Byzantine-Resilient Secure Federated Learning

Jinhyun So, Başak Güler, A. Salman Avestimehr

Abstract—Secure federated learning is a privacy-preserving framework to improve machine learning models by training over large volumes of data collected by mobile users. This is achieved through an iterative process where, at each iteration, users update a global model using their local datasets. Each user then masks its local update via random keys, and the masked models are aggregated at a central server to compute the global model for the next iteration. As the local updates are protected by random masks, the server cannot observe their true values. This presents a major challenge for the resilience of the model against adversarial (Byzantine) users, who can manipulate the global model by modifying their local updates or datasets. Towards addressing this challenge, this paper presents the first single-server Byzantine-resilient secure aggregation framework (BREA) for secure federated learning. BREA is based on an integrated stochastic quantization, verifiable outlier detection, and secure model aggregation approach to guarantee Byzantine-resilience, privacy, and convergence simultaneously. We provide theoretical convergence and privacy guarantees and characterize the fundamental trade-offs in terms of the network size, user dropouts, and privacy protection. Our experiments demonstrate convergence in the presence of Byzantine users, and comparable accuracy to conventional federated learning benchmarks.

Index Terms—Federated learning, privacy-preserving machine learning, Byzantine-resilience, distributed training in mobile networks.

I. INTRODUCTION

Federated learning is a distributed training framework that has received significant interest in the recent years, by allowing machine learning models to be trained over the vast amount of data collected by mobile devices [1], [2]. In this framework, training is coordinated by a central server who maintains a global model, which is updated by the mobile users through an iterative process. At each iteration, the server sends the current version of the global model to the mobile devices, who update it using their local data and create a local update. The server then aggregates the local updates of the users and updates the global model for the next iteration [1]–[8].

Security and privacy considerations of distributed learning are mainly focused around two seemingly separate directions: 1) ensuring robustness of the global model against adversarial manipulations and 2) protecting the privacy of individual users. The first direction aims at ensuring that the trained model is robust against Byzantine faults that may occur in the training data or during protocol execution. These faults may result either from an adversarial user who can manipulate the training data or the information exchanged during the protocol, or

Jinhyun So is with the Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA, 90089 USA (e-mail: jinhyuns@usc.edu). Başak Güler is with the Department of Electrical and Computer Engineering, University of California, Riverside, CA, 92521 USA (email: bguler@ece.ucr.edu). A. Salman Avestimehr is with the Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA, 90089 USA (e-mail: avestimehr@ee.usc.edu).

due to device malfunctioning. Notably, it has been shown that even a single Byzantine fault can significantly alter the trained model [9]. The primary approach for defending against Byzantine faults is by comparing the local updates received from different users and removing the outliers at the server [9]–[13]. Doing so, however, requires the server to learn the true values of the local updates of each individual user. The second direction aims at protecting the privacy of the individual users, by keeping each local update private from the server and the other users participating in the protocol [2]–[7]. This is achieved through what is known as a *secure aggregation* protocol [2]. In this protocol, each user masks its local update through additive secret sharing using private and pairwise random keys before sending it to the server. Once the masked models are aggregated at the server, the additional randomness cancels out and the server learns the aggregate of all user models. At the end of the protocol, the server learns no information about the individual models beyond the aggregated model, as they are masked by the random keys unknown to the server. In contrast, conventional distributed training frameworks that perform gradient aggregation and model updates using the true values of the gradients may reveal extensive information about the local datasets of the users, as shown in [14]–[16].

This presents a major challenge in developing a Byzantine-resilient, and at the same time, privacy-preserving federated learning framework. On the one hand, robustness against Byzantine faults requires the server to obtain the individual model updates in the clear, to be able to compare the updates from different users with each other and remove the outliers. On the other hand, protecting user privacy requires each individual model to be masked with random keys, as a result, the server only observes the masked model, which appears as a uniformly random vector that could correspond to any point in the parameter space. Our goal is to reconcile these two critical directions. In particular, we want to address the following question, *"How can one make federated learning protocols robust against Byzantine adversaries while preserving the privacy of individual users?"*.

In this paper, we propose the first single-server Byzantine-resilient secure aggregation framework, BREA, towards addressing this problem. Our framework is built on the following main principles. Given a network of N mobile users with up to A adversaries, each user initially secret shares its local update with the other users, through a verifiable secret sharing protocol [17]. However, doing so requires the local updates to be masked by uniformly random vectors in a finite field [18], whereas the model updates during training are performed in the domain of real numbers. In order to handle this problem, BREA utilizes stochastic quantization to transfer the local updates from the real domain into a finite field.

Verifiable secret sharing allows the users to perform consistency checks to validate the secret shares and ensure that every user follows the protocol. However, a malicious user can still manipulate the global model by modifying its local update or private dataset. BREA handles such attacks through a robust gradient descent approach, enabled by secure computations over the secret shares of the local updates. To do so, each user locally computes the pairwise distances between the secret shares of the local updates belonging to other users, and sends the computation results to the server. Since these computations are carried out using the secret shares, users do not learn the true values of the local updates belonging to other users.

In the final phase, the server collects the computation results from a sufficient number of users, recovers the pairwise distances between the local updates, and performs user selection for model aggregation. The user selection protocol is based on a distance-based outlier removal mechanism [9], to remove the effect of potential adversaries and to ensure that the selected models are sufficiently close to an unbiased gradient estimator. After the user selection phase, the secret shares of the models belonging to the selected users are aggregated locally by the mobile users. The server then gathers the secure computation results from the users, reconstructs the true value of the aggregate of the selected user models, and updates the global model. Our framework guarantees the privacy of individual user models, in particular, the server learns no information about the local updates, beyond their aggregated value and the pairwise distances.

In our theoretical analysis, we demonstrate provable convergence guarantees for the model and robustness guarantees against Byzantine adversaries. We then identify the theoretical performance limits in terms of the fundamental trade-offs between the network size, user dropouts, number of adversaries, and privacy protection. Our results demonstrate that, in a network with N mobile users, BREA can theoretically guarantee: i) robustness of the trained model against up to A Byzantine adversaries, ii) tolerance against up to D user dropouts, iii) privacy of each local update, against the server and up to T colluding users, as long as $N \geq 2A + 1 + \max\{m + 2, D + 2T\}$, where m is the number of selected models for aggregation.

We then numerically evaluate the performance of BREA and compare it to the conventional federated learning protocol, the federated averaging scheme of [1]. To do so, we implement BREA in a distributed network of $N = 40$ users with up to $A = 12$ Byzantine users who can send arbitrary vectors to the server or to the honest users. We demonstrate that BREA guarantees convergence against Byzantine users and its convergence rate is comparable to the convergence rate of federated averaging. BREA also has comparable test accuracy to the federated averaging scheme while BREA entails quantization loss to preserve the privacy of individual users.

Finally, while BREA provides the first distributed training framework that is both Byzantine-resilient and privacy-preserving, there are several directions for further improvements. The first is that our theoretical analysis for convergence needs the assumption of an independent and identically distributed (i.i.d.) data distribution over the users. Second, we rely on distance-based outlier removal mechanisms, which make it difficult to distinguish whether a large distance between local updates is due to a non-i.i.d. data distribution or a Byzantine attack. Providing theoretical performance guarantees for non-i.i.d. data in the Byzantine-robust training setups is an important and interesting direction for future research. However, one should also note that even without the privacy requirements, this is still an open problem and an active area of research in the literature [19], [20]. Other open questions include whether it is possible to break the quadratic communication and computation complexity of Byzantine-resilient secure aggregation in the current work, and investigating the fundamental performance limits of Byzantine-resilient distributed learning with multiple local updates.

II. RELATED WORK

In the non-Byzantine federated learning setting, secure aggregation is performed through a procedure known as additive masking [2], [21]. In this setup, users first agree on pairwise secret keys using a Diffie-Hellman type key exchange protocol [22], and send a masked version of their local update to the server, where the masking is done using pairwise and private secret keys. When the masked models are aggregated at the server, additive masks cancel out, allowing the server to learn the aggregate of the local updates. This process works well if no users drop during the execution of the protocol. In wireless environments, however, users may drop from the protocol anytime due to the variations in channel conditions. Such user dropouts are handled by letting each user secret share their private and pairwise keys through Shamir's secret sharing [18]. The server can remove the additive masks by collecting the secret shares from the surviving users. This approach leads to a quadratic communication overhead in the number of users. More recent approaches have focused on reducing the communication overhead, by training in a smaller parameter space [23], autotuning the parameters [24], or by utilizing coding techniques [6].

Another line of work has focused on differentially-private federated learning approaches [25], [26], to protect the privacy of personally-identifiable information against inference attacks. Although our focus is not on differential-privacy, our approach may in principle be combined with differential privacy techniques [27], which is an interesting future direction. Another important direction in federated learning is the study of fairness and how to avoid biasing the model towards specific users [28], [29]. The convergence properties of federated learning models are investigated in [30].

Distributed training protocols have been extensively studied in the Byzantine setting using clear (unmasked) model updates [9]–[13], [19], [20]. The main defense mechanism to protect the trained model against Byzantine users is by comparing the model updates received from different users, and removing the outliers. Doing so ensures that the selected model updates are close to each other, as long as the network has a sufficiently large number of honest users. A related line of work is model poisoning attacks, which are studied in [31], [32].

In concurrent work, a Byzantine-robust secure gradient descent algorithm has been proposed for a two-server model
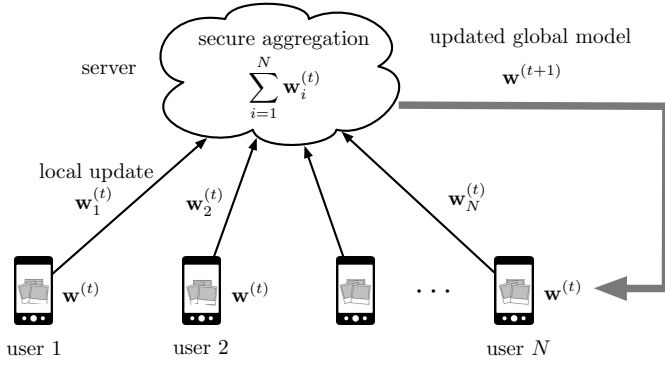
Fig. 1. Secure aggregation in federated learning. At iteration t, the server sends the current state of the global model, denoted by $\mathbf{w}^{(t)}$, to the mobile users. User $i \in [N]$ forms a local update $\mathbf{w}_i^{(t)}$ by updating the global model using its local dataset. The local updates are aggregated in a privacy-preserving protocol at the server, who then updates the global model, and sends the new model, $\mathbf{w}^{(t+1)}$, to the mobile users.

in [33], however, unlike federated learning (which is based on a single-server architecture) [1], [2], this work requires two honest (non-colluding) servers who both interact with the mobile users and communicate with each other to carry out a secure two-party protocol, but do not share any sensitive information with each other in an attempt to breach user privacy. In contrast, our goal is to develop a single-server Byzantine-resilient secure training framework, to facilitate robust and privacy-preserving training architectures for federated learning. Compared to the two-server models, single server models carry the additional challenge where all information has to be collected at a single server, while still being able to keep the individual models of the users private.

The remainder of the paper is organized as follows. In Section III, we provide background on federated learning. Section IV presents our system model along with the key parameters used to evaluate the system performance. Section V introduces our framework and the details of the specific components. Section VI presents our theoretical results, whereas our numerical evaluations are provided in Section VII, to demonstrate the convergence and Byzantine-resilience. The paper is concluded in Section VIII. The following notation is used throughout the paper. We represent a scalar variable with x, whereas $\mathbf{x}$ represents a vector. A set is denoted by $\mathcal{X}$, whereas $[N]$ refers to the set $\{1, \ldots, N\}$.

III. BACKGROUND

Federated learning is a distributed training framework for machine learning in mobile networks while preserving the privacy of users. Training is coordinated by a central server who maintains a global model $\mathbf{w} \in \mathbb{R}^d$ with dimension d. The goal is to train the global model using the data held at mobile devices, by minimizing a global objective function $C(\mathbf{w})$ as,

$$\min_{\mathbf{w}} C(\mathbf{w}). \tag{1}$$

The global model is updated locally by mobile users on sensitive private datasets, by letting

$$C(\mathbf{w}) = \sum_{i=1}^{N} \frac{B_i}{B} C_i(\mathbf{w}) \tag{2}$$

where N is the total number of mobile users, $C_i(\mathbf{w})$ denotes the local objective function of user i, B_i is the number of data points in user i's private dataset $\mathcal{D}_i$, and $B := \sum_{i=1}^{N} B_i$. For simplicity, we assume that users have equal-sized datasets, i.e., $B_i = \frac{B}{N}$ for all $i \in [N]$.

Training is performed through an iterative process where mobile users interact through the central server to update the global model. At each iteration, the server shares the current state of the global model, denoted by $\mathbf{w}^{(t)}$, with the mobile users. Each user i creates a local update,

$$\mathbf{w}_i^{(t)} = g(\mathbf{w}^{(t)}, \xi_i^{(t)}) \tag{3}$$

where g is an estimate of the gradient $\nabla C(\mathbf{w}^{(t)})$ of the cost function C and $\xi_i^{(t)}$ is a random variable representing the random sample (or a mini-batch of samples) drawn from $\mathcal{D}_i$. We assume that the private datasets $\{\mathcal{D}_i\}_{i \in [N]}$ have the same distribution and $\{\xi_i^{(t)}\}_{i \in [N]}$ are i.i.d. $\xi_i^{(t)} \sim \xi$ where ξ is a uniform random variable such that each $\mathbf{w}_i^{(t)}$ is an unbiased estimator of the true gradient $\nabla C(\mathbf{w}^{(t)})$, i.e.,

$$\mathbb{E}_\xi[g(\mathbf{w}^{(t)}, \xi_i^{(t)})] = \nabla C(\mathbf{w}^{(t)}). \tag{4}$$

The local updates are aggregated at the server in a privacy-preserving protocol, such that the server only learns the aggregate of a large fraction of the local updates, ideally the sum of all user models $\sum_{i \in [N]} \mathbf{w}_i^{(t)}$, but no further information is revealed about the individual models beyond their aggregated value. Using the aggregate of the local updates, the server updates the global model for the next iteration,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \gamma^{(t)} \sum_{i \in [N]} \mathbf{w}_i^{(t)} \tag{5}$$

where $\gamma^{(t)}$ is the learning rate, and sends the updated model $\mathbf{w}^{(t+1)}$ to the users. This process is illustrated in Figure 1.

Conventional secure aggregation protocols require each user to mask its local update using random keys before aggregation [2], [6], [34]. This is typically done by creating pairwise keys between the users through a key exchange protocol [22]. Using the pairwise keys, each pair of users $i, j \in [N]$ agree on a pairwise random seed $a_{ij}^{(t)}$. User i also creates a private random seed $b_i^{(t)}$, which protects the privacy of the local update in case the user is delayed instead of being dropped, in which case the pairwise keys are not sufficient for privacy, as shown in [2]. User $i \in [N]$ then sends a masked version of its local update $\mathbf{w}_i^{(t)}$, given by

$$\mathbf{y}_i^{(t)} := \mathbf{w}_i^{(t)} + \text{PRG}(b_i^{(t)}) + \sum_{j:i<j} \text{PRG}(a_{ij}^{(t)}) - \sum_{j:i>j} \text{PRG}(a_{ji}^{(t)}) \tag{6}$$

to the server, where PRG is a pseudo random generator. User i then secret shares $b_i^{(t)}$ and $\{a_{ij}^{(t)}\}_{j \in [N]}$ with the other users, via Shamir's secret sharing [18]. For computing the aggregate of the user models, the server collects either the secret shares of the pairwise seeds belonging to a dropped user, or the shares of the private seed belonging to a surviving user (but not both). The server then recovers the private seeds of the surviving users and the pairwise seeds of the dropped users, and removes

them from the aggregate of the masked models,

$$
\begin{aligned}
\mathbf{y}^{(t)} &= \sum_{i \in \mathcal{U}} \left(\mathbf{y}_i^{(t)} - \mathrm{PRG}(b_i^{(t)}) \right) \\
&\quad - \sum_{i \in \mathcal{D}} \left(\sum_{j:i<j} \mathrm{PRG}(a_{ij}^{(t)}) - \sum_{j:i>j} \mathrm{PRG}(a_{ji}^{(t)}) \right) \\
&= \sum_{i \in \mathcal{U}} \mathbf{w}_i^{(t)}
\end{aligned}
\tag{7}
$$

and obtains the aggregate of the local updates, where $\mathcal{U} \subseteq [N]$ and $\mathcal{D} \subseteq [N]$ denote the set of surviving and dropped users, respectively. In (7), $\sum_{i \in \mathcal{U}} \mathrm{PRG}(b_i^{(t)})$ corresponds to the reconstructed private seeds belonging to the surviving users. On the other hand, $\sum_{i \in \mathcal{D}} \left(\sum_{j:i<j} \mathrm{PRG}(a_{ij}^{(t)}) - \sum_{j:i>j} \mathrm{PRG}(a_{ji}^{(t)}) \right)$ corresponds to the reconstructed pairwise seeds belonging to the dropped users. Both of these terms are reconstructed by the server to remove the random masks in the aggregate of the masked versions of the surviving users, $\sum_{i \in \mathcal{U}} \mathbf{y}_i^{(t)}$. At the end, all of the random masks cancel out, and the server recovers the summation of the original models belonging to all of the surviving users, i.e., $\sum_{i \in \mathcal{U}} \mathbf{w}_i^{(t)}$.

IV. PROBLEM FORMULATION

In this section, we describe the Byzantine-resilient secure aggregation problem, by extending the conventional secure aggregation scenario from Section III to the case when some users, known as Byzantine adversaries, can manipulate the trained model by modifying their local datasets or by sharing false information during the protocol.

We consider a distributed network with N mobile users and a single server. User $i \in [N]$ holds a local update[1] $\mathbf{w}_i$ of dimension d. The goal is to aggregate the local updates at the server, while protecting the privacy of individual users. However, unlike the non-Byzantine setting of Section III, the aggregation operation in the Byzantine setting should be robust against potentially malicious users. To this end, we represent the aggregation operation by a function,

$$
f(\mathbf{w}_1, \ldots, \mathbf{w}_N) = \sum_{i \in \mathcal{S}} \mathbf{w}_i
\tag{8}
$$

where $\mathcal{S}$ is a set of users selected by the server for aggregation. The role of $\mathcal{S}$ is to remove the effect of potentially Byzantine adversaries on the trained model, by removing the outliers. Similar to prior works on federated learning, our focus is on computationally-bounded parties, whose strategies can be described by a probabilistic polynomial time algorithm [2].

We evaluate the performance of a Byzantine-resilient secure aggregation protocol according to the following key parameters:

- *Robustness against Byzantine users:* We assume that up to A users are Byzantine (malicious), who manipulate the protocol by modifying their local datasets or by sharing false information during protocol execution. The protocol should be robust against such Byzantine adversaries.
- *Privacy of local updates:* The aggregation protocol should protect the privacy of any individual user from the server

and any collusions between up to T users. Specifically, the local update of any user should not be revealed to the server or the remaining users, even if up to T users cooperate with each other by sharing information.[2]
- *Tolerance to user dropouts:* Due to potentially poor wireless channel conditions, we assume that up to D users may get dropped or delayed at any time during protocol execution. The protocol should be able to tolerate such dropouts, i.e., the privacy and convergence guarantees should hold even if up to D users drop or get delayed.

In this paper, we present a single-server Byzantine-resilient secure aggregation framework (BREA) for the computation of (8). BREA consists of the following key components:

1) *Stochastic quantization:* Users initially quantize their local updates from the real domain to the domain of integers, and then embed them in a field $\mathbb{F}_p$ of integers modulo a prime p. To do so, our framework utilizes stochastic quantization, which is instrumental in our theoretical convergence guarantees.
2) *Verifiable secret sharing of the user models:* Users then secret share their quantized models using a verifiable secret sharing protocol. This ensures that the secret shares created by the mobile users are valid, i.e., Byzantine users cannot cheat by sending invalid secret shares.
3) *Secure distance computation:* In this phase, users compute the pairwise distances between the secret shares of the local updates, and send the results to the server. Since this computation is performed using the secret shares of the models instead of their true values, users do not learn any information about the actual model parameters.
4) *User selection at the server:* Upon receiving the computation results from the users, the server recovers the pairwise distances between the local updates and selects the set of users whose models will be included in the aggregation, by removing the outliers. This ensures that the aggregated model is robust against potential manipulations from Byzantine users. The server then announces the list of the selected users.
5) *Secure model aggregation:* In the final phase, each user locally aggregates the secret shares of the models selected by the server, and sends the computation result to the server. Using the computation results, the server recovers the aggregate of the models of the selected users, and updates the model.

In the following, we describe the details of each phase.

V. THE BREA FRAMEWORK

In this section, we present the details of the BREA framework for Byzantine-resilient secure federated learning.

A. Stochastic Quantization

In BREA, the operations for verifiable secret sharing and secure distance computations are carried out over a finite field $\mathbb{F}_p$ for some large prime p. To this end, user $i \in [N]$

---

[1] For notational clarity, throughout Sections IV and V, we omit the iteration number (t) from $\mathbf{w}_i^{(t)}$.

[2] Collusions that may occur between the server and the users are beyond the scope of our paper.

initially quantizes its local update $\mathbf{w}_i$ from the domain of real numbers to the finite field. We assume that the field size p is large enough to avoid any wrap-around during secure distance computation and secure model aggregation, which will be described in Sections V-C and V-E, respectively.

Quantization requires a challenging task as it should be performed in a way to ensure the convergence of the model. Moreover, the quantization function should allow the representation of negative integers in the finite field, and facilitate computations to be performed in the quantized domain. Therefore, we cannot utilize well-known gradient quantization techniques such as in [35], which represents the sign of a negative number separately from its magnitude. BREA addresses this challenge with a simple stochastic quantization strategy as follows. For any integer $q \geq 1$, we define a stochastic rounding function:

$$Q_q(x) = \begin{cases} \frac{\lfloor qx \rfloor}{q} & \text{with prob. } 1 - (qx - \lfloor qx \rfloor) \\ \frac{\lfloor qx \rfloor + 1}{q} & \text{with prob. } qx - \lfloor qx \rfloor \end{cases} \quad (9)$$

where $\lfloor x \rfloor$ is the largest integer less than or equal to x, and note that this function is unbiased, i.e., $\mathbb{E}_Q[Q_q(x)] = x$. Parameter q is a tuning parameter that corresponds to the number of quantization levels. Variance of $Q_q(x)$ decreases as the value of q increases, which will be detailed in Lemma 1 in Section VI. We then define the quantized model,

$$\overline{\mathbf{w}}_i := \phi(q \cdot Q_q(\mathbf{w}_i)) \quad (10)$$

where the function Q_q from (9) is carried out element-wise, and a mapping function $\phi : \mathbb{R} \to \mathbb{F}_p$ is defined to represent a negative integer in the finite field by using two's complement representation,

$$\phi(x) = \begin{cases} x & \text{if } x \geq 0 \\ p + x & \text{if } x < 0. \end{cases} \quad (11)$$

B. Verifiable Secret Sharing of the User Models

BREA protects the privacy of individual user models through verifiable secret sharing. This is to ensure that the individual user models are kept private while preventing the Byzantine users from breaking the integrity of the protocol by sending invalid secret shares to the other users.

To do so, user $i \in [N]$ secret shares its quantized model $\overline{\mathbf{w}}_i$ with the other users through a non-interactive verifiable secret sharing protocol [17]. Our framework leverages Feldman's verifiable secret sharing protocol from [17], which combines Shamir's secret sharing [18] with homomorphic encryption. In this setup, each party creates the secret shares, then broadcasts commitments to the coefficients of the polynomial they use for Shamir's secret sharing, so that other parties can verify that the secret shares are constructed correctly. To verify the secret shares from the given commitments, the protocol leverages the homomorphic property of exponentiation, i.e., $exp(a + b) = exp(a)exp(b)$, whereas the privacy protection is based on the assumption that computation of the discrete logarithm in the finite field is intractable.

The individual steps carried out for verifiable secret sharing in our framework are as follows. Initially, the server and users agree on N distinct elements $\{\theta_i\}_{i \in [N]}$ from $\mathbb{F}_p$. This can be done offline by using a conventional majority-based consensus protocol [36], [37]. User $i \in [N]$ then generates secret shares of the quantized model $\overline{\mathbf{w}}_i$ by forming a random polynomial $f_i : \mathbb{F}_p \to \mathbb{F}_p^d$ of degree T,

$$f_i(\theta) = \overline{\mathbf{w}}_i + \sum_{j=1}^{T} \mathbf{r}_{ij} \theta^j \quad (12)$$

in which the vectors $\mathbf{r}_{ij}$ are generated uniformly at random from $\mathbb{F}_p^d$ by user i. User i then sends a secret share of $\overline{\mathbf{w}}_i$ to user j, denoted by,

$$\mathbf{s}_{ij} = f_i(\theta_j). \quad (13)$$

To make these shares verifiable, user i also broadcasts commitments to the coefficients of f_i, given by

$$\mathbf{c}_{ij} = \begin{cases} \psi^{\overline{\mathbf{w}}_i} & \text{for } j = 0 \\ \psi^{\mathbf{r}_{ij}} & \text{for } j = 1, \dots, T. \end{cases} \quad (14)$$

where ψ denotes a generator of $\mathbb{F}_p$, and all arithmetic is taken modulo λ for some large prime λ such that p divides $\lambda - 1$.

Upon receiving the commitments in (14), each user $j \in [N]$ can verify the secret share $\mathbf{s}_{ij} = f_i(\theta_j)$ by checking

$$\psi^{\mathbf{s}_{ij}} = \prod_{k=0}^{T} \mathbf{c}_{ik}^{\theta_j^k} \quad (15)$$

where all arithmetic is taken modulo λ. This commitment scheme ensures that the secret shares are created correctly from the polynomial in (12), hence they are valid. On the other hand, as we assume the intractability of computing the discrete logarithm [17], the server or the users cannot compute the discrete logarithm $\log_\psi(\mathbf{c}_{it})$ and reveal the quantized model $\overline{\mathbf{w}}_i$ from $\mathbf{c}_{i0}$ in (14).

C. Secure Distance Computation

Verifiable secret sharing of the model parameters, as described in Section V-B, ensures that the users follow the protocol correctly by creating valid secret shares. However, malicious users can still try to manipulate the trained model by modifying their local updates instead. In this case, the secret shares will be created correctly but according to a false model. In order to ensure that the trained model is robust against such adversarial manipulations, BREA leverages a distance-based outlier detection mechanism, such as in [9], [38]. The main principle behind these mechanisms is to compute the pairwise distances between the local updates and select a set of models that are sufficiently close to each other. On the other hand, the outlier detection mechanism in BREA has to protect the privacy of local updates, and performing the distance computations on the true values of the model parameters would breach the privacy of individual users.

We address this by a privacy-preserving distance computation approach, in which the pairwise distances are computed locally by each user, using the secret shares of the model parameters received from the other users. In particular, upon receiving the secret shares of the model parameters as described in Section V-B, user i computes the pairwise distances,

$$d_{jk}^{(i)} := \|\mathbf{s}_{ji} - \mathbf{s}_{ki}\|^2 \quad (16)$$

between each pair of users $j, k \in [N]$, and sends the result to the server. Since the computations in (16) are performed over the secret shares, user i learns no information about the true values of the model parameters $\overline{\mathbf{w}}_j$ and $\overline{\mathbf{w}}_k$ of users j and k, respectively. Finally, we note that the computation results from (16) are scalar values.

D. User Selection at the Server

Upon receiving the computation results in (16) from a sufficient number of users, the server reconstructs the true values of the pairwise distances. During this phase, Byzantine users may send incorrect computation results to the server, hence the reconstruction process should be able to correct the potential errors that may occur in the computation results due to malicious users. Our decoding procedure is based on the decoding of Reed-Solomon codes.

The intuition of the decoding process is that the computations from (16) correspond to evaluation points of a univariate polynomial $h_{jk} : \mathbb{F}_p \to \mathbb{F}_p$ of degree at most $2T$, where

$$h_{jk}(\theta) := \| f_j(\theta) - f_k(\theta) \|^2 \tag{17}$$

for $\theta \in \{\theta_i\}_{i \in [N]}$ and $j, k \in [N]$. Accordingly, h_{jk} can be viewed as the encoding polynomial of a Reed-Solomon code with degree at most $2T$, such that the missing computations due to the dropped users correspond to the *erasures* in the code, and manipulated computations from Byzantine users refer to the *errors* in the code. Therefore, the decoding process of the server corresponds to decoding an $[N, 2T+1, N-2T]_p$ Reed-Solomon code with at most D erasures and at most A errors. By utilizing well-known Reed-Solomon decoding algorithms [39], the server can recover the polynomial h_{jk} and obtain the true value of the pairwise distances by using the relation $h_{jk}(0) = \| f_j(0) - f_k(0) \|^2 = \| \overline{\mathbf{w}}_j - \overline{\mathbf{w}}_k \|^2$. At the end, the server learns the pairwise distances

$$\overline{d}_{jk} := \| \overline{\mathbf{w}}_j - \overline{\mathbf{w}}_k \|^2 \tag{18}$$

between the models of users $j, k \in [N]$. Then the server converts (18) from the finite field to the real domain as follows,

$$d_{jk} = \frac{\phi^{-1}(\overline{d}_{jk})}{q^2} \tag{19}$$

for $j, k \in [N]$, where q is the integer parameter in (9) and the demapping function $\phi^{-1} : \mathbb{F}_p \to \mathbb{R}$ is defined as

$$\phi^{-1}(\overline{x}) = \begin{cases} \overline{x} & \text{if} \quad 0 \le \overline{x} < \frac{p-1}{2} \\ \overline{x} - p & \text{if} \quad \frac{p-1}{2} \le \overline{x} < p \end{cases} . \tag{20}$$

We assume the field size p is large enough to ensure the correct recovery of the pairwise distances,

$$d_{jk} = \frac{\phi^{-1}\big(\| \phi(q \cdot Q_q(\mathbf{w}_j)) - \phi(q \cdot Q_q(\mathbf{w}_k)) \|^2\big)}{q^2}$$

$$= \frac{\phi^{-1}\big(\phi(q^2 \| Q_q(\mathbf{w}_j) - Q_q(\mathbf{w}_k) \|^2)\big)}{q^2} \tag{21}$$

$$= \| Q_q(\mathbf{w}_j) - Q_q(\mathbf{w}_k) \|^2 \tag{22}$$

where Q_q is the stochastic rounding function defined in (9)

and (21) holds if

$$q^2 \| Q_q(\mathbf{w}_j) - Q_q(\mathbf{w}_k) \|^2 < (p-1)/2. \tag{23}$$

By utilizing the pairwise distances in (22), the server carries out a distance-based outlier removal algorithm to select the set of users to include in the final model aggregation. The outlier removal procedure of BREA follows the multi-Krum algorithm from [9], [38]. The main difference is that our framework considers the multi-Krum algorithm in a quantized stochastic gradient setting, as BREA utilizes quantized gradients instead of the true gradients, in order to enable privacy-preserving computations on the secret shares. We present the theoretical convergence guarantees of this quantized multi-Krum algorithm in Section VI, and numerically demonstrate its convergence behaviour in our experiments in Section VII.

In this setup, the server selects m users through the following iterative process. At each iteration $k \in [m]$, the server selects one user, denoted by $i^{(k)}$, by finding

$$i^{(k)} = \arg \min_{j \in [N] \setminus S^{(k-1)}} s^{(k)}(j) \tag{24}$$

where $S^{(k)}$ denotes the index set of the users selected in up to k iterations and $s^{(k)}(j)$ is a score function assigned to user j at iteration k. The score function of user j is defined as

$$s^{(k)}(j) = \sum_{u \in \mathcal{I}_j^{(k-1)}} d_{ju} \tag{25}$$

where $\mathcal{I}_j^{(k-1)} \subseteq [N] \setminus S^{(k-1)}$ denotes the set of $(N-k+1) - A - 2$ users whose models are closest to the model of user j. After selecting $i^{(k)}$, the server updates the selected index set as $S^{(k)} = \{S^{(k-1)}, i^{(k)}\}$ where $S^{(0)} = \emptyset$. After m iterations, the server obtains the index set $S = S^{(m)}$.

E. Secure Model Aggregation

The final phase of BREA is to securely aggregate the local updates of the selected users, without revealing the individual models to the server. To do so, the server initially announces the list of selected users via broadcasting. We denote the set of selected users by S. Then, each user locally aggregates the secret shares belonging to the selected users,

$$\mathbf{s}_i = \sum_{j \in S} \mathbf{s}_{ji} \tag{26}$$

and sends the result to the server. Upon receiving the computation results from a sufficient number of users, the server can decode the aggregate of the models $\sum_{j \in S} \overline{\mathbf{w}}_j$ through the decoding of Reed-Solomon codes.

The intuition of the decoding process is similar to the decoding of the pairwise distances described in Section V-D. Specifically, the computations from (26) can be viewed as evaluation points of a univariate polynomial $h : \mathbb{F}_p \to \mathbb{F}_p^d$ of degree at most T,

$$h(\theta) := \sum_{j \in S} f_j(\theta). \tag{27}$$

One can then observe that h is the encoding polynomial of a Reed-Solomon code with degree at most T, the missing computations due to the dropped users correspond to the erasures

in the code, and manipulated computations from Byzantine users correspond to the errors in the code. Therefore, the decoding process of the server corresponds to decoding an $[N, T+1, N-T]_p$ Reed-Solomon code with at most D erasures and at most A errors. Hence, by using a Reed-Solomon decoding algorithm, the server can recover the polynomial h and obtain the true value of the aggregate of the selected user models by using the relation $h(0) = \sum_{j \in \mathcal{S}} f_j(0) = \sum_{j \in \mathcal{S}} \overline{\mathbf{w}}_j$. We note that the total number of users selected by the server for aggregation, i.e., $|\mathcal{S}| = m$, should be sufficiently large, which can be agreed offline between the users and the server. Then, if the set announced by the server is too small (e.g., consisting of a single user), the honest users may opt to not send the computation results.

Upon learning the aggregate of the user models, the server updates the global model for the next iteration as follows,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\gamma^{(t)}}{q} \phi^{-1} \left(\sum_{j \in \mathcal{S}} \overline{\mathbf{w}}_j^{(t)} \right) \qquad (28)$$

where ϕ^{-1} is the demapping function defined in (20) and q is the integer parameter in (9). We assume that the field size p is large enough to avoid wrap-around in $\sum_{j \in \mathcal{S}} \overline{\mathbf{w}}_j^{(t)}$ such that

$$\sum_{j \in \mathcal{S}} \overline{\mathbf{w}}_j^{(t)} = \sum_{j \in \mathcal{S}} \phi(q \cdot Q_q(\mathbf{w}_j^{(t)})) \qquad (29)$$

$$= \phi\left(q \sum_{j \in \mathcal{S}} \mathbf{w}_j^{(t)} \right) \qquad (30)$$

where (29) follows from (10). Finally, it follows from (30) that the update equation in (28) is equivalent to

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \gamma^{(t)} \sum_{j \in \mathcal{S}} Q_q(\mathbf{w}_j^{(t)}) \qquad (31)$$

where Q_q is the stochastic rounding function defined in (9).

Having all above steps, the overall BREA framework can now be presented in Algorithm 1.

VI. THEORETICAL ANALYSIS

In this section, we analyze the fundamental performance limits of BREA. The global model update equation of BREA can be expressed as follows,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \gamma^{(t)} f\left(Q_q(\mathbf{w}_1^{(t)}), \ldots, Q_q(\mathbf{w}_N^{(t)}) \right) \qquad (32)$$

where f is the aggregation operation from (8) and represents the user selection and model aggregation procedures from Sections V-D and V-E, respectively, while Q_q is the stochastic rounding function defined in (9).

As described in Section III, the local update $\mathbf{w}_i^{(t)}$ created by an honest user is an unbiased estimator of the true gradient, i.e., $\mathbf{w}_i^{(t)} = g(\mathbf{w}^{(t)}, \xi_i^{(t)})$ with $\mathbb{E}_\xi[g(\mathbf{w}^{(t)}, \xi_i^{(t)})] = \nabla C(\mathbf{w}^{(t)})$ where $\xi_i^{(t)} \sim \xi$ and ξ is a uniform random variable representing the random sample (or a mini-batch of samples) drawn from the dataset. We define the local standard deviation σ of the gradient estimator g by

$$d\sigma^2(\mathbf{w}) := \mathbb{E}_\xi \|g(\mathbf{w}, \xi) - \nabla C(\mathbf{w})\|^2 \qquad (33)$$

Algorithm 1 Byzantine-Resilient Secure Aggregation (BREA)

input Local dataset $\mathcal{D}_i$ of users $i \in [N]$, number of iterations J.

output Global model ($\mathbf{w}^{(J)}$).

1: **for** iteration $t = 0, \ldots, J - 1$ **do**
2:     **for** user $i = 1, \ldots, N$ **do**
3:         Download the global model $\mathbf{w}^{(t)}$ from the server.
4:         Create a local model $\mathbf{w}_i^{(t)}$ from (3).
5:         Create the quantized model $\overline{\mathbf{w}}_i = \phi(q \cdot Q_q(\mathbf{w}_i))$ from (10).
6:         Generate secret shares $\{\mathbf{s}_{ij}\}_{j \in [N]}$ from (13) and send $\mathbf{s}_{ij}$ to user $j \in [N]$.
7:         Generate commitments $\{\mathbf{c}_{ij}\}_{j \in [T]}$ from (14) and broadcast $\{\mathbf{c}_{ij}\}_{j \in [T]}$ to all users.
8:         Verify the secret shares $\{\mathbf{s}_{ji}\}_{j \in [N]}$ by testing (15).
9:         Compute $\{d_{jk}^{(i)}\}_{j,k \in [N]}$ from (16) and send the results to the server.
10:     Server recovers $\{\overline{d}_{jk}\}_{j,k \in [N]}$ in (18) from the computation results $\{d_{jk}^{(i)}\}_{i,j,k \in [N]}$ by utilizing Reed-Solomon decoding.
11:     Server converts $\{\overline{d}_{jk}\}_{j,k \in [N]}$ from the finite field to the real domain to obtain the pairwise distances $\{d_{jk}\}_{j,k \in [N]}$ from (19).
12:     Server selects the set $\mathcal{S}$ by utilizing the multi-Krum algorithm [9] based on the pairwise distances $\{d_{jk}\}_{j,k \in [N]}$.
13:     Server broadcasts the set $\mathcal{S}$ to all users.
14:     **for** user $i = 1, \ldots, N$ **do**
15:         Compute $\mathbf{s}_i = \sum_{j \in \mathcal{S}} \mathbf{s}_{ji}$ from (26) and send the result to the server.
16:     Server recovers $\sum_{i \in \mathcal{S}} \mathbf{w}_i^{(t)}$ from the computation results $\{\mathbf{s}_i\}_{i \in [N]}$ by utilizing Reed-Solomon decoding.
17:     Server updates the global model, $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\gamma^{(t)}}{q} \phi^{-1} \left(\sum_{i \in \mathcal{S}} \overline{\mathbf{w}}_i^{(t)} \right)$.

---

for all $\mathbf{w} \in \mathbb{R}^d$. The model created by a Byzantine user can refer to any random vector $\mathbf{b}_i^{(t)} \in \mathbb{R}^d$, which we represent as $\mathbf{w}_i^{(t)} = \mathbf{b}_i^{(t)}$. Accordingly, the quantized model $Q(\mathbf{w}_i^{(t)})$ of Byzantine user could refer to any vector in $\mathbb{Z}^d$.

Our first lemma states the unbiasedness and bounded variance of the quantized gradient estimator $Q_q(g(\mathbf{w}, \xi))$ for any vector $\mathbf{w} \in \mathbb{R}^d$.

Lemma 1. *For the quantized gradient estimator $Q_q(g(\mathbf{w}, \xi))$ with a given vector $\mathbf{w} \in \mathbb{R}^d$ where ξ is a uniform random variable representing the sample drawn, g is a gradient estimator such that $\mathbb{E}_\xi[g(\mathbf{w}, \xi)] = \nabla C(\mathbf{w})$ and $\mathbb{E}_\xi \|g(\mathbf{w}, \xi) - \nabla C(\mathbf{w})\|^2 = d\sigma^2(\mathbf{w})$, and the stochastic rounding function Q_q is given in (9), the following holds,*

$$\mathbb{E}_{Q,\xi}[Q_q(g(\mathbf{w}, \xi))] = \nabla C(\mathbf{w}) \qquad (34)$$

$$\mathbb{E}_{Q,\xi}\|Q_q(g(\mathbf{w}, \xi)) - \nabla C(\mathbf{w})\|^2 \leq d\sigma'^2(\mathbf{w}) \qquad (35)$$

where $\sigma'(\mathbf{w}) = \sqrt{\frac{1}{4q^2} + \sigma^2(\mathbf{w})}$.

Proof. (Unbiasedness) Given Q_q in (9) and any random vari-

able x, it follows that,

$$\mathbb{E}_Q[Q_q(x) \mid x] = \frac{\lfloor qx \rfloor}{q}(1 - (qx - \lfloor qx \rfloor))$$
$$+ \frac{(\lfloor qx \rfloor + 1)}{q}(qx - \lfloor qx \rfloor)$$
$$= x \qquad (36)$$

from which we obtain the unbiasedness condition in (34),

$$\mathbb{E}_{Q,\xi}[Q_q(g(\mathbf{w}, \xi))] = \mathbb{E}_\xi\big[\mathbb{E}_Q[Q_q(g(\mathbf{w}, \xi)) \mid g(\mathbf{w}, \xi)]\big]$$
$$= \mathbb{E}_\xi\big[g(\mathbf{w}, \xi)\big]$$
$$= \nabla C(\mathbf{w}). \qquad (37)$$

(Bounded variance) Next, we observe that,

$$\mathbb{E}_Q\Big[(Q_q(x) - \mathbb{E}_Q[Q_q(x) \mid x])^2 \mid x\Big]$$
$$= \left(\frac{\lfloor qx \rfloor}{q} - x\right)^2 (1 - (qx - \lfloor qx \rfloor))$$
$$+ \left(\frac{\lfloor qx \rfloor + 1}{q} - x\right)^2 (qx - \lfloor qx \rfloor)$$
$$= \frac{1}{q^2}\Big(\frac{1}{4} - (qx - \lfloor qx \rfloor - \frac{1}{2})^2\Big)$$
$$\leq \frac{1}{4q^2} \qquad (38)$$

from which one can obtain the bounded variance condition in (35) as follows,

$$\mathbb{E}_{Q,\xi}\|Q_q(g(\mathbf{w}, \xi)) - \nabla C(\mathbf{w})\|^2$$
$$= \mathbb{E}_\xi\big[\mathbb{E}_Q[\|Q_q(g(\mathbf{w}, \xi)) - \nabla C(\mathbf{w})\|^2 \mid g(\mathbf{w}, \xi)]\big]$$
$$\leq \mathbb{E}_\xi\big[\mathbb{E}_Q[\|Q_q(g(\mathbf{w}, \xi)) - g(\mathbf{w}, \xi)\|^2 \mid g(\mathbf{w}, \xi)]\big]$$
$$+ \mathbb{E}_\xi\big[\mathbb{E}_Q[\|g(\mathbf{w}, \xi) - \nabla C(\mathbf{w})\|^2 \mid g(\mathbf{w}, \xi)]\big] \qquad (39)$$
$$\leq \frac{d}{4q^2} + d\sigma^2(\mathbf{w}) \qquad (40)$$
$$= d\sigma'^2(\mathbf{w})$$

where (39) follows from the triangle inequality and (40) follows form (38). $\qquad \square$

As discussed in Section IV, Byzantine users can manipulate the training protocol via two means, either by modifying their local update (directly or by modifying the local dataset), or by sharing false information during protocol execution. In this section, we demonstrate how BREA provides robustness in both cases. We first focus on the former case and study the resilience of the global model, i.e., conditions under which the trained model remains close to the true model, even if some users modify their local updates adversarially. The second case, i.e., robustness of the protocol when some users exchange false information during the protocol execution, will be considered in Theorem 1.

In order to evaluate the resilience of the global model against Byzantine adversaries, we adopt the notion of (α, A)-Byzantine resilience from [38].

Definition 1 ((α, A)-Byzantine resilience, [38])**.** *Let $0 \leq \alpha < \pi/2$ be any angular value and $0 \leq A \leq N$ be any integer. Let $\mathbf{w}_1, \ldots, \mathbf{w}_N \in \mathbb{R}^d$ be any i.i.d random vectors such that*

$\mathbf{w}_i \sim \mathbf{w}$ *with $\mathbb{E}[\mathbf{w}] = \mathbf{W}$. Let $\mathbf{b}_1, \ldots, \mathbf{b}_A \in \mathbb{R}^d$ be any random vectors. Then, function f is (α, A)-Byzantine resilient if, for any $1 \leq j_1 < \cdots < j_A \leq N$,*

$$\mathbf{f} := f(\mathbf{w}_1, \ldots, \underbrace{\mathbf{b}_1}_{j_1}, \ldots, \underbrace{\mathbf{b}_A}_{j_A}, \ldots, \mathbf{w}_N) \qquad (41)$$

satisfies, i) $\mathbf{W}^\top \mathbb{E}[\mathbf{f}] \geq (1 - \sin\alpha)\|\mathbf{W}\|^2$, and ii) for $r \in \{2, 3, 4\}$, $\mathbb{E}\|\mathbf{f}\|^r$ is bounded above by $\mathbb{E}\|\mathbf{f}\|^r \leq K \sum_{r_1 + \cdots + r_{N-A} = r} \mathbb{E}\|\mathbf{W}\|^{r_1} \ldots \mathbb{E}\|\mathbf{W}\|^{r_{N-A}}$ where K denotes a generic constant.

Lemma 2 below states that if the standard deviation caused by random sample selection and quantization is smaller than the norm of the true gradient, and $2A + 2 < N - m$, then the aggregation function f from (32) is (α, A)-Byzantine resilient where α depends on the ratio of the standard deviation over the norm of the gradient [38].

Lemma 2. *Assume that $2A + 2 < N - m$ and $\eta(N, A)\sqrt{d}\sigma' < \|\nabla C(\mathbf{w})\|$ where*

$$\eta(N, A) := \sqrt{2\Big(N - A + \frac{A(N - A - 2) + A^2(N - A - 1)}{N - 2A - 2}\Big)}. \qquad (42)$$

Let $\mathbf{w}_1, \ldots, \mathbf{w}_N$ be i.i.d. random vectors in $\mathbb{R}^d$ such that $\mathbf{w}_i \sim \mathbf{w}$ with $\mathbb{E}_\xi[g(\mathbf{w}, \xi)] = \nabla C(\mathbf{w})$ and $\mathbb{E}_\xi\|g(\mathbf{w}, \xi) - \nabla C(\mathbf{w})\|^2 = d\sigma^2(\mathbf{w})$. Then, the aggregation function f from (32) is (α, A)-Byzantine resilient where $0 \leq \alpha < \pi/2$ is defined by $\sin\alpha = \frac{\eta(N, A)\sqrt{d}\sigma'}{\|\nabla C(\mathbf{w})\|}$.

Proof. From Lemma 1, $\mathbb{E}_{Q,\xi}[Q_q(g(\mathbf{w}, \xi))] = \nabla C(\mathbf{w})$ and $\mathbb{E}_{Q,\xi}\|Q_q(g(\mathbf{w}, \xi)) - \nabla C(\mathbf{w})\|^2 \leq d\sigma'^2(\mathbf{w})$. Then, the quantized multi-Krum algorithm described in Section V-D, where the multi-Krum algorithm applied to the quantized vectors $Q_q(\mathbf{w}_i)$, is (α, A)-Byzantine resilient from Proposition 3 of [38]. Hence, function f in (32) is (α, A)-Byzantine resilient. $\qquad \square$

We now state our main result for the theoretical performance guarantees of BREA.

Theorem 1. *We assume that: 1) the cost function C is three times differentiable with continuous derivatives, and is bounded from below, i.e., $C(x) \geq 0$; 2) the learning rates satisfy, $\sum_{t=1}^\infty \gamma^{(t)} = \infty$ and $\sum_{t=1}^\infty (\gamma^{(t)})^2 < \infty$; 3) the second, third, and fourth moments of the quantized gradient estimator do not grow too fast with the norm of the model, i.e., $\forall r \in 2, 3, 4$, $\mathbb{E}_{Q,\xi}\|Q_q(g(\mathbf{w}, \xi))\|^r \leq A_r + B_r\|\mathbf{w}\|^r$ for some constants A_r and B_r; 4) there exist a constant $0 \leq \alpha < \pi/2$ such that for all $\mathbf{w} \in \mathbb{R}^d$, $\eta(N, A)\sqrt{d}\sigma'(\mathbf{w}) \leq \|\nabla C(\mathbf{w})\| \sin\alpha$; 5) the gradient of the cost function C satisfies that for $\|\mathbf{w}\|^2 \geq R$, there exist constants $\epsilon > 0$ and $0 \leq \beta < \pi/2 - \alpha$ such that*

$$\|\nabla C(\mathbf{w})\| \geq \epsilon > 0, \qquad (43)$$
$$\frac{\mathbf{w}^\top C(\mathbf{w})}{\|\mathbf{w}\| \cdot \|\nabla C(\mathbf{w})\|} \geq \cos\beta. \qquad (44)$$

Then, BREA guarantees,

- *(Robustness against Byzantine users) The protocol executes correctly against up to A Byzantine users and the trained model is (α, A)-Byzantine resilient.*

- *(Convergence) The sequence of the gradients $\nabla C(\mathbf{w}^{(t)})$ converges almost surely to zero,*

$$\nabla C(\mathbf{w}^{(t)}) \xrightarrow[t\to\infty]{a.s.} 0. \tag{45}$$

- *(Privacy) The server or any group of up to T users cannot compute an unknown local update. For any set of users $\mathcal{T} \subset [N]$ of size at most T,*

$$\mathbb{P}[\text{User } i \text{ has secret } \overline{\mathbf{w}}_i \mid \text{view}_{\mathcal{T}}]$$
$$= \mathbb{P}[\text{User } i \text{ has secret } \overline{\mathbf{w}}_i] \tag{46}$$

for all $i \in [N] \setminus \mathcal{T}$ where $\text{view}_{\mathcal{T}}$ denotes the messages that the members of $\mathcal{T}$ receive.

for any $N \geq 2A + 1 + \max\{m + 2, D + 2T\}$, where m is the number of selected models for aggregation.

Remark 1. *The two conditions $\sum_{t=1}^{\infty} \gamma^{(t)} = \infty$ and $\sum_{t=1}^{\infty} (\gamma^{(t)})^2 < \infty$ are instrumental in the convergence of stochastic gradient descent algorithms [40]. Condition $\sum_{t=1}^{\infty} (\gamma^{(t)})^2 < \infty$ states that the learning rates decrease fast enough, whereas condition $\sum_{t=1}^{\infty} \gamma^{(t)} = \infty$ bounds the rate of their decrease, to ensure that the learning rates do not decrease too fast.*

Remark 2. *We consider a general (possibly non-convex) objective function C. In such scenarios, proving the convergence of the model directly is challenging, and various approaches have been proposed instead. Our approach follows [40] and [9], where we prove the convergence of the gradient to a flat region instead. We note, however, that such a region may refer to any stationary point, including the local minima as well as saddle and extremal points.*

Proof. (Robustness against Byzantine users) The (α, A)-Byzantine resilience of the trained model follows from Lemma 2. We next provide sufficient conditions for BREA to correctly evaluate the update function (32), in the presence of A Byzantine users. Byzantine users may send any arbitrary random vector to the server or other users in every step of the protocol in Section V. In particular, Byzantine users can create and send incorrect computations in three attack scenarios: i) sending invalid secret shares $\mathbf{s}_{ij}$ in (13), ii) sending incorrect secure distance computations $d_{jk}^{(i)}$ in (16), and iii) sending incorrect aggregate of the secret shares $\mathbf{s}_i$ in (26).

The first attack scenario occurs when the secret shares $\mathbf{s}_{ij}$ in (13) do not refer to the same polynomial from (12). BREA utilizes verifiable secret sharing to prevent such attempts. The correctness (validity) of the secret shares can be verified by testing (15), whenever the majority of the surviving users are honest, i.e., $N > 2A + D$ [17], [36].

The second attack scenario can be detected and corrected by the Reed-Solomon decoding algorithm. In particular, as described in Section V-D, given $j, k \in [N]$, $\{d_{jk}^{(i)}\}_{i \in [N]}$ can be viewed as N evaluation points of the polynomial h_{jk} given in (17) whose degree is at most $2T$. The decoding process at the server then corresponds to the decoding of an $[N, 2T + 1, N - 2T]_p$ Reed-Solomon code with at most D erasures and at most A errors. As an $[n, k, n - k + 1]_p$ Reed-Solomon code with e erasures can tolerate a maximum

number of $\lfloor \frac{n-k-e}{2} \rfloor$ errors [39], the server can recover the correct pairwise distances as long as $A \leq \lfloor \frac{N-(2T+1)-D}{2} \rfloor$, i.e. $N \geq D + 2A + 2T + 1$.

The third attack scenario can also be detected and corrected by the Reed-Solomon decoding algorithm. As described in Section V-E, $\{\mathbf{s}_i\}_{i \in [N]}$ are evaluation points of polynomial h in (27) of degree at most T. This decoding process corresponds to the decoding of an $[N, T + 1, N - T]_p$ Reed-Solomon code with at most D erasures and at most A errors. As such, the server can recover the desired aggregate model $h(0) = \sum_{j \in \mathcal{S}} \overline{\mathbf{w}}_j$ as long as $N \geq D + 2A + T + 1$. Therefore, combining with the condition of Lemma 2, the sufficient conditions under which BREA guarantees robustness against Byzantine users is given by

$$N \geq 2A + 1 + \max\{m + 2, D + 2T\}. \tag{47}$$

(Convergence) We now consider the update equation in (32) and prove the convergence of the random sequence $\nabla C(\mathbf{w}^{(t)})$. From Lemma 2, the quantized multi-Krum function f in (32) is (α, A)-Byzantine resilient. Hence, from Proposition 2 of [38], $\nabla C(\mathbf{w}^{(t)})$ converges almost surely to zero,

$$\nabla C(\mathbf{w}^{(t)}) \xrightarrow[t\to\infty]{a.s.} 0. \tag{48}$$

(Privacy) As described in Section V-B, we assume the intractability of computing discrete logarithms, hence the server or any user cannot compute $\overline{\mathbf{w}}_i$ from $\mathbf{c}_{i0}$ in (14). It is therefore sufficient to prove the privacy of each individual model against a group of T colluding users, in the case where $\mathcal{T}$ has size T. If T users cannot get any information about $\overline{\mathbf{w}}_i$, then neither can fewer than T users. Without loss of generality, let $\mathcal{T} = \{1, \ldots, T\} = [T]$ and $\text{view}_{\mathcal{T}} = \left\{ (\mathbf{s}_{kj})_{k \in [N]}, (d_{kl}^j)_{k,l \in [N]}, \mathbf{s}_j \right\}_{j \in [T]}$ where $\mathbf{s}_{kj}$ in (13) is the secret share of $\overline{\mathbf{w}}_k$ sent from user k to user j, d_{kl}^j in (16) is the pairwise distance of the secret shares sent from users k and l to user j, and $\mathbf{s}_j$ in (26) is the aggregate of the secret shares. As $\left\{ (d_{kl}^j)_{k,l \in [N]} \right\}_{j \in [T]}$ and $\{\mathbf{s}_j\}_{j \in [T]}$ are determined by $\left\{ (\mathbf{s}_{kj})_{k,l \in [N]} \right\}_{j \in [T]}$, we can simplify the left hand side (LHS) of (46) as

$$\mathbb{P}[\text{User } i \text{ has secret } \overline{\mathbf{w}}_i \mid \text{view}_{\mathcal{T}}]$$
$$= \mathbb{P}[\text{User } i \text{ has secret } \overline{\mathbf{w}}_i \mid \{\mathbf{s}_{kj}\}_{k \in [N], j \in [T]}] \tag{49}$$
$$= \mathbb{P}[\text{User } i \text{ has secret } \overline{\mathbf{w}}_i \mid \{\mathbf{s}_{ij}\}_{j \in [T]}]. \tag{50}$$

where (50) follows from the fact that $\mathbf{s}_{kj}$ is independent of $\overline{\mathbf{w}}_i$ for any $k \neq i$, $\mathbf{s}_{kj}$ is independent of $\overline{\mathbf{w}}_i$. Then, for any realization of vectors $\rho_0, \ldots, \rho_T \in \mathbb{F}_p^d$, we obtain,

$$\mathbb{P}[\text{User } i \text{ has secret } \overline{\mathbf{w}}_i \mid \text{view}_{\mathcal{T}}]$$
$$= \mathbb{P}[\overline{\mathbf{w}}_i = \rho_0 \mid \mathbf{s}_{i1} = \rho_1, \ldots, \mathbf{s}_{iT} = \rho_T]$$
$$= \frac{\mathbb{P}[\overline{\mathbf{w}}_i = \rho_0, \mathbf{s}_{i1} = \rho_1, \ldots, \mathbf{s}_{iT} = \rho_T]}{\mathbb{P}[\mathbf{s}_{i1} = \rho_1, \ldots, \mathbf{s}_{iT} = \rho_T]}$$
$$= \frac{1/|\mathbb{F}_p^d|^{T+1}}{1/|\mathbb{F}_p^d|^T} \tag{51}$$
$$= \frac{1}{|\mathbb{F}_p^d|} = \mathbb{P}[\overline{\mathbf{w}}_i = \rho_0]$$

where (51) follows from the fact that any $T + 1$ evaluation

TABLE I
COMPLEXITY SUMMARY OF BREA.

	Computation	Communication
Server	$O((N^3 + dN)\log^2 N \log\log N)$	$O(dN + N^3)$
User	$O(dN\log^2 N + dN^2)$	$O(dN + N^2)$

TABLE II
COMPLEXITY SUMMARY OF GENERALIZED BREA.

	Computation	Communication
Server	$O((N^2 + dN)\log^2 N \log\log N)$	$O(dN + N^2)$
User	$O(dN\log^2 N + dN^2)$	$O(dN + N)$

points define a unique polynomial of degree T, which completes the proof of privacy. □

A. Complexity Analysