Collaborative Byzantine Resilient Federated Learning

A. Gouissem, K. Abualsaud, *Senior Member, IEEE*, E. Yaacoub, *Senior Member, IEEE*, T. Khattab, *Senior Member, IEEE*, and M. Guizani, *Fellow, IEEE*

***Abstract*—Federated learning (FL) enables an effective and private distributed learning process. However, it is vulnerable against several types of attacks, such as Byzantine behaviors. The first purpose of this work is to demonstrate mathematically that the traditional arithmetic-averaging model-combining approach will ultimately diverge to an unstable solution in the presence of Byzantine agents. This article also proposes a low-complexity, decentralized Byzantine resilient training mechanism. The proposed technique identifies and isolates hostile nodes rather than just mitigating their impact on the global model. In addition, the suggested approach may be used alone or in conjunction with other protection techniques to provide an additional layer of security in the event of misdetection. The suggested solution is decentralized, allowing all participating nodes to jointly identify harmful individuals using a novel cross check mechanism. To prevent biased assessments, the identification procedure is done blindly and is incorporated into the regular training process. A smart activation mechanism based on flag activation is also proposed to reduce the network overhead. Finally, general mathematical proofs combined with extensive experimental results applied in a healthcare electrocardiogram (ECG) monitoring scenario show that the proposed techniques are very efficient at accurately predicting heart problems.**

***Index Terms*—Byzantine attacks, convergence analysis, distributed learning, E-health, federated learning (FL).**

I. Introduction

FEDERATED Learning (FL) is a recent technique that has been proposed just five years ago in [1]. However, it has attracted a lot of attention in just few years thanks to its ability to enable efficient machine learning (ML) prediction models training while preserving the privacy of users' data [2]. FL can be very useful in several applications where the data is private and decentralized [3], [4], [5]. In particular, healthcare Internet of Things (IoT) networks constitute a major application domain for potential application domain for FL due to the widespread and heterogeneity of IoT networks in addition to the sensitivity of the private health-related data [6], [7], [8], [9].

However, the data privacy preservation in FL comes at the cost of a decreased level of transparency in model updates which makes it vulnerable to several types of attacks, including but not limited to, poisoning attacks, exploratory attacks, and evasion attacks [10]. In particular, the poisoning attacks are among the most common attacks in FL as they can use limited resources to degrade the performance of the training. Such attacks can be initiated by malicious users controlling one or multiple data owners in order to falsify the data or model updates. These users may inject falsified report updates [11], [12], data, labels, or even data sizes during the training phase [11], [13], [14], [15]. This allows malicious users to exploit the linearity of model aggregation to alter the classification decision into a specific target [11], [16]. To prevent the FL network vulnerabilities to Byzantine (adversarial) attacks, several strategies have been presented in the literature. These techniques can isolate malicious nodes and reports based on norm and weight control or distance-based analysis. They can even be detected using performance-based metrics and encryption mechanisms.

These methods may isolate malicious reports using weight or norm control. In particular, Portnoy et al. [13] proposed a model balancing approach based on data size truncation in order to combat a specific type of poisoning attack related to reporting large data volumes that could bias the aggregated model. Norm bounding is also used in [17] to truncate reports with unusually high norms. However, in FL networks, where the attackers' strength and weights are already constrained, such approaches could be ineffective.

A comparison of reports' distance may also be effective in identifying dishonest nodes. In particular, instead of the traditional arithmetic averaging, a stable variation of the gradient descent is presented to conduct model aggregation based on the geometric median in [15]. Another solution was proposed in [16] based on cross distance norm minimization to design an aggregation scheme named Krum that can be robust against a preset specific number of Byzantine attackers. In [18], a robust set of techniques is designed based on median and trimmed mean operations to provide order-optimal statistical error rates for highly convex losses and protect the FL training. The distance between the legitimate and poisoned reports is analyzed in [19] using autoencoders to learn the general training model

Manuscript received 13 November 2022; revised 28 January 2023; accepted 25 March 2023. Date of publication 11 April 2023; date of current version 7 September 2023. This work was supported by NPRP Award from the Qatar National Research Fund (a member of The Qatar Foundation) under Grant NPRP13S-0205-200270. The statements made herein are solely the responsibility of the authors. *(Corresponding author: K. Abualsaud.)*

A. Gouissem is with the College of Computing and Information Technology, University of Doha for Science and Technology, Doha, Qatar (e-mail: ala.gouissem@ieee.org).

K. Abualsaud and E. Yaacoub are with the Department of Computer Science and Engineering, Qatar University, Doha, Qatar (e-mail: k.abualsaud@ieee.org; eliasy@ieee.org).

T. Khattab is with the Department of Electrical Engineering, Qatar University, Doha, Qatar (e-mail: tkhattab@ieee.org).

M. Guizani is with the Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE (e-mail: mguizani@ieee.org).

Digital Object Identifier 10.1109/JIOT.2023.3266347

trend in order to identify Byzantine nodes in an iteratively distributed fashion. Performance comparison methods present other strategies that can protect against poisoning attacks by using a centralized validation data set that is publicly accessible. For example, the malicious nodes' reports are weighted based on a ReLU-clipped cosine similarity trust score computed in FLTrust [20] with reference to a centralized validation data set. However, such methods make use of a centralized validation data set or make impractical assumptions on the perfect knowledge of legitimate node reports. Additionally, a number of strategies in the literature make use of detection that requires complex iterations.

The protection from Byzantine attackers can also be performed by modifying the training algorithm itself. For example, Li et al. [14] proposed a robust decentralized algorithm that can combat Byzantine data falsification by using a modified version of the alternating direction method of multipliers (ADMMs) algorithm. Some other techniques make use of key sharing and encryption mechanisms to protect FL. For example, So et al. [21] used a key sharing and participants selection mechanisms based on private pairwise distance calculations. However, such approaches might be challenging to implement in resource-constrained networks.

The key concern when evaluating and combating the threat from Byzantine attackers in [16] is based on the linearity of the FL model aggregation and the assumption that the hostile nodes can perfectly eavesdrop the other users' reported updates. This allows malicious nodes to report specific updates that cause the model to converge to a false target. In addition, most of the above Byzantine combating techniques aim to design an aggregation mechanism or criteria (e.g., geometric mean or Krum) that tries to minimize the effect of the falsified reports on the global model without actually identifying the hostile node. Furthermore, because data set volumes are growing at an exponential rate, effective prediction, and classification models often require a high number of layers and neurons. Therefore, since most of these techniques are based on the computation and processing of the cross-distance weight norm between all the participating data owners, the centralized protection mechanism becomes a heavy computational process, particularly when the involvement of different agents is large.

We show in this article that even if a single Byzantine attacker is unaware of the collaborative updates and sends random reports, the entire model can diverge to extreme nonuseful values. A low-complexity decentralized Byzantine robust training mechanism is proposed thereafter. First, the proposed mechanism aims to identify the malicious nodes instead of alleviating their effect on the global model. This can give the network administrators enough information to exclude such participants from the FL network when necessary. The proposed scheme can be applied either separately or in conjunction with traditional Byzantine combating schemes to add an extra layer of protection in case of misdetection. The proposed approach is decentralized and allows all the participating nodes to collaboratively identify malicious users. To avoid biased evaluations, the identification process is incorporated into the standard training process, causing all the nodes to blindly participate in the collaborative evaluations. To reduce network overhead caused by the cross-evaluation, a smart activation mechanism is designed based on specific flags that indicate the potential appearance of Byzantine nodes in the network.

The main contributions of this work are summarized as follows.

1) Investigate the effect of blind Byzantine attack on FL model convergence.
2) Design a low complexity, decentralized Byzantine resilient FL training mechanism.
3) Design a collaborative blind cross evaluation mechanism that allows the identification of malicious users.
4) Propose a smart activation mechanism to reduce the evaluation network overhead.
5) Validate the efficiency of the proposed scheme with a convergence mathematical proof.
6) Validate the accuracy of the proposed scheme in predicting heart diseases with experimental simulations in a healthcare monitoring scenario using the Physikalisch-Technische Bundesanstalt database electrocardiogram (PTBDB ECG) data set.

The remainder of this article is organized as follows. Section II presents the adopted system model. The convergence of the federated averaging scheme in the presence of blind Byzantine attackers is investigated in Section III. The proposed collaborative cross-check mechanism is then detailed in Section IV. The numerical results are presented in Section V to evaluate the performance of the proposed scheme. Finally, Section VI concludes the article.

II. System Model

As depicted in Fig. 1, the adopted system model consists of a set of N_U healthcare facilities denoted by $N_1, \ldots, N_{N_U}$. A private data set related to patients' electrocardiogram (ECG) readings is assumed to be locally stored in each of these facilities. The objective in the sequel is to exploit the available data to design an automated artificial neural network (ANN) model that can efficiently predict whether a patient has a heart disease based on his/her ECG readings. Each hospital or healthcare facility can train a local model based on its own data. However, such an approach would lack prediction accuracy due to the limited number of available data entries at each facility. Consequently, FL is used to let all the nodes collaborate in designing a global prediction model without having to share the patient's private data.

The satisfaction of the privacy constraints during the model training process comes at the cost of a raised vulnerability to several potential attacks, such as poisoning and Byzantine attacks. N_B nodes are assumed to be compromised and trying to mislead the global model by sending falsified model updates to the server.

Each data owner N_i aims to train an ANN model that minimizes its cross entropy loss function $\mathcal{L}_i(.)$ [22], [23] computed based on its locally stored data set $\mathfrak{D}_i$. For an ANN model with weights $\boldsymbol{w}$, the combined loss function $\mathcal{L}(\boldsymbol{w})$ is defined as

$$\mathfrak{L}(\boldsymbol{w}) = \sum_{i=1}^{N_U} \sigma_i \mathfrak{L}_i(\boldsymbol{w}) \tag{1}$$

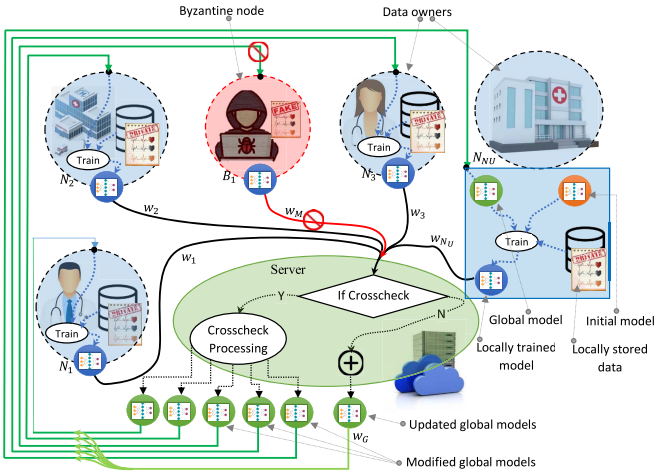

Fig. 1. System model.

where σ_i denotes the ratio of the data set $\mathfrak{D}_i$ size compared to the cumulative size of all data sets of the participating agents ($\sum_{i=1}^{N_U} \sigma_i = 1$).

The convergence of the model training when using the synchronous distributed gradient descent learning algorithm, also referred to as FedSGD [22], [24], [25] is investigated in the presence of Byzantine nodes. A blind crosscheck mechanism is also proposed to enable the different nodes to blindly identify the malicious node and protect the model design.

III. Limitations of Federated Averaging Under Byzantine Attacks

A. Model Update Process

The server begins the learning by constructing a random baseline ANN model with weights $\boldsymbol{w}_G^0$. This model is shared with all N_U data owners, who utilize it to retrain the ANN model by making use of the locally stored data. Using the FedSGD [25] with a stepsize μ, each data owner N_i generates its weights vector $\boldsymbol{w}_i^t$. These weights are computed to minimize the local loss while using the received global model $\boldsymbol{w}_G^{t-1}$ as [22], [24], [25]

$$\boldsymbol{w}_i^t = \boldsymbol{w}_G^{t-1} - \mu \nabla \mathcal{L}_i\left(\boldsymbol{w}_G^{t-1}\right) \tag{2}$$

where $\nabla f(.)$ denotes the gradient of the function $f(.)$.

In the Byzantine free case, the models received by the server at time t are combined into a global model $\boldsymbol{w}_G^t$ as follows:

$$\boldsymbol{w}_G^t = \sum_{i=1}^{N_U} \sigma_i \boldsymbol{w}_i^t. \tag{3}$$

When the network is under Byzantine attack from N_B malicious nodes, the model update is done as follows:

$$\boldsymbol{w}_G^t = \sum_{i=1}^{N_U} \hat{\sigma}_i \boldsymbol{w}_i^t + \sum_{i=1}^{N_B} \sigma_{B_i} \boldsymbol{w}_{B_i}^t \tag{4}$$

where $\boldsymbol{w}_{B_i}^t$ $(i = 1 \dots N_B)$ denotes the falsified report shared by the Byzantine node B_i. Also, σ_i' denotes the ratio of the data set $\mathfrak{D}_i$ size compared to the cumulative reported data set sizes of both legitimate and malicious users. Similarly, σ_{B_i} corresponds to the ratio of the reported size by B_i, relative to the cumulative legit and malicious data set sizes.

B. Convergence Proofs and Analysis

In this section, we show that even a single Byzantine node unaware of the FL model updates and just sending random reports would cause the whole FL model to diverge. In fact, we prove in the sequel that the global model will not only be nonaccurate, but it will totally diverge to nonuseful solutions.

Theorem 1: The federated averaging model under Byzantine attacks will diverge independently from the number of attackers, the number of legit nodes, and their report powers, even if the Byzantine node just sends a random message.

Proof: The proof is done by contradiction. In particular, we assume that the FL model is close to convergence (i.e., $\nabla \mathcal{L}_i(\boldsymbol{w}_G^{t-1}) \approx 0 \quad \forall i = 1 \cdots N_U$). and we prove that the model will diverge to nonuseful values. ■

From (2), it follows that when the FL model is close to convergence, the different local models become stable, i.e.,

$$\boldsymbol{w}_i^t = \boldsymbol{w}_G^{t-1} - \mu \nabla \mathfrak{L}_i\left(\boldsymbol{w}_G^{t-1}\right) \approx \boldsymbol{w}_G^{t-1}. \tag{5}$$

By aggregating the stable local models, the global model at time t can be approximated by

$$\boldsymbol{w}_G^t = \overbrace{\left(\sum_{i=1}^{N_U} \sigma_i'\right) w_G^{t-1}}^{w_L^t} + \overbrace{\sigma_B w_B^{t-1}}^{w_M^t}, \quad \text{where} \sum_{i=1}^{N_U} \sigma_i' + \sigma_B = 1. \tag{6}$$

Depending on the power of the reports, the global model can be interpreted differently for the below three cases.

1) *Case 1:* $\left\|w_M^t\right\| << \left\|w_L^t\right\|$.
2) *Case 2:* $\left\|w_M^t\right\| >> \left\|w_L^t\right\|$.
3) *Case 3:* $\left\|w_M^t\right\| \approx \left\|w_L^t\right\|$.

1) Case 1 (Small Malicious Contribution): When $\left\|w_M^t\right\| << \left\|w_L^t\right\|$, the malicious reports' weights can be neglected and (6) can be approximated as follows:

$$w_G^t \approx \sum_{i=1}^{N_U} \sigma_i w_G^{t-1}. \tag{7}$$

Consequently, after n iterations, the global weights would become

$$w_G^{t+n} \approx \left(\sum_{i=1}^{N_U} \sigma_i\right)^n w_G^{t-1}. \tag{8}$$

Since $(\sum_{i=1}^{N_U} \sigma_i) = 1 - \sigma_M < 1$, it follows that after a large number of iterations, W_G^∞ becomes

$$W_G^\infty = \lim_{n \to \infty} \left(\left(\sum_{i=1}^{N_U} \sigma_i \right)^n W_G^{t-1} \right) = 0. \tag{9}$$

This proves first that the global model diverges when the malicious reported weights are very small compared to the legit ones (case 1). It also shows that if the global model gets close to the optimal solution then a Byzantine node with small weights appears, the global model will eventually diverge to 0.

2) Case 2 (Small Legit Contribution): When $\|w_M^t\| >> \|w_L^t\|$, it follows from (6) that:

$$\boldsymbol{w}_G^t = \sigma_B w_B^{t-1}. \tag{10}$$

This shows that the malicious node has full control over the global model and that the global weight would converge to the random model suggested by the malicious node.

3) Case 3: When $\|W_M^t\| \approx \left\|W_G^{t-1}\right\|$, no term can be ignored in comparison to the other.

Assume that $w_{G_i}^t$ and $w_{B_i}^t$ are the weights of the global and Byzantine models, respectively, at time t. This section analyzes the variance of $w_{G_i}^t$ variation during the distributed training process to determine the model convergence status. Due to the independence of the Byzantine model from the global model, the individual weights variance update can be expressed as follows:

$$v_i^{t+1} = \sigma_L^2 v_i^t + \sigma_B^2 b_i^t \tag{11}$$

where

$$v_i^t = E\left(w_{G_i}^t\right), \quad b_i^t = E\left(w_{B_i}^t\right), \quad \sigma_L = \sum_{i=1}^{N_U} \sigma_i'. \tag{12}$$

Proposition 1: By assuming that the statistical properties of the random Byzantine reports do not change over time, the variance of the global model weights will converge, i.e.,

$$\forall i \quad \text{If } \forall t,\ b_i^{t+1} = b_i^t,\ \exists \tilde{v}_i,\ \text{s.t. } \lim_{t\to\infty}\left(v_i^t\right) = \tilde{v}_i. \tag{13}$$

Proof: See Appendix A ∎

Assuming $d_i^{t+1} = v_i^{t+1} - v_i^t$, we demonstrate that $\tilde{v}_i$ corresponds to the extreme zero model by evaluating the various probable convergence spots. In particular, when the expression of d_i^{t+1} is analyzed as a function of v_i^t, it can be proven that

$$\begin{cases} d_i^{t+1} < 0, & \text{if } v_i^t > \bar{v}_i \\ d_i^{t+1} > 0, & \text{if } v_i^t < \bar{v}_i \\ d_i^{t+1} = 0, & \text{if } v_i^t = \bar{v}_i \end{cases} \tag{14}$$

where

$$\bar{v}_i = \frac{b_i^t \sigma_B}{\sigma_L + 1}. \tag{15}$$

According to (14), if $v_i^t > \bar{v}_i$, v_i^t will continue to decrease until it reaches $\bar{v}_i$. Furthermore, if $v_i^t < \bar{v}_i$, v_i^t will continue to grow continuously until it reaches $\bar{v}_i$. Additionally, the last term in (14) indicates that the only value that could maintain the stability of v_i^t ($v_i^{t+1} = v_i^t$) is $v_i^t = \bar{v}_i$. This indicates that if v_i^t converges, it will converge to $\bar{v}_i$. When combining this result with Proposition 1, it follows that:

$$\lim_{t\to\infty}\left(v_i^t\right) = \frac{b_i^t \sigma_B}{\sigma_L + 1}. \tag{16}$$

Note that (16) reveals an interesting fact which is that the variance of the weight after convergence (v_i^t) is linearly dependent on the variance of the Byzantine model (b_i^t). This indicates that the Byzantine node has full control over the final model weights variance. For example, just by sending very high reports, the model will diverge to infinity. Similarly, if the Byzantine reports very small weights, the global model will converge to zero. Such behavior can be easily detected with weight norm checking. However, what is more important here is that even if the Byzantine node sends weights in the normal range, the limit of v_i^t is, generally, very close to zero. In fact, the limit expression in (16) is proportional to the ratio of the Byzantine data size (σ_B) to the full data size added to all the legitimate data sizes combined ($1 + \sigma_L$). For instance, if ten nodes, including one malicious agent, are involved in the distributed training, and if all the nodes have almost the same reported data size, then the variance v_i^t would converge to just 5% of the original b_i^t. As a result

$$\lim_{t\to\infty}\left(v_i^t\right) \approx 0. \tag{17}$$

This concludes the proof of Theorem 1 and confirms that when only one malicious node is present in the FL network, using the traditional weights averaging would result in a model with weights that are most of the time equal to zero, or just random values reported by the malicious node in some extreme scenarios. In all cases, the FL can never converge to any useful model.

IV. Proposed Byzantine Resilient Mechanism

A. Motivation

A low-complexity cooperative aggregation scheme is proposed to identify the malicious nodes in this article. The identification process is done in a blind decentralized way that allows all the nodes to participate in the identification process with less complexity and without actually knowing that they are evaluating each other's trustworthiness. This technique can also be used in conjunction with the schemes existing in the literature to combat Byzantine attacks in the event of a misdetection.

B. Trustworthiness

To identify malicious/Byzantine nodes, mutual trustworthiness $\mathfrak{T}_{i,j}$ is defined between each couple of nodes N_i and N_j. In particular, $\mathfrak{T}_{i,j}$ is a computed expectation of the contribution of N_j on the global model loss according to node N_i. Equivalently, $\mathfrak{T}_{i,j}$ reflects how much node N_i trusts node N_j. Mathematically speaking, the trustworthiness $\mathfrak{T}_{i,j}$ at time t is defined as follows:

$$\mathfrak{T}_{i,j}^t = \mathfrak{L}_{\mathrm{i}}\left(w_G^{t-1}\right) - \mathfrak{L}_{\mathrm{i}}\left(w_j^t\right). \tag{18}$$

Since the global objective of FL is to minimize the loss function, a higher value of $\mathfrak{T}_{i,j}$ corresponds to a high level of trust. The cooperative global trust of a node N_j is defined as the average trust level of the node N_j according to all the remaining nodes in the network

$$\begin{aligned} \mathfrak{T}_i^t &= \frac{1}{N-1} \sum_{i=1, i\neq j}^{N} \mathfrak{T}_{i,j} \\ &\approx \mathfrak{L}\left(w_{G_j}^{t-1}\right) - \mathfrak{L}\left(w_j^t\right). \end{aligned} \tag{19}$$

By evaluating the trustworthiness of each node, Byzantine nodes can be identified and isolated to improve the global

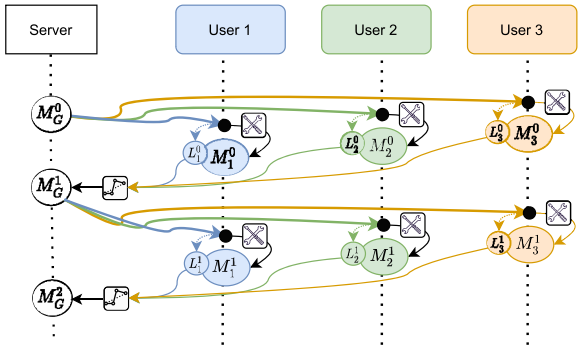

Fig. 2. Model training operation mode.

Algorithm 1 *Data_Owner_Training*(w_G)

1: $Loss \leftarrow \mathfrak{L}_i(w_G)$
2: Train the ANN model using the local data $\mathfrak{D}_i$ starting from w_G
3: $w_i \leftarrow$ Updated model
4: Send w_i and $Loss$ to the server

model. However, the computation of this trust level is challenging because the data that would be used to evaluate a model of another node is only available locally. Also, if a bunch of malicious nodes realize that they are actually evaluating the performance of the other nodes, they can simply report high trust levels to the other Byzantine devices. Therefore, a secure effective and blind trustworthiness cross evaluation mechanism is proposed in the sequel.

C. Cross-Evaluation Mechanism

In order for the cross-check mechanism to be blind, a node should not be able to recognize whether it is actually evaluating another node's performance or not. In particular, similar to a standard FL model, each device should receive a global model, try to improve it using the local data and then send it back to the server. Therefore, the blind cross-check should be incorporated in this standard FL model training process without any extra steps that would make the Byzantine nodes suspicious.

The evaluation is done by defining two possible operation modes for the FL network, namely:

1) model training mode;
2) malicious nodes identification mode.

As detailed in Fig. 2, during the model training mode, the server sends the up-to-date global model to the data owners. Each node updates the received model based on its own local data and sends it back to the server. The server combines all the locally trained models to a single global model.

To keep track of the training progress, and as detailed in Algorithm 1 each node reports the computed loss of the global model before training it. This allows the server to evaluate the global model loss and to stop the training when a maximum loss threshold is reached.

Algorithm 2 presents the regular update at each iteration at the server side when the system is operating in the model training mode. The regular update uses as inputs all the collected

Algorithm 2 *Regular_Update*()

1: **if** *LoadMdl* **then**
2: $\quad w_G \leftarrow \sum_{i\in\mathfrak{S}_L} \sigma_i w_i^{Best}$
3: **else**
4: $\quad w_G \leftarrow \sum_{i\in\mathfrak{S}_L} \sigma_i w_i$
5: $\quad \mathfrak{L} \leftarrow \sum_{i\in\mathfrak{S}_L} \sigma_i \mathfrak{L}_i$
6: **end if**
7: **if** $\mathfrak{L} < \mathfrak{L}_{max}$ *or* $k > k_{max}$ **then**
8: $\quad$ break
9: **end if**
10: **if** *Not*(*LoadMdl*) & $\mathfrak{L} \leq \mathfrak{L}_{best}$ **then**
11: $\quad \mathfrak{L}_{best} \leftarrow \mathfrak{L}$
12: $\quad w_i^{Best} \leftarrow w_i \quad \forall\ 1 \leq i \leq N_U$
13: **end if**

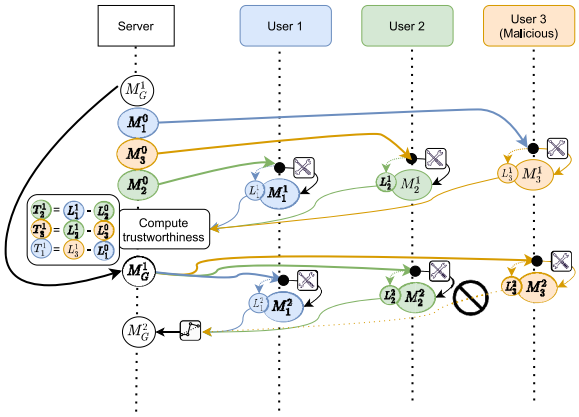

Fig. 3. Malicious nodes identification operation mode.

models w_i from all the data owners as well as an input variable denoted by *LoadMdl* that indicates that the server should go back and load the best-reported model so far.

The *LoadMdl* variable will be mainly used in the second operation mode after the end of the identifications. However, in the first mode, the server simply aggregates all the collected models (line 4) and all the reported losses (line 5). If the computed loss is below the preset threshold ($\mathfrak{L}_{\max}$) or if the number of iterations exceeds $k_{\max}$, the training stop and the best reported model is shared with the data owners. Otherwise, the lowest combined loss and its corresponding models are stored at $\mathfrak{L}_{\text{best}}$ (line 11) and w_i^{Best} (line 12), respectively. The computed global model is then shared with the data owners for further updates. In the next rounds, the updates will be based only on the computed set of legitimate users ($\mathfrak{S}_L$) as it will be detailed in the sequel.

Fig. 3 details the identification process in the second operation mode. In particular, assume that the last computed global model at time $t-1$ before switching to malicious nodes identification operation mode is equal to w_G^{t-1}, i.e., at time $t-1$, w_G^{t-1} is sent to all the participating nodes to retrain it using their local data. As detailed above, each data owner N_i computes the loss of the reported global model based on its own data $\mathfrak{L}_i(w_G^{t-1})$, retrain it to a new local model w_i^{t-1} and reports both to the server. By the end of time slot $t-1$, the server would have collected the evaluation of all the nodes to the global model ($\mathfrak{L}_i(w_G^{t-1}), \quad i = 1 \cdots N$).

Algorithm 3 *Init*()

1: $w_G \leftarrow BaselineModel()$
2: $Count \leftarrow 0$
3: $\mathfrak{S}_L \leftarrow 1..N_U$
4: $\mathfrak{S}_B \leftarrow \emptyset$
5: $postchek \leftarrow 0$
6: $\mathfrak{T} \leftarrow 0$
7: $k_0 \leftarrow 1$

As detailed in Fig. 3, when the network is switched to identification mode, instead of updating the global model and sending the up-to-date version to all the nodes, at time t, each node N_i receives the local model w_j^{t-1} of another random node N_j. By thinking that it is just a new global model, the node N_i computes the loss of the jth node model at time $t-1$ (w_j^{t-1}) based on its own local data. This local loss is then reported with the updated model to the server. Consequently, by the end of time slot t, the server collects the computed losses $\mathfrak{L}_{\mathrm{i}}(w_{r_i^t}^t), \quad i = 1 \cdots N$, where r_i^t denotes the index of the random node that has its model sent to node N_i at time t.

The server can then compute the cross trustworthiness of node $N_{r_i^t}$ based on node N_i data as

$$\mathfrak{T}_{i,r_i^t}^t = \mathfrak{L}_{\mathrm{i}}\left(w_G^{t-1}\right) - \mathfrak{L}_{\mathrm{i}}\left(w_{r_i^t}^t\right). \tag{20}$$

In particular, node $N_{r_i^t}$ retrained the model from w_G^{t-1} to $w_{r_i^t}^t$. Since the losses of these two models computed based on the local data of node N_i are available to the server, it can compute the contribution of $N_{r_i^t}$ on the global model improvement according to N_i. All this is done without any node knowing that it is actually evaluating another device's performance. Since $\cup_{i=1}^{N}(r_i^t) = \{1 \ldots N\}$, each device will get a random evaluation from one other node at each cross checking process.

D. Periodic and Smart Activation Mechanisms

Switching between the two operation modes is controlled by the server without any knowledge of the data owners. Two switching mechanisms are proposed in this article to activate the Byzantine nodes identification mode.

1) Periodic Cross Check: When the periodic cross check (PCS) activation mechanism is adopted, the network keeps operating in the normal learning mode for N_P iterations. In particular, Algorithm 3 describes the process at the server that initializes the baseline global model and all the required variables.

In Algorithm 4, the server uses the collected models from the data owners and keeps training the model till the stopping criterion detailed above in the regular update algorithm (see Algorithm 2) is reached.

The server update process operates differently in three scenarios, namely, regular update, check, and post-check scenarios. First, check scenario is activated periodically after each *Period* iterations starting from k_0 (See line 4) and indicates the starting of the Malicious nodes identification mode by setting the variable *Chk* to true. The post-check scenario is activated using the variable *PstCheck* after collecting all the

Algorithm 4 *PCS_Server_Update*()

1: *Init*()
2: **while** true **do**
3: $LoadMdl \leftarrow 0$
4: $Chk \leftarrow (mod(k - k_0, Period) == 0 \ \& \ k \geq k_0)$
5: $RegUpdt \leftarrow Not(Chk \ or \ PstChk)$
6: **if** *RegUpdt* **then**
7: *Regular_Update*()
8: **end if**
9: **if** *PstChk* **then**
10: *Post_Check*()
11: **end if**
12: **if** *Chk* **then**
13: $RU \leftarrow Random\_Order(1 : N_U)$
14: $w_G^i \leftarrow w_{RU(i)}, \quad \forall 1 \leq i \leq N_U$
15: $\mathfrak{L}_i^{prev} \leftarrow \mathfrak{L}_i$
16: **else**
17: $w_G^i \leftarrow w_G, \quad \forall 1 \leq i \leq N_U$
18: **end if**
19: $Send(w_G^i)$ to N_i $\forall 1 \leq i \leq N_U$
20: $Receive(\mathfrak{L}_i, w_i)$ from N_i $\forall 1 \leq i \leq N_U$
21: $PstChk \leftarrow Chk$
22: $k \leftarrow k + 1$
23: **end while**
24: Send(w_G) *to* N_i $\forall i, \ 1 \leq i \leq N_U$

models when the system is back to the model training mode. Equivalently, *PstCheck* is set true for the next iteration whenever *Chk* is set to true (see line 21). This allows the processing of the collected cross evaluation metrics. If neither *PstCheck* nor *Chk* are set to true, a regular update is performed by activating *RegUpdt*.

When *RegUpdt* is enabled (line 7 of Algorithm 4), the server performs regular model aggregation update as detailed in Algorithm 2. When *Chk* is activated, the cross-check evaluation is performed first by creating a vector *RU* that contains the indices of all the participating users in a random order.[1] Instead of reporting a unique global model to all users, the global model for user N_i is defined as the reported local model for $N_{RU(i)}$ ($w_{RU(i)}$). This allows each data owner to compute the loss of another user's model based on its local data. The reported loss from each user is then stored to evaluate its evolution in the next iteration. After receiving the updated models from the data owners related to the crafted global models (line 20), the *PstChk* phase is activated.

As detailed in Algorithm 5, the post-check process is performed by computing a new trust level for each user N_i defied as $\mathfrak{T}_{\text{new}}(RU(i))$ defined as the contribution of the user N_i in the loss of random user $N_{RU(i)}$. Because the trust level can vary when computed by one user or another, and to avoid sudden fluctuations of the trust level, the final trust level considered in the identification of malicious nodes is computed by taking into consideration all the previous evaluations in a decreasing

[1]The function *Random_Order* is defined so that to make sure an index cannot remain in the same position, i.e., a user cannot be assigned to take the model from himself.

Algorithm 5 *Post_Check*()

1: **for** $i = 1..N_U$ **do**
2: $\quad \mathfrak{T}_{new}(RU(i)) \leftarrow \mathfrak{L}_i^{prev} - \mathfrak{L}_i$
3: **end for**
4: $\mathfrak{T} \leftarrow \alpha\mathfrak{T}_{new} + (1-\alpha)\mathfrak{T}$
5: $\mathfrak{S}_L \leftarrow find(\mathfrak{T} > 0)$
6: $\mathfrak{S}_B \leftarrow find(\mathfrak{T} < 0)$
7: $LoadMdl \leftarrow 1$

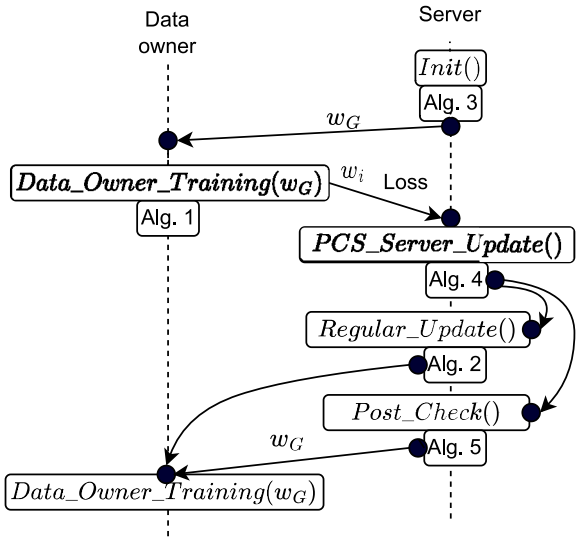

Fig. 4. PCS collaborative training.

factor (line 4). Any user with a positive trust level is then considered as legitimate (line 5) and the remaining ones are considered as malicious nodes (line 6). The computed sets of legitimate ($\mathfrak{S}_L$) and malicious ($\mathfrak{S}_B$) users are then used in the model aggregation to include only legit model updates. Fig. 4 summarizes the PCS collaboration and flag update process detailed in Algorithms 1–5.

2) Smart Cross Check: The second activation mechanism denoted by smart cross check (SCS) avoids wasting the network time in identifying Byzantine nodes when the FL model is operating normally. The identification mode is activated using specific flags that detect suspicious behavior. In particular, as proven in Section III, when FedAvg is used to combine the received reports, the FL training process becomes very sensitive to Byzantine attacks and diverges even in the presence of a single malicious device. Also, as proven in Section III, the divergence in such scenarios occurs mainly because of the extreme evolution of the model weights norm most probably to zero. Therefore, two flags are defined to detect potential existence of Byzantine nodes. The first flag is denoted by weight fluctuation flag (WFF) and detects sudden extreme fluctuations of the model weights norm. The second flag is denoted by the loss fluctuation flag (LFF) and detects sudden deterioration of the model loss.

The model update process when SCS is adopted is performed in a similar way to PCS but by replacing line 4 of Algorithm 4 with Algorithm 6 to activate the variable *chk* using the above-mentioned flags. In particular, the WFF and

Algorithm 6 *SCS_StatusChk*()

1: **if** $k > k_0$ **then**
2: $\quad WFF \leftarrow Not\Big((1-\gamma_{th}) \leq \|w_0\|/w_G \leq (1+\gamma_{th})\Big)$
3: $\quad LFF \leftarrow \mathfrak{L}/\mathfrak{L}_{best} > (1+\gamma_{th})$
4: $\quad$ **if** *WFF or LFF* **then**
5: $\quad\quad chk \leftarrow 1$
6: $\quad\quad k \leftarrow 0$
7: $\quad\quad k_0 \leftarrow k_{min}^{SCS}$
8: $\quad$ **else**
9: $\quad\quad chk \leftarrow 0$
10: $\quad$ **end if**
11: **else**
12: $\quad chk \leftarrow 0$
13: **end if**

LFF flags are activated when the norm of the global model and loss are not within a normal fluctuation range controlled by the parameter γ_{th} (line 2). In addition, to have accurate decisions, the algorithm is allowed to activate the cross-check only if at least $k_{\min}^{\text{SCS}}$ iterations have passed. Such mechanism avoid inaccurate decisions that might follow the post-check phase where the global model may vary a lot by isolating the Byzantine nodes and moving back to the best reported model.

E. Convergence Analysis

This section provides a proof that the proposed cross-check mechanism enables a model train process that converges to the optimal solution. This is done first by analyzing the monotony and the convergence of the loss function assuming successful Byzantine detection. The efficiency of the trust computation in differentiating the malicious from the legitimate nodes is then investigated. Finally, a mathematical proof is presented to validate the sensitivity of the defined flags to Byzantine attacks. The convergence analysis is performed under the assumptions that the loss function is smooth and convex as detailed below.

Assumption 1: A realistic and widely adopted assumption in the literature consists of having the loss function $\mathfrak{L}(.)$ β smooth, i.e.,

$$\|\nabla\mathfrak{L}(y) - \nabla\mathfrak{L}(x)\| \leq \beta\|y-x\| \quad \forall \quad x, y \in \mathbb{R}^{N_{ANN}}. \tag{21}$$

As stated in [24], this assumption is also equivalent to

$$\mathfrak{L}(y) \leq \mathfrak{L}(x) + < \nabla\mathfrak{L}(x), y-x > + \frac{\beta}{2}\|y-x\|^2 \quad x, y \in \mathbb{R}^{N_{ANN}}. \tag{22}$$

Assumption 2: If the loss function is differentiable, assuming that it is convex is equivalent to the following inequality:

$$\mathfrak{L}(y) \geq \mathfrak{L}(x) + \langle \nabla\mathfrak{L}(x), y-x\rangle. \tag{23}$$

Proposition 2: If all the Byzantine nodes are identified, and there is at least one identified legitimate user, there exists a stepsize μ that guarantees

$$\text{if} \quad \Big(\mathfrak{S}_B \subset \hat{\mathfrak{S}}_B \iff \hat{\mathfrak{S}_L} \subset \mathfrak{S}_L\Big) \text{ and } \hat{\mathfrak{S}_L} \neq \emptyset$$
$$\hat{\mathfrak{L}}\big(w_G^t\big) - \hat{\mathfrak{L}}\Big(w_G^{t-1}\Big) \leq -\frac{\mu}{2}\Big\|\nabla\hat{\mathfrak{L}}\Big(w_G^{t-1}\Big)\Big\|^2 \tag{24}$$

where $\hat{\mathfrak{L}}$ denotes the loss computed based on the data of identified legitimate users, and $\hat{\mathfrak{L}}$ satisfies Assumptions 1 and 2. The sets $\hat{\mathfrak{S}}_L$ and $\hat{\mathfrak{S}}_B$ denote the set of legitimate and Byzantine nodes, respectively.

Proof: See Appendix B. ∎

Theorem 2: Under Assumptions 1 and 2, if all the Byzantine nodes are identified, and there is at least one identified legitimate user, the FL model will eventually converge to the optimal solution, i.e.,

$$\text{if} \quad \left(\mathfrak{S}_B \subset \hat{\mathfrak{S}}_B \iff \hat{\mathfrak{S}}_L \subset \mathfrak{S}_L\right) \text{ and } \hat{\mathfrak{S}}_L \neq \emptyset,$$
$$\lim_{t\to+\infty}\left(\hat{\mathfrak{L}}\left(w_G^t\right)\right) = \hat{\mathfrak{L}}\left(\hat{w}^*\right) \tag{25}$$

where $\hat{w}^*$ is the optimal solution that corresponds to the lowest loss $\hat{\mathfrak{L}}$.

Proof: See Appendix C. ∎

Note that Theorem 2 guarantees that if all the Byzantine nodes are detected, the model will converge to the optimal solution in terms of $\hat{\mathfrak{L}}$ and not in terms of $\mathfrak{L}$. Equivalently, the obtained solution is the optimal based only on the data of the nodes identified as legitimate. In addition, the convergence proof in Theorem 2 is performed under the condition of perfect detection of all the Byzantine nodes. Therefore, Theorems 3 and 4 investigate the ability of the proposed approach to differentiate between the legitimate and malicious nodes.

Theorem 3: Under Assumption 3 and in the presence of at least one Byzantine node, there exists a stepsize μ that guarantees the perfect distinction between all the Byzantine and legitimate nodes, i.e.,

$$\mathfrak{T}_{i,j} \geq 0, \quad \text{if } j \in \mathfrak{S}_L \tag{26}$$
$$\mathfrak{T}_{i,j} \leq 0, \quad \text{if } j \in \mathfrak{S}_M. \tag{27}$$

Assumption 3: Similar to [26], we assume in this section that the gradient norm is bounded

$$\exists \gamma_G > 0, \quad \text{such that, } \left\|\mathfrak{L}\left(\boldsymbol{w}_G^t\right)\right\| \leq \gamma_G \quad \forall t \geq 0.$$

Proof: See Appendix D. ∎

Theorem 3 confirms that there exists a stepsize that guarantees not only the perfect detection of all Byzantine nodes, but also all the perfect detection of all legitimate nodes.

In the absence of Byzantine nodes, Theorem 2 confirms that any selected nodes will eventually drive the model to convergence. Theorem 4 investigates in the sequel the ability of the model to detect the absence of Byzantine nodes and the ability to identify all the nodes as legitimate.

Theorem 4: When the network is Byzantine free or when all the Byzantine nodes are isolated, there exists a stepsize μ and a small positive number ϵ that guarantee the following:

$$\mathfrak{T}_{i,j} \geq -\epsilon, \quad \text{if } j \in \mathfrak{S}_L. \tag{28}$$

Proof: See Appendix E. ∎

Theorems 3 and 4[2] confirm that with the adequate choice of stepsize, all the Byzantine node can be detected. Also, when the network is Byzantine free or when all the malicious nodes are already detected and isolated, there is small risk of false alarm when the model gets close to convergence. However, as proven Theorem 2, small chances of false alarm do not stop the convergence as long as there is no misdetection ($\mathfrak{S}_B \subset \hat{\mathfrak{S}}_B$).

Since the cross check is done periodically for PCS, Theorem 3 confirms that all the Byzantine nodes will be detected if the stepsize is well selected. Therefore, Theorem 2 confirms that PCS will make the FL model converge to the optimal solution.

However, for SCS, the cross check is activated by specific flags. Therefore, it is necessary to prove that these flags are efficient and at least one of them will raise in the presence of a Byzantine node.

Corollary 1: In the event of appearance of a Byzantine node, the SCS norm fluctuation flag will be eventually activated to detect the malicious presence.

Proof: According to Theorem 1, depending on the FL model and Byzantine nodes characteristics, the FL will eventually diverge in the presence of any number of Byzantine nodes. Also, the model can diverge in any of the three following ways:

$$a: \lim_{t\to+\infty}\left\|w_G^t\right\| = 0, \quad \text{or} \tag{29}$$
$$b: \lim_{t\to+\infty} w_G^t = w_M^t. \tag{30}$$

First, the divergence of the FL to the Byzantine nodes model w_M^t (divergence of type a) occurs only when the norm of the malicious combined reports is much bigger than the norm of the legitimate data owners. Such type of attack can be easily overcome by truncating the reported data sizes as in [13] or by eliminating suspicious models with very high values compared to other reported values. Also, such scenario would result in the loss function to converge to a very high number (loss of the byzantine model), therefore, it can be eventually detected using the LFF.

By definition of the norm limit to zero, it follows that:

$$\lim_{t\to+\infty}\left\|w_G^t\right\| = 0 \iff$$
$$\forall x > 0, \quad \exists t_1 > 0 \text{ such that} \left\|w_G^t\right\| < x \quad \forall t > t_1. \tag{31}$$

In particular, let $x = \left\|w_G^{t_0}\right\|(1-\gamma_{th})$, therefore

$$\exists t_1 > 0 \text{ such that } \frac{\left\|w_G^t\right\|}{\left\|w_G^{t_0}\right\|} < (1-\gamma_{th}) \quad \forall t > t_1. \tag{32}$$

This confirms that if the Byzantine nodes are causing a divergence of type b, the norm fluctuation flag will eventually be activated. ∎

In addition, concerning the LFF, there is no guarantee that it will actually detect the presence of malicious nodes. However, if the FL model diverges to zero, it will eventually go far from the optimal solution and the loss would most probably increase to high values. Therefore, the loss fluctuations flag is used in conjunction with the weights norm fluctuation flag to accelerate the detection of the Byzantine nodes.

[2]Note that Theorem 4 is not necessary to prove the convergence of the model and is used just to confirm that the model can operate with low false alarm probabilities in the absence of malicious nodes.

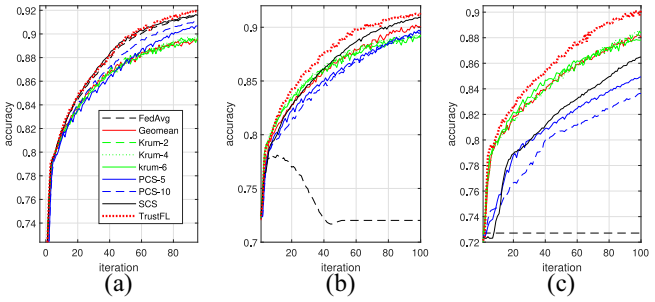

Fig. 5. Prediction accuracy using FL under Byzantine attack. (a) Byzantine-free. (b) One Byzantine node. (c) Five Byzantine nodes.

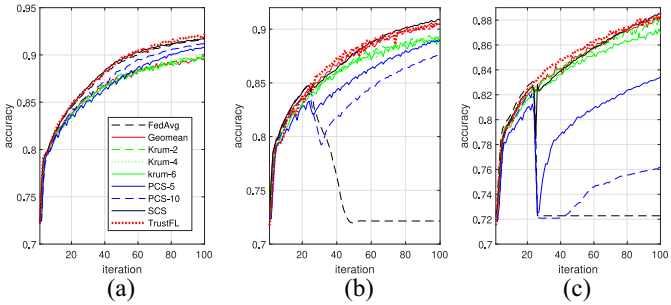

Fig. 6. Prediction accuracy using FL when Byzantine nodes join the network during the training. (a) Byzantine-free. (b) One Byzantine node. (c) Five Byzantine nodes.

V. Numerical Results Using the PTBDB ECG Data Set

The ability of PCS and SCS in diagnosing cardiac diseases based on ECG recordings is examined in this part to evaluate and confirm the effectiveness of the designed Byzantine robust training mechanism. The simulations are done using the PTBDB ECG data set [27] that includes 14 552 samples with 16-lead ECGs. The PTBDB ECG data set includes a variety of cardiac abnormalities, such as cardiomyopathy, heart failure, bundle branch block, dysrhythmia, myocardial hypertrophy, valvular heart disease, and myocarditis. To mimic the experience of collaborative training, the data is split into $N = 10$ equal parts, one for each user. Following this, 75% of the data is used to train the model, while the remaining 25% is utilized to evaluate the performance of the trained models. Python Tensorflow and Keras libraries are used to implement the FL process using Anaconda Spyder platform.

The performances of SCS and PCS are compared in this section with the following techniques.

1) FedAvg [25], where the aggregation is performed using a standard arithmetic averaging mechanism.
2) Geometric mean (GeoMean), where the aggregation is done using a geometric mean cross distance minimization scheme as detailed in [15].
3) Krum-k, which refers to the aggregation approach described in [16], where the Krum metric is used to eliminate k possibly Byzantine reports after comparing the cross weight distance among all the node pairs.
4) FLTrust [20], where a centralized data set is assumed to be known by the server and used to compute a trustworthiness score with cosine similarity and ReLU operations.

Fig. 5 illustrates the evolution of the global model accuracy in detecting cardiac illnesses over time using the PCS, SCS, FedAvg, GeoMean, Krum-2, Krum-4, Krum-6, and FLTrust aggregation mechanisms. First, 5(a) investigates the Byzantine free scenario. In such case, the best accuracy is obtained by performing arithmetic aggregation of all the models. As a result, FedAvg outperforms all other investigated approaches, which suffer from a slight loss of accuracy for attempting to combat nonexistent hostile nodes. In fact, by assuming the existence of malicious agents, the GeoMean and Krum approaches impose an additional inaccuracy throughout the aggregation process. However, FLTrust, is able provide the same performance of FedAvg since it isolates nodes based on trustworthiness score computed using a centralized data set available at the server. Note also that by detecting the absence of malicious behavior in the network using the designed flags, SCS provides a very close performance to FedAvg. The proposed PCS also provides a close-optimal accuracy but with a slower convergence rate. In particular, PCS loses some of its iterations to evaluate the trustworthiness of the nodes and make sure that the training network is Byzantine free. That is why PCS-5 turns out to be slower than PCS-10 for example.

Fig. 5(b) demonstrates that, as shown in Section III, even when a single-blind Byzantine node is attacking the FL training, FedAvg diverges and produces a model with very poor accuracy in comparison to Byzantine robust approaches. Fig. 5(b) shows also that all the investigate Byzantine robust aggregation schemes succeed in maintaining an efficient training with a loss of one to two percent when 10% of the agents are malicious. Note also, that SCS provides the highest accuracy compared to GeoMean and Krum techniques since the proposed scheme eliminates the causes of the problem by isolating malicious nodes instead of reducing their effects using mathematical metrics. Although PCS-5 and PCS-10 succeed in identifying malicious nodes after the activation of the cross-check mode, they lag behind as they lose many iterations in redetecting the same hostile nodes. When five Byzantine nodes (50%) are participating in the training, after 100 iterations, GeoMean provides a better performance than SCS as the flags of SCS fail to detect the performance deterioration before ten iterations. In fact, since 50% malicious nodes are present at the beginning of the simulation, the training starts with a very bad performance at the first few iterations, which makes it slightly challenging for the WFF and LFF to perform detection. However, once the hostile nodes are detected, the accuracy of SCS starts increasing at a higher rate by eliminating and isolating the malicious reports.

Fig. 6 analyzes the effect of Byzantine attacks when the malicious nodes start their hostile behavior during the training process. In particular, no attacker is assumed to be present in the network till iteration 22. In such a scenario, it can be seen that PCS loses a lot of iterations before detecting the threat. GeoMean and Krum however remain robust and SCS can almost instantly detect the malicious behavior and continue the training unaffected. Additionally, Fig. 6(b) and (c) demonstrate how fragile the FedAvg is. For instance, FedAvg's performance fell immediately with the appearance of

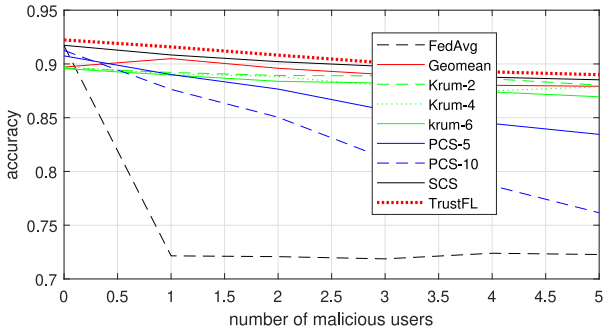

Fig. 7. Effect of the number of Byzantine agents on the prediction accuracy.

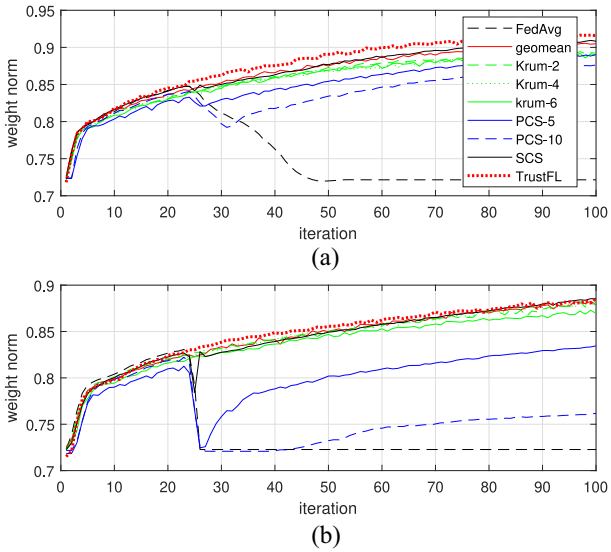

Fig. 8. Weight norm evolution. (a) One Byzantine node. (b) Five Byzantine nodes.

the Byzantine agent. Although this behavior has a significant impact on the training efficiency of FedAvg, it is very important for the SCS flags to quickly detect the malicious behaviors. Note that SCS can provide the same accuracy of FLTrust while outperforming it in terms of data requirements. In particular, to reach high accuracy levels, FLTrust assumes the existence of a centralized data set on the server that is publicly available, which might be challenging in practical scenarios.

Fig. 7 reveals the effect of the number of Byzantine agents in the FL network on the accuracy of the global model. First, it can be seen that by isolating the source of the malicious reports, SCS holds the best performance compared to Krum and GeoMean that only aim to minimize the effect of the introduced error with a specific aggregation process. It can be seen also that because of the long time spent in the cross check, PCS would require more time to converge.

According to the proof of Theorem 1, the primary cause of FedAvg divergence is global weight norm fluctuation. To corroborate this trend, Fig. 8 examines the global weight norm fluctuation as shown in Theorem 1, after the Byzantine node began acting maliciously at iteration 22, the weights norm began diverging toward zero. This result is in accordance with the accuracy behavior shown in Fig. 5. In addition, Fig. 8

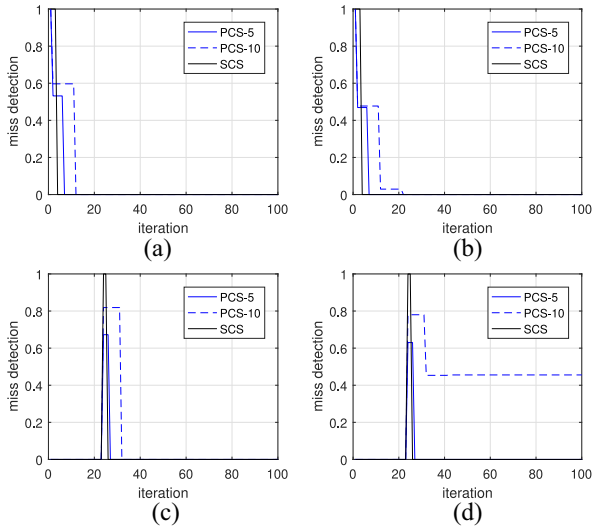

Fig. 9. Effect of Byzantine attacks on misdetection probability. (a) Byzantine node from iteration 0. (b) Five Byzantine nodes from iteration 0. (c) One Byzantine node from iteration 22. (d) Five Byzantine nodes from iteration 22.

shows that if PCS spends a lot of time before detecting the Byzantine node, the model weight norm starts dropping similar to FedAvg.[3]

Fig. 9 presents the misdetection probability evolution over time for PCS-5, PCS-10, and SCS. First, it can be seen that the three techniques fail to detect the Byzantine node at the exact time when they joined. In particular, the detection for PCS-5 and PCS-10 require up to five and ten iterations, respectively. The SCS requires only one iteration after the cross check activation to isolate the Byzantine. Interestingly, Fig. 9(d) shows that when the number of Byzatine nodes is equal to 5 out of 10, PCS-10 has a high chance of misdetecting some of them. In particular, the global model would have started to diverge and it might be too late for the cross-check to differentiate between legit and malicious nodes.

VI. Conclusion

In conclusion, this article investigates the effect of blind Byzantine attacks on FL model convergence and designs a number of solutions to address this issue. Specifically, we prove that the traditional arithmetic averaging approach is susceptible to diverge due to Byzantine behaviors, even when malicious agents provide random reports. Simulation findings demonstrate that in the presence of compromised nodes, this approach's performance will deteriorate significantly. To address this issue, we propose the use of FedAvg only when the network administrator is sure about the nodes' identities. Additionally, we design a low complexity, decentralized Byzantine resilient FL training mechanism and a collaborative blind cross-evaluation mechanism that allows for the identification of malicious users. Furthermore, we propose a smart activation mechanism to reduce the evaluation network overhead. Our theoretical proofs and simulation results, using the PTBDB ECG data set, confirm the ability of the proposed

[3]Note that the best model loading is not activated in this figure.

cross-check approaches to converge to optimal or suboptimal solutions even in the presence of Byzantine nodes, making it a suitable method for healthcare monitoring scenarios. In summary, we believe that the findings of this article make significant contributions to the field of FL security by investigating the effect of blind Byzantine attacks on FL model convergence, designing a low complexity, decentralized Byzantine resilient FL training mechanism, designing a collaborative blind cross-evaluation mechanism that allows the identification of malicious users, proposing a smart activation mechanism to reduce the evaluation network overhead, and validating the efficiency of the proposed scheme with a convergence mathematical proof and experimental simulations in a healthcare monitoring scenario using the PTBDB ECG data set.

APPENDIX A

Let $d_i^{t+1} = v_i^{t+1} - v_i^t$, it follows from (11) that at any instant time t_0:

$$d_i^{t+2} = \left(\sigma_L^2 - 1\right)\left(\sigma_L^2 v_i^t + \sigma_B^2 b_i^t\right) + \sigma_B^2 b_i^t = \sigma_L^{2n} d_i^{t_0}. \quad (33)$$

The factor σ_L is by definition smaller than one. Therefore, when n goes to infinity, d_i^t will also tend to zero, i.e.,

$$\lim_{t\to\infty}\left(d_i^t\right) = \lim_{n\to\infty}\left(d_i^{t_0+n}\right) = \lim_{n\to\infty}\left(\sigma_L^{2n} d_i^{t_0}\right) = 0. \quad (34)$$

This suggests that in the presence of Byzantine agents, the variance of the model weights will eventually stabilize and converge to a fixed value, i.e.,

$$\exists\ \tilde{v}_i,\ \text{s.t.}\ \lim_{t\to\infty}\left(v_i^t\right) = \tilde{v}_i. \quad (35)$$

This concludes the proof of Proposition 1.

APPENDIX B

Let the identified set of legitimate ($\hat{\mathfrak{S}}_L$) and Byzantine users ($\hat{\mathfrak{S}}_B$) guarantee $\left(\mathfrak{S}_B \subset \hat{\mathfrak{S}}_B \iff \hat{\mathfrak{S}}_L \subset \mathfrak{S}_L\right)$ and $\hat{\mathfrak{S}}_L \neq \emptyset$. Once all the Byzantine nodes are identified, they can be isolated and the model update would be computed based on the reported models of $\hat{\mathfrak{S}}_L \subset \mathfrak{S}_L$

$$\boldsymbol{w}_G^t = \sum_{i\in\hat{\mathfrak{S}}_L} \hat{\sigma}_i \boldsymbol{w}_i^t = w_G^{t-1} - \mu\nabla\hat{\mathfrak{L}}\left(w_G^{t-1}\right) \quad (36)$$

where $\hat{\sigma}_i$ denotes the ratio of the data size for node N_i compared to all the nodes in $\hat{\mathfrak{S}}_L$ that satisfies

$$\sum_{i\in\hat{\mathfrak{S}}_L} \hat{\sigma}_i = 1,\ \hat{\mathfrak{L}}(w) = \textstyle\sum_{i\in\hat{\mathfrak{S}}_L} \hat{\sigma}_i \mathfrak{L}_i(w). \quad (37)$$

By assuming that the loss function $\mathfrak{L}_i$ is β smooth, it follows that $\hat{\mathfrak{L}}$ is also β smooth. Consequently, by applying the smoothness property between w_G^t and w_G^{t-1}, it follows that:

$$\begin{aligned}\hat{\mathfrak{L}}(w_G^t) - \hat{\mathfrak{L}}\left(w_G^{t-1}\right) &\le \frac{\beta}{2}\left\|w_G^t - w_G^{t-1}\right\|^2 \\ &+ \left\langle\nabla\hat{\mathfrak{L}}\left(w_G^{t-1}\right), w_G^t - w_G^{t-1}\right\rangle \\ &\le \left(\frac{\beta\mu^2}{2} - \mu\right)\left\|\nabla\hat{\mathfrak{L}}\left(w_G^{t-1}\right)\right\|^2. \end{aligned} \quad (38)$$

Therefore, any stepsize μ smaller than $(1/\beta)$ guarantees a strictly monotonically decreasing loss evolution, i.e.,

$$\begin{aligned}\hat{\mathfrak{L}}(w_G^t) - \hat{\mathfrak{L}}\left(w_G^{t-1}\right) &\le -\delta\left\|\nabla\hat{\mathfrak{L}}\left(w_G^{t-1}\right)\right\|^2 \\ &\le -\frac{\mu}{2}\left\|\nabla\hat{\mathfrak{L}}\left(w_G^{t-1}\right)\right\|^2 \end{aligned} \quad (39)$$

where

$$\delta = \mu - \frac{\beta\mu^2}{2} > \frac{\mu}{2}. \quad (40)$$

This concludes the proof of Proposition 2.

APPENDIX C

Let $\hat{w}^*$ denote the optimal solution that minimizes $\hat{\mathfrak{L}}$, i.e.,

$$\hat{\mathfrak{L}}(\hat{w}^*) \le \hat{\mathfrak{L}}(w)\ \forall\ w. \quad (41)$$

By applying the convexity inequality (Assumption 2) between the two weights w_G^{t-1} and $\hat{w}^*$, it follows that:

$$\hat{\mathfrak{L}}(\hat{w}^*) \ge \hat{\mathfrak{L}}\left(w_G^{t-1}\right) + \left\langle\nabla\hat{\mathfrak{L}}\left(w_G^{t-1}\right), \hat{w}^* - w_G^{t-1}\right\rangle. \quad (42)$$

Consequently

$$\hat{\mathfrak{L}}\left(w_G^{t-1}\right) - \mathfrak{L}(\hat{w}^*) \le \left\langle\nabla\hat{\mathfrak{L}}\left(w_G^{t-1}\right), w_G^{t-1} - \hat{w}^*\right\rangle. \quad (43)$$

By combining the results of Proposition 2 and (43), it follows that:

$$\begin{aligned}&\hat{\mathfrak{L}}\left(\hat{W}_G^t\right) - \hat{\mathfrak{L}}(\hat{w}^*) = \hat{\mathfrak{L}}\left(\hat{W}_G^t\right) - \hat{\mathfrak{L}}\left(W_G^{t-1}\right) + \hat{\mathfrak{L}}\left(w_G^{t-1}\right) - \hat{\mathfrak{L}}(\hat{w}^*) \\ &\le \left\langle\nabla\hat{\mathfrak{L}}\left(w_G^{t-1}\right), w_G^{t-1} - \hat{w}^*\right\rangle - \frac{\delta}{2}\left\|\nabla\hat{\mathfrak{L}}\left(W_G^{t-1}\right)\right\|^2 \\ &\le \left\langle\nabla\hat{\mathfrak{L}}\left(w_G^{t-1}\right), w_G^{t-1} - \hat{w}^*\right\rangle - \frac{\bar{\mu}}{2}\left\|\nabla\hat{\mathfrak{L}}\left(W_G^{t-1}\right)\right\|^2 \\ &\frac{1}{2\bar{\mu}}\left(\left\|w_G^{t-1} - \hat{w}^*\right\|^2 - \left\|w_G^{t-1} - \bar{\mu}\nabla\hat{\mathfrak{L}}\left(W_G^{t-1}\right) - \hat{w}^*\right\|^2\right) \\ &\frac{1}{2\bar{\mu}}\left(\left\|w_G^{t-1} - \hat{w}^*\right\|^2 - \left\|w_G^t - \hat{w}^*\right\|^2\right). \end{aligned} \quad (44)$$

By summing (44) from $t = 1$ to $t = t_N$, the inequality becomes

$$\begin{aligned}\sum_{t=1}^{N_T}\left(\hat{\mathfrak{L}}\left(\hat{W}_G^t\right) - \hat{\mathfrak{L}}(\hat{w}^*)\right) &\le \frac{1}{2\bar{\mu}}\left(\left\|w_G^0 - \hat{w}^*\right\|^2 - \left\|w_G^{t_N} - \hat{w}^*\right\|^2\right) \\ &\le \frac{1}{2\bar{\mu}}\left(\left\|w_G^0 - \hat{w}^*\right\|^2\right). \end{aligned} \quad (45)$$

Since $\hat{\mathfrak{L}}$ is a monotonically decreasing function, it follows that:

$$\hat{\mathfrak{L}}\left(\hat{W}_G^{t_N}\right) - \hat{\mathfrak{L}}(\hat{w}^*) \le \frac{\left\|w_G^0 - \hat{w}^*\right\|^2}{2\bar{\mu}t_N}. \quad (46)$$

Since $\hat{w}^*$ is defined as the optimal solution, then it follows that:

$$\hat{\mathfrak{L}}\left(\hat{W}_G^{t_N}\right) - \hat{\mathfrak{L}}(\hat{w}^*) \ge 0. \quad (47)$$

Consequently

$$\lim_{t_N\to+\infty}\left(\hat{\mathfrak{L}}\left(\hat{w}_G^{t_N}\right)\right) = \hat{\mathfrak{L}}(\hat{w}^*). \quad (48)$$

This concludes the proof of Theorem 2 and confirms that after a large number of iterations, the loss function will eventually reach its global minimum.

APPENDIX D

By considering the training of the ANN model by the data owner N_i at time t, the model is transformed from w_G^{t-1} to w_i^t using gradient descent. Therefore, the updated data owner model is expressed by

$$w_i^t = w_G^{t-1} - \mu \nabla_i^{t-1} \tag{49}$$

where $\nabla_i^{t-1} = \mathfrak{L}_i(w_G^{t-1})$ denotes the loss function of the global model at time t computed based on the local data of node N_i. By assuming that $\mathfrak{L}_j$ is a β smooth function, it follows that:

$$\begin{aligned} &\exists \beta > 0, \text{ such that} \\ &\mathfrak{L}_j\left(w_i^t\right) - \mathfrak{L}_j\left(w_G^{t-1}\right) \leq \frac{\beta}{2}\left\|w_i^t - w_G^{t-1}\right\|^2 + \\ &\left\langle \nabla \mathfrak{L}_j\left(w_G^{t-1}\right), w_i^t - w_G^{t-1}\right\rangle. \end{aligned} \tag{50}$$

By combining (49) and (50), it follows that:

$$\begin{aligned} \mathfrak{L}_j\left(w_i^t\right) - \mathfrak{L}_j\left(w_G^{t-1}\right) &\leq \frac{\beta}{2}\mu^2\left\|\nabla_i^{t-1}\right\|^2 - \mu\left\|\nabla_i^{t-1}\right\|\left\|\nabla_j^{t-1}\right\| \\ &\leq \mu\left\|\nabla_i^{t-1}\right\|\left(\frac{\beta}{2}\mu\left\|\nabla_i^{t-1}\right\| - \left\|\nabla_j^{t-1}\right\|\right). \end{aligned} \tag{51}$$

Consequently, the upperbound of the loss function difference becomes negative if the stepsize is chosen to satisfy the following condition:

$$\mu \leq \frac{2\left\|\nabla_j^{t-1}\right\|}{\beta\left\|\nabla_i^{t-1}\right\|}. \tag{52}$$

Having $\left\|\nabla_j^{t-1}\right\|$ equal to zero is equivalent to having ∇_j^{t-1} equal to 0, which means that the FL model converges. However, as proven in Theorem 1, this cannot happen because an FL model cannot converge in the presence of a Byzantine node that is not yet isolated. Therefore, it follows that:

$$\exists\ \gamma_{m_j}^G > 0 \text{ such that } \left\|\nabla_j^{t-1}\right\| \geq \gamma_{m_j}^G\ \forall t \geq 1. \tag{53}$$

By using Assumption 3, the gradient is bounded by γ_G and using (53), it follows that:

$$\exists \gamma_{m_j}^G, \gamma_G > 0 \text{ such that } \frac{2\left\|\nabla_j^{t-1}\right\|}{\beta\left\|\nabla_i^{t-1}\right\|} \geq \frac{2\gamma_{m_j}^G}{\beta\gamma_G}. \tag{54}$$

Consequently, one possible stepsize that guarantees a strictly negative loss function difference for a data owner N_i is given be

$$\mu_i = \frac{\gamma_{m_i}^G}{\beta\gamma_G}. \tag{55}$$

To make this property correct for all the nodes, the stepsize can be set as

$$\hat{\mu} = \frac{1}{\beta\gamma_G}\min_{i\in\mathfrak{S}_L}\left(\gamma_{m_i}^G\right). \tag{56}$$

Consequently, when the stepsize is equal or smaller than $\hat{\mu}$, the computed trustworthiness of a legitimate node N_j based on the local data of any node N_i is always positive, i.e.,

$$\mathfrak{T}_{i,j} \geq 0, \quad \text{if } j \in \mathfrak{S}_L. \tag{57}$$

When the trustworthiness is computed for a Byzantine node N_i by another node N_j, i.e., $j \in \mathfrak{S}_M$, it follows by definition that:

$$\mathfrak{L}_j\left(w_i^t\right) \geq \mathfrak{L}_j\left(w_G^{t-1}\right). \tag{58}$$

In fact, a Byzantine node is by definition a "lazy" or malicious node that sends random or falsified model weights to the server. Therefore, it is extremely rare for a Byzantine node to reduce the loss since its objective is exactly the opposite. Even if such rare event occurred and a randomly generated Byzantine model reduced the loss, it can be considered as beneficial to the global model training at least for that particular iteration. Consequently

$$\mathfrak{T}_{i,j} \leq 0, \quad \text{if } j \in \mathfrak{S}_M. \tag{59}$$

This concludes the proof of Theorem 4.

APPENDIX E

When no Byzantine nodes are present in the network or when the malicious nodes are already identified and isolated, the model can eventually converge. Therefore, there is no guarantee to have a lower bound on the gradient norm. Consequently, (53) is no longer valid. At a particular time t, if the global model w_G^t corresponds to a zero loss according to the data in node N_j ($\|\nabla_j^t\| = 0$), there exists no positive stepsize μ that provides a negative loss evolution unless the loss is also equal to zero at node N_i ($\|\nabla_i^t\| = 0$). To solve this issue, the loss evolution is investigated before and after convergence separately.

Let ϵ denote a small convergence gradient norm threshold. When $\|\nabla_j^t\| \geq \epsilon$, the loss upperbound in (51) becomes

$$\mathfrak{L}_j\left(w_i^t\right) - \mathfrak{L}_j\left(w_G^{t-1}\right) \leq \mu\left\|\nabla_i^{t-1}\right\|\left(\frac{\beta}{2}\mu\gamma_G - \epsilon\right). \tag{60}$$

Consequently, there exists a stepsize that guarantees a negative loss evolution (positive trustworthiness) defined as

$$\mu_i = \frac{\epsilon}{\beta\gamma_G}. \tag{61}$$

When $\|\nabla_j^t\| \leq \epsilon$, i.e., the network converged according to the data of node N_j, the model should also be at least close to the convergence point for the other nodes, i.e.,

$$\exists \epsilon_C > 0 \quad \text{such that if } \left\|\nabla_j^t\right\| \leq \epsilon, \text{ Then, } \left\|\nabla_i^t\right\| \leq \epsilon_C\ \ \forall i \in \mathfrak{S}_L.$$

In practice, γ_C denotes the highest gradient norm a node can get when another data owner reaches convergence. Note that if the data sizes and properties are similar, ϵ_C should be very small. Therefore, the loss upperbound in (51) becomes

$$\mathfrak{L}_j\left(w_i^t\right) - \mathfrak{L}_j\left(w_G^{t-1}\right) \leq \frac{\beta}{2}\mu^2\epsilon_C^2. \tag{62}$$

In this case, there is no guarantee that the computed trustworthiness is positive for the legitimate users. However, there is a guarantee that it is bigger than a negative number with a very small magnitude. This means that there is a small chance of getting false alarms when the model is close to convergence.

However, if the model already converged, it would either consider the obtained model or remain in the convergence state since the adopted mechanism should theoretically not have misdetection for an adequate stepsize choice.

ACKNOWLEDGMENT

Open Access funding provided by the Qatar National Library.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Stat.*, Apr. 2017, pp. 1273–1282.

[2] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surveys*, vol. 54, no. 6, pp. 1–36, 2021.

[3] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, Mar. 2021, Art. no. 106775.

[4] Z. Jiehan et al., "A survey on federated learning and its applications for accelerating Industrial Internet of Things," 2021, *arXiv:2104.10501*.

[5] S. Prathiba, G. Raja, S. Anbalagan, S. Gurumoorthy, N. Kumar, and M. Guizani, "Cybertwin-driven federated learning based personalized service provision for 6G-V2X," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 4632–4641, May 2022.

[6] K. Abualsaud, A. Mohamed, T. Khattab, E. Yaacoub, M. Hasna, and M. Guizani, "Classification for imperfect EEG epileptic seizure in IoT applications: A comparative study," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, 2018, pp. 364–369.

[7] A. B. Said, A. Mohamed, T. Elfouly, K. Abualsaud, and K. Harras, "Deep learning and low rank dictionary model for mHealth data classification," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, 2018, pp. 358–363.

[8] A. Gouissem, K. Abualsaud, E. Yaacoub, T. Khattab, and M. Guizani, "Game theory for anti-jamming strategy in multichannel slow fading IoT networks," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 16880–16893, Dec. 2021.

[9] A. Gouissem, K. Abualsaud, E. Yaacoub, T. Khattab, and M. Guizani, "IoT anti-jamming strategy using game theory and neural network," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, 2020, pp. 770–776.

[10] E. M. El Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3521–3530.

[11] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. 36th Int. Conf. Mach. Learn.*, Jun. 2019, pp. 634–643.

[12] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proc. IEEE Symp. Security Privacy (SP)*, 2018, pp. 19–35.

[13] A. Portnoy, Y. Tirosh, and D. Hendler, "Towards realistic Byzantine-robust federated learning," 2020, *arXiv:2004.04986*.

[14] Q. Li, B. Kailkhura, R. Goldhahn, P. Ray, and P. K. Varshney, "Robust federated learning using ADMM in the presence of data falsifying Byzantines," 2017, *arXiv:1710.05241*.

[15] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, pp. 1–25, 2017.

[16] P. Blanchard, E. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 118–128.

[17] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" 2019, *arXiv:1911.07963*.

[18] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 5650–5659.

[19] J. Schneible and A. Lu, "Anomaly detection on the edge," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, 2017, pp. 678–682.

[20] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," 2020, *arXiv:2012.13995*.

[21] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2021.

[22] A. A. Abdellatif et al., "Communication-efficient hierarchical federated learning for IoT heterogeneous systems with imbalanced data," *Future Gener. Comput. Syst.*, vol. 128, pp. 406–419, Mar. 2022.

[23] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018, *arXiv:1806.00582*.

[24] A. Khaled, K. Mishchenko, and P. Richtárik, "First analysis of local GD on heterogeneous data," 2019, *arXiv:1909.04715*.

[25] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[26] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for SVM," *Math. Program.*, vol. 127, no. 1, pp. 3–30, 2011.

[27] A. L. Goldberger et al., "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.