluate our FLPhish against untargeted attacks under different conditions. Furthermore, we compare our FLPhish's performance with FedAvg's performance, and the results show that FLPhish outperforms FedAvg in defending against Byzantine attacks.

A. Experiment Setup

1) The fractions p of Byzantine clients: We evaluate our FLPhish under the circumstances of different fractions p of Byzantine clients: 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 [12], 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 clients: The number of clients is set to 50 in our experiment.

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

5) The datasets d: We take three different datasets as our experiment datasets:

- MNIST: MNIST is a 10-class digit image classification dataset consisting of 60,000 training examples and 10,000 testing examples.
- Fashion-MNIST: Fashion-MNIST is a 10-class fashion image classification dataset. It has a predefined training set of 60,000 fashion images and a testing set of 10,000 fashion images.
- CIFAR-10: CIFAR-10 is a color image classification dataset. It consists of predefined 50,000 training examples and 10,000 testing examples. Each example belongs to one of the 10 classes.

Each client has 1,000 samples taken from the training dataset. Among the samples, 800 of them are used as training datasets, while another 200 are treated as test datasets. And the server has 10000 testing examples from the testing dataset.

6) Evaluated Byzantine Attacks: We evaluate our FLPhish-weight and FLPhish-threshold against two types of Byzantine attacks:

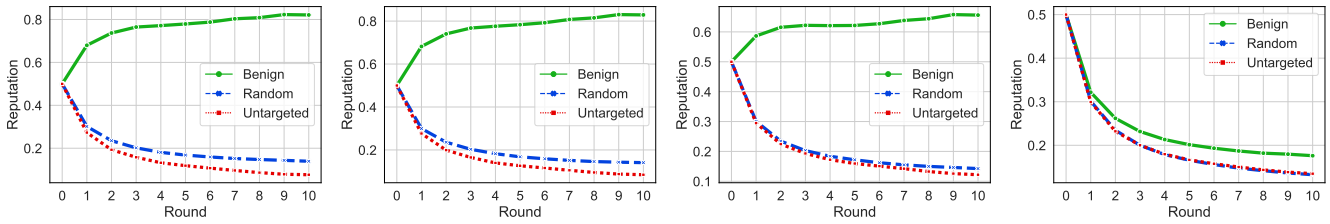
- Untargeted Byzantine Attacks: For each Byzantine client, it mislabels the data l to $(l-1)\ mod\ M$ to launch the attacks against FLPhish. The attack is known as the Label-flipping attack.
- Random Byzantine Attacks: For each Byzantine client, it mislabels the labels by returning a randomly chosen result.

7) Experiments Environment:: We conduct all the experiments on a laptop with Intel(R) Core(TM) i7-11800H CPU 2.30GHz and an NVIDIA GeForce RTX 3060 GPU with the video memory of 6 GB. We implement all deep learning models using Keras[3].

B. Performance Comparison under Random Byzantine Attacks

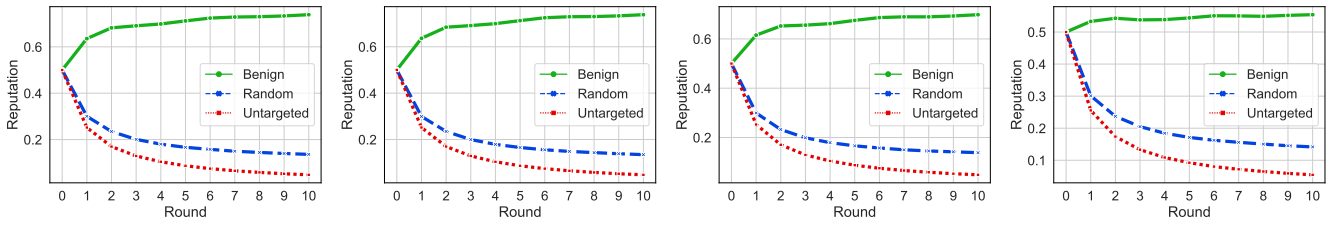
From Fig. 6 and Table IV, we can see that random Byzantine attacks have poor performance against FLPhish-weight and FLPhish-threshold (threshold=0.1, 0.2). We refer from the result that a random Byzantine attack has

[3]https://keras.io/



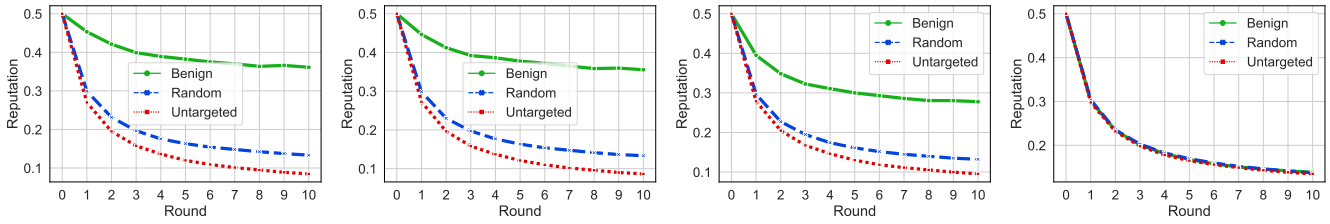
(a) $q = 0.1$ (b) $q = 0.2$ (c) $q = 0.5$ (d) $q = 0.9$

Fig. 3. Reputation values of experiments on MNIST on different imbalance degrees.



(a) $q = 0.1$ (b) $q = 0.2$ (c) $q = 0.5$ (d) $q = 0.9$

Fig. 4. Reputation values of experiments on Fashion-MNIST on different imbalance degrees.



(a) q =0.1 (b) q =0.2 (c) q =0.5 (d) q =0.9

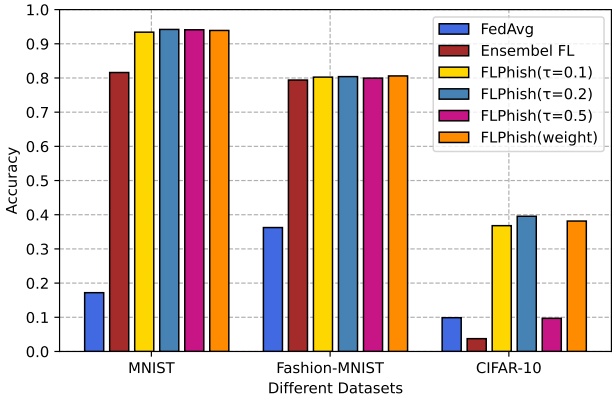
Fig. 5. Reputation values of experiments on CIFAR-10 on different imbalance degrees.

a drawback that it can not gather its influence at one data point like untargeted Byzantine attackers do. While we can see that it has good results in attacking FLPhish-threshold(threshold=0.5) in the CIFAR-10 dataset. From Fig. 5, we can see that the reputation of benign clients in CIFAR-10 experiments rapidly falls under the threshold of 0.5 due to the complexity of the tasks. With the increase of imbalance degree value q, the reputation of the benign clients falls more rapidly in the experiments of the CIFAR-10 dataset than in the experiments of the MNIST and the Fashion-MNIST dataset. As benign clients' reputations in the experiments of the CIFAR-10 dataset become under the threshold τ of 0.5, they are identified as malicious Byzantine clients as well. As many benign clients are falsely identified as Byzantine attackers, the FL server lacks enough trusted FL clients to assit the aggregation, therefore causing a bad performance of FLPhish-threshold (threshold=0.5). While, we can see that FLPhish-threshold (threshold=0.1), FLPhish-threshold (threshold=0.2) and FLPhish-weight stood still against the random Byzantine attacks. The results indicate that the value of the threshold τ should be considered carefully in the application in some complex tasks.

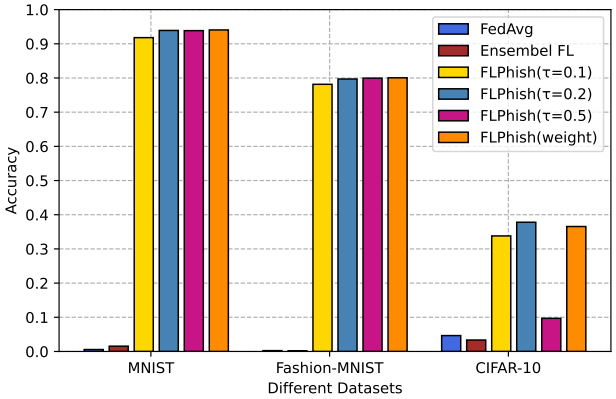
C. Performance Comparison under Untargeted Byzantine Attacks

We further evaluate FLPhish towards untargeted Byzantine attacks in Ensemble FL.

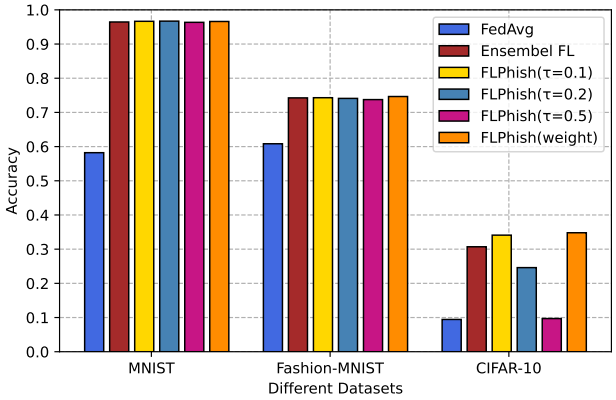
1) Performance Comparison with Different Distributions: We evaluate our FLPhish-threshold and FLPhish-weight under the condition where Byzantine client portion is a fixed value 0.5 and distribution imbalance value q is different across the Experiments. The experiment results are shown in Table VI. From Table VI, we can observe that both the FLPhish-threshold and FLPhish-weight outperform baseline until the 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-threshold with a threshold of 0.5 rapidly falls in this case. When facing



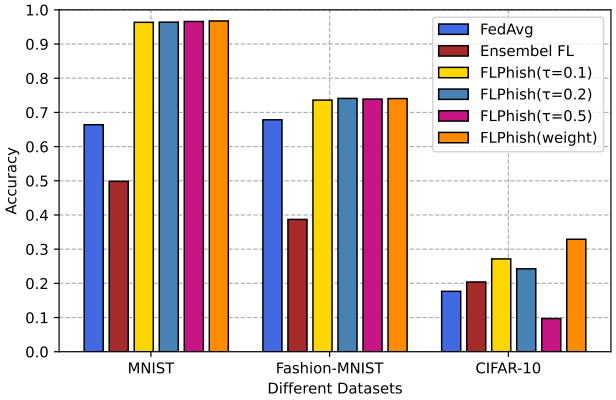
(a) Byzantine Fraction 0.9 & Imbalance Degree 0.2 & Random Attack



(b) Byzantine Fraction 0.9 & Imbalance Degree 0.2 & Untargeted Attack

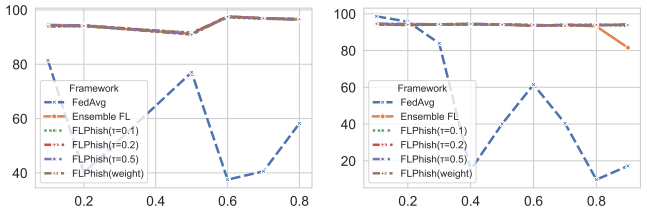


(c) Byzantine Fraction 0.5 & Imbalance Degree 0.9 & Random Attack



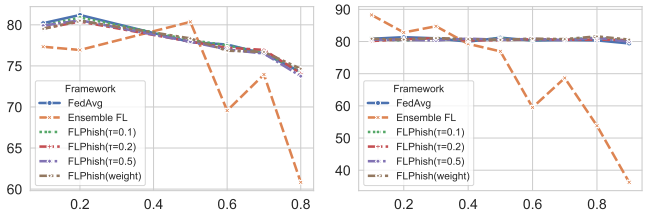
(d) Byzantine Fraction 0.5 & Imbalance Degree 0.9 & Untargeted Attack

Fig. 6. Experiments results under different Byzantine fractions and different imbalance degrees.



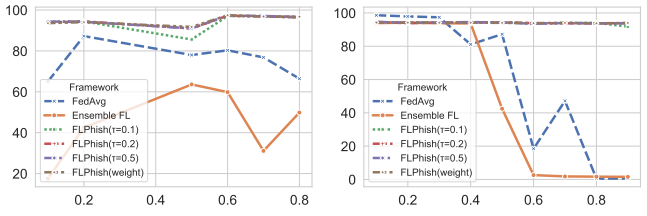
(a) $p\,0.5$ & Different q (b) $q\,0.2$ & Different p

Fig. 7. Accuracy values on MNIST under Random Attack.



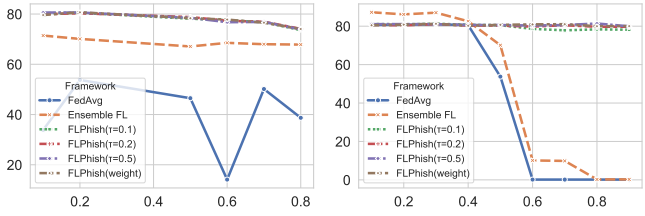
(a) $p\,0.5$ & Different q (b) $q\,0.2$ & Different p

Fig. 9. Accuracy values on MNIST under Random Attack.



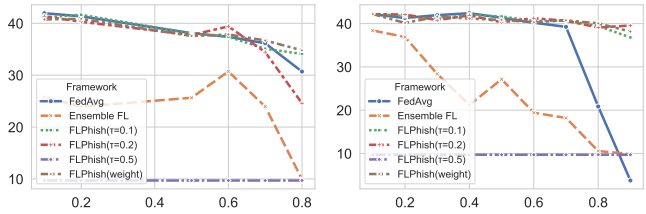
(a) $p\,0.5$ & Different q (b) $q\,0.2$ & Different p

Fig. 8. Accuracy values on MNIST under Untargeted Attack.



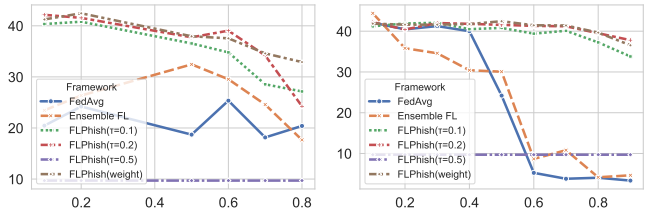
(a) $p\,0.5$ & Different q (b) $q\,0.2$ & Different p

Fig. 10. Accuracy values on MNIST under Untargeted Attack.



(a) $p\,0.5$ & Different q (b) $q\,0.2$ & Different p

Fig. 11. Accuracy values on MNIST under Random Attack.



(a) $p\,0.5$ & Different q (b) $q\,0.2$ & Different p

Fig. 12. Accuracy values on MNIST under Untargeted Attack.

the distribution of imbalance degree $q = 0.9$, each client performs badly, bringing a decline to its reputation. The rise of imbalance degree brings an explosive decline to the global model's accuracy. The accuracy of the global model under the FLPhish-threshold of threshold 0.5 stays 0.1 in the whole learning process. It means that the FLPhish-threshold of threshold 0.5 identifies all the clients as Byzantine clients, making the aggregation process invalid. We can see that the FLPhish-threshold of threshold 0.2 outperforms others. When the distribution becomes non-IID, the threshold of FLPhish-threshold 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 with Different Fractions of Byzantine Clients: Different fractions of Byzantine clients are taken into account as well. Table V shows that the accuracy for Ensemble FL without any defense mechanisms begins to fall rapidly when the Byzantine portion reaches nearly 50%. Furthermore, FedAvg performs extremely invalidly (the accuracy falls below 1%) when encountered with high fractions of Byzantine attackers. In comparison, FLPhish's (including FLPhish-threshold and FLPhish-weight) performance under various thresholds maintains a high level of performance as the portion grows. Both FLPhish-threshold and FLPhish-weight effectively detect Byzantine clients and accurately discards them from the aggregation process. The global model can be successfully trained without the involvement of Byzantine clients in aggregation procedures. Furthermore, the data show that FLPhish-threshold with a 0.1 threshold performs worse than the other two. This is due to the fact that the 0.1 threshold is far too low to adequately detect Byzantine clients. Though the Byzantine attackers will try to make right predictions of the public dataset and then send the opposite wrong predictions to the FL server. But they will also make wrong predictions first and coincidentally transfer the opposite right predictions to the FL server. Thus the reputation values of some Byzantine clients, in particular, can exceed 0.1. It indicates that if the threshold is not well set, Byzantine clients can avoid being detected by FLPhish-threshold. As the fractions of Byzantine clients go over the threshold of 50%, the harm brought by untargeted Byzantine attacks will increase rapidly for they outnumber the fractions of the benign clients.

VI. Conclusion

In this paper, we first design a FL architecture termed Ensemble Federated Learning in this study, which allows us to use the unlabeled dataset to transfer knowledge between the FL server and FL clients. We craft an FLPhish technique to make Ensemble FL resistant to Byzantine attacks by using a labeled dataset as 'bait' to detect malicious Byzantine clients. Furthermore, we propose a reputation technique based on Bayesian inference for determining a client's level of trust. We also present two aggregation techniques, FLPhish-threshold and FLPhish-weight, to improve FLPhish's performance. Finally, we test our suggested FLPhish in a variety of scenarios. The results of the experiment reveal that FLPhish performs admirably against Byzantine attacks. Even when confronted with very non-IID distributions and high fractions of Byzantine clients, FLPhish outperforms FedAvg and Ensemble FL significantly.

Our future work will be the design of more robust aggregation rules in Ensemble Federated Learning and evaluate them against more types of Byzantine attacks.

References

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Tech.*, vol. 10, no. 2, Jan. 2019.

[2] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5615–5624, Sep. 2021.

[3] Q. Kong, F. Yin, R. Lu, B. Li, X. Wang, S. Cui, and P. Zhang, "Privacy-preserving aggregation for federated learning-based navigation in vehicular fog," *IEEE Trans. Ind. Informat.*, Apr. 2021.

[4] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, Apr. 2020.

[5] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 911–926, Jul. 2020.

[6] A. Abrardo, M. Barni, K. Kallas, and B. Tondi, "A game-theoretic framework for optimum decision fusion in the presence of byzantines," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1333–1345, Feb. 2016.

[7] C.-Y. Wei, P.-N. Chen, Y. S. Han, and P. K. Varshney, "Local threshold design for target localization using error correcting codes in wireless sensor networks in the presence of byzantine attacks," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 7, pp. 1571–1584, Feb. 2017.

[8] X. Liu, T. J. Lim, and J. Huang, "Optimal byzantine attacker identification based on game theory in network coding enabled wireless ad hoc networks," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2570–2583, Feb. 2020.

[9] R. Cao, T. F. Wong, T. Lv, H. Gao, and S. Yang, "Detecting byzantine attacks without clean reference," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2717–2731, Jul. 2016.
[10] Y. Amir, C. Danilov, D. Dolev, J. Kirsch, J. Lane, C. Nita-Rotaru, J. Olsen, and D. Zage, "Steward: Scaling byzantine fault-tolerant replication to wide area networks," *IEEE Trans. Dependable Secure Comput.*, vol. 7, no. 1, pp. 80–93, Sep. 2010.
[11] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Model poisoning attacks in federated learning," in *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS)*, Palais des Congrès de Montréal, Montréal CANADA, Dec. 2-8 2018.
[12] 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)*, Boston, MA, USA, Aug. 12-14 2020.
[13] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, New York, NY, USA, Jun. 26-Jul. 1 2012.
[14] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, San Francisco, California, USA, Apr. 14-15 2008.
[15] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, Virtual, Aug. 26-28 2020.
[16] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, Nov 2019.
[17] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," in *Advances in Neural Information Processing Systems (NeurIPS)*. Virtual: Curran Associates, Inc., Dec. 6-12 2020.
[18] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *7th International Conference on Learning Representations (ICLR)*, New Orleans, USA, May 6-9 2019.
[19] 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 (NeurIPS)*, Long Beach, CA, USA, Dec. 4-9 2017.
[20] 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 (ICML)*, Stockholmsmässan, Stockholm SWEDEN, Jul. 10-15 2018.
[21] 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 (ICML)*, Long Beach, CA, USA, Jun. 9-15 2019.
[22] 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 (ICML)*, Stockholmsmässan, Stockholm SWEDEN, Jul. 10-15 2018.
[23] 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. 21-25 2021.
[24] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun*, pp. 1–1, Jul. 2020.
[25] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "Robust federated learning in a heterogeneous environment," *arXiv preprint arXiv:1906.06629*, Jun. 2019.
[26] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek, "On the byzantine robustness of clustered federated learning," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Catalonia, Spain, May. 4-8 2020, pp. 8861–8865.
[27] A. Portnoy and D. Hendler, "Towards realistic byzantine-robust federated learning," *arXiv preprint arXiv:2004.04986*, Apr. 2020.
[28] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," *arXiv preprint arXiv:1909.05125*, Sep. 2019.
[29] C. Xie, O. Koyejo, and I. Gupta, "SLSGD: Secure and efficient distributed on-device machine learning," in *Machine Learning and Knowledge Discovery in Databases (PKDD)*, Ghent, Belgium, Sep. 14-18 2020.
[30] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4574–4588, 2021.
[31] G. Liu, Q. Yang, H. Wang, and A. X. Liu, "Trust assessment in online social networks," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 2, pp. 994–1007, May 2021.
[32] B. Li, R. Lu, W. Wang, and K.-K. R. Choo, "DDOA: A dirichlet-based detection scheme for opportunistic attacks in smart grid cyber-physical system," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2415–2425, Jun. 2016.
[33] B. Li, R. Lu, and G. Xiao, *Detection of False Data Injection Attacks in Smart Grid Cyber-Physical Systems*. Springer, 2020.
[34] A. Das and M. M. Islam, "Securedtrust: A dynamic trust computation model for secured communication in multiagent systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 2, pp. 261–274, 2012.
[35] C. Zhu, H. Nicanfar, V. C. M. Leung, and L. T. Yang, "An authenticated trust and reputation calculation and management system for cloud and sensor networks integration," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 1, pp. 118–131, 2015.
[36] K. Lei, Q. Zhang, L. Xu, and Z. Qi, "Reputation-based byzantine fault-tolerance for consortium blockchain," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, Sentosa, Singapore, Dec. 11-13 2018, pp. 604–611.
[37] S. Chouikhi, L. Khoukhi, S. Ayed, and M. Lemercier, "An efficient reputation management model based on game theory for vehicular networks," in *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, Sydney, Australia, Nov. 16-19 2020, pp. 413–416.
[38] Y. Wen, Y. Huo, T. Jing, and Q. Gao, "A reputation framework with multiple-threshold energy detection in wireless cooperative systems," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, Virtual, Jun. 7-11 2020, pp. 1–6.
[39] J. Liang and M. Ma, "Ecf-mrs: An efficient and collaborative framework with markov-based reputation scheme for idss in vehicular networks," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 278–290, 2021.
[40] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, Tianjin, China, Dec. 4-6 2019.

Beibei Li (S'15-M'19) is currently an associate professor (doctoral supervisor) at the School of Cyber Science and Engineering, Sichuan University, P.R. China. He received his B.E. degree in communication engineering from Beijing University of Posts and Telecommunications, P.R. China, in 2014 and his Ph.D. degree in cybersecurity from Nanyang Technological University, Singapore, in 2019. He was invited as a visiting researcher at the Faculty of Computer Science, University of New Brunswick, Canada, from March to August 2018.

His research interests span several areas in security and privacy issues on cyber-physical systems (e.g., smart grids, industrial control systems, intelligent transportation systems, etc.), with a focus on intrusion detection techniques, artificial intelligence, and applied cryptography. He is serving or has served as a Publicity Chair, Publication Co-Chair, or a TPC member for several international conferences, including IEEE ICC, IEEE GLOBECOM, IEEE ICNC, IEEE ATC, and WCSP, etc. His works have been published in IEEE Transactions on Information Forensics and Security, IEEE Transactions on Industrial Informatics, ACM Transactions on Cyber-Physical Systems, IEEE Internet of Things Journal, Automatica, Information Sciences, IEEE ICC, and IEEE GLOBECOM, etc.

Peiran Wang is currently pursuing his B.E. degree in cybersecurity with the School of Cyber Science and Engineering, Sichuan University, Chengdu, P.R. China. His research interests includes IoT security, privacy preservation, federated learning and system security.

Lei Zhang received the Ph.D. degree from Sichuan University, Chengdu, Sichuan, China, in 2015. He is currently an Assistant Researcher with the School of Cyber Science and Engineering, Sichuan University. His current research interests include information security and machine learning.

Yuan Zhang (S'16-M'20) received his B.Sc. degree and Ph.D. degree in the University of Electronic Science Technology of China (UESTC) in 2013 and 2019, respectively. He was a visiting Ph.D. student from 2017 to 2019 with BBCR Lab, Department of ECE, University of Waterloo, Canada. He is currently an associate professor at the School of Computer Science and Engineering at UESTC. His research interests are applied cryptography, data security, and blockchain technology. He is a member of IEEE.

Rongxing Lu (S'09-M'11-SM'15-F'21) is an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from April 2013 to August 2016. Rongxing Lu worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He was awarded the most prestigious "Governor General's Gold Medal", when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. Also, Dr. Lu received his first PhD degree at Shanghai Jiao Tong University, China, in 2006. Dr. Lu is an IEEE Fellow. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise (with H-index 76 from Google Scholar as of May 2021), and was the recipient of 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Vice-Chair (Conferences) of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.

TABLE III
EXPERIMENTS RESULTS UNDER FIXED RANDOM BYZANTINE FRACTION=0.5 AND DIFFERENT IMBALANCE DEGREES q

Scheme	FedAvg			Ensemble FL			FLPhish(τ=0.1)			FLPhish(τ=0.2)			FLPhish(τ=0.5)			FLPhish(weight)		
q \ d	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR
0.1	81.43	77.33	25.76	94.25	80.20	41.95	93.90	79.90	41.15	94.20	79.55	41.35	94.50	79.90	9.70	93.90	79.50	40.75
0.2	40.06	76.94	23.94	94.05	81.20	41.35	94.00	80.90	41.65	94.35	80.40	40.30	94.30	80.55	9.70	94.15	80.50	40.85
0.5	77.04	80.36	25.65	91.20	77.90	38.10	91.70	78.15	38.05	91.45	77.95	37.75	91.10	77.95	9.70	90.90	78.35	37.55
0.6	37.51	69.56	30.72	97.65	77.60	37.40	97.25	77.50	37.40	97.60	77.20	39.4	97.35	77.00	9.70	97.50	76.85	37.85
0.7	40.57	73.95	23.96	96.80	76.50	36.15	96.70	76.60	35.10	96.95	76.95	34.45	97.00	76.50	9.70	96.90	76.70	36.70
0.8	58.22	60.83	9.94	96.45	74.25	30.70	96.65	74.30	34.10	96.70	74.10	24.60	96.35	73.75	9.70	96.60	74.65	34.80

TABLE IV
EXPERIMENTS RESULTS UNDER DIFFERENT RANDOM BYZANTINE FRACTIONS AND FIXED IMBALANCE DEGREE q=0.2

Scheme	FedAvg			Ensemble FL			FLPhish(τ=0.1)			FLPhish(τ=0.2)			FLPhish(τ=0.5)			FLPhish(weight)		
q \ d	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR
0.1	98.64	88.25	38.40	94.30	80.85	42.15	94.40	80.90	42.30	94.15	80.15	42.15	94.65	80.60	9.70	94.55	80.85	42.20
0.2	95.71	82.82	36.91	93.95	81.35	41.30	93.75	80.30	41.70	94.05	80.70	42.10	94.50	80.50	9.70	94.10	80.60	40.15
0.3	83.92	84.74	28.36	94.15	81.00	41.95	94.15	81.10	41.05	94.25	81.05	40.80	94.30	80.35	9.70	94.35	81.05	41.60
0.4	15.11	79.27	21.20	94.25	79.95	42.40	94.65	80.40	42.05	94.10	80.35	41.90	94.50	80.90	9.70	94.50	80.75	41.15
0.5	40.06	76.94	27.15	94.05	81.20	41.35	94.00	80.90	41.65	94.35	80.40	40.30	94.30	80.55	9.70	94.15	80.50	40.85
0.6	61.50	59.52	19.40	93.45	80.30	40.20	93.90	80.45	40.60	93.40	80.70	41.25	93.70	80.80	9.70	94.00	80.90	40.30
0.7	40.43	68.66	18.21	94.00	80.50	39.25	93.60	80.65	40.70	94.10	80.55	40.65	93.90	80.40	9.70	93.45	80.75	40.75
0.8	9.76	53.83	10.55	93.15	80.35	20.85	94.20	81.15	39.45	93.85	80.60	39.10	93.80	81.15	9.70	93.90	81.60	40.05
0.9	17.20	36.25	9.88	81.60	79.40	3.75	93.40	80.25	36.80	94.20	80.40	39.55	94.10	79.95	9.70	93.90	80.60	38.15

TABLE V
EXPERIMENTS RESULTS UNDER DIFFERENT UNTARGETED BYZANTINE FRACTIONS AND FIXED IMBALANCE DEGREE q=0.2

Scheme	FedAvg			Ensemble FL			FLPhish(τ=0.1)			FLPhish(τ=0.2)			FLPhish(τ=0.5)			FLPhish(weight)		
q \ d	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR
0.1	98.63	87.24	44.39	93.90	80.70	42.10	94.50	81.15	41.20	94.65	80.70	42.20	94.65	81.05	9.70	94.25	80.50	41.85
0.2	97.93	86.15	35.82	94.20	80.75	40.45	94.10	80.95	41.90	93.80	80.45	40.40	94.15	81.10	9.70	94.20	80.80	41.60
0.3	97.40	87.04	34.59	93.80	81.00	41.25	94.30	81.30	42.05	94.25	81.05	42.00	94.30	81.05	9.70	94.15	81.25	41.60
0.4	81.12	82.56	30.42	93.70	80.55	40.00	94.15	80.75	40.50	94.30	80.35	41.80	94.55	80.65	9.70	94.15	80.20	41.75
0.5	87.25	70.10	30.06	42.45	53.75	24.20	94.45	80.50	40.80	94.45	80.30	41.55	94.25	80.60	9.70	94.05	80.70	42.45
0.6	18.43	10.11	8.67	2.60	0.15	5.25	93.65	78.60	39.40	93.85	79.80	41.30	93.55	80.35	9.70	93.90	80.95	41.45
0.7	47.18	9.87	10.81	1.80	0.15	3.80	94.05	77.85	40.10	93.80	80.50	41.20	93.95	80.60	9.70	94.05	81.05	41.40
0.8	0.51	0.23	4.16	1.65	0.15	4.05	94.10	78.35	37.25	93.70	79.70	39.50	93.75	81.40	9.70	93.80	80.00	39.65
0.9	0.57	0.25	4.65	1.55	0.15	3.35	91.80	78.15	33.80	93.90	79.70	37.80	93.85	79.95	9.70	94.00	80.05	36.55

TABLE VI
EXPERIMENTS RESULTS UNDER FIXED UNTARGETED BYZANTINE FRACTION=0.5 AND DIFFERENT IMBALANCE DEGREES q

Scheme	FedAvg			Ensemble FL			FLPhish(τ=0.1)			FLPhish(τ=0.2)			FLPhish(τ=0.5)			FLPhish(weight)		
q \ d	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR	MNIST	Fashion	CIFAR
0.1	65.06	71.43	23.46	17.55	33.85	20.45	94.25	80.35	40.35	93.85	79.95	42.15	94.35	80.60	9.70	93.60	79.60	41.25
0.2	87.25	70.10	26.35	42.45	53.75	24.20	94.45	80.50	40.80	94.45	80.30	41.55	94.25	80.60	9.70	94.05	80.70	42.45
0.5	78.01	67.09	32.45	63.60	46.50	18.70	85.60	78.15	36.55	91.00	78.80	37.75	91.00	78.50	9.70	91.75	78.35	37.95
0.6	80.28	68.53	29.54	59.85	14.10	25.35	97.05	77.55	34.80	97.20	77.30	39.05	97.30	76.75	9.70	97.45	77.80	37.55
0.7	76.85	67.99	24.58	31.10	50.15	18.15	96.95	76.75	28.55	96.90	76.95	34.20	97.10	76.85	9.70	96.85	76.55	34.60
0.8	66.40	67.84	17.66	49.85	38.70	20.40	96.35	73.60	27.15	96.40	74.10	24.25	96.60	73.90	9.70	96.75	74.05	32.90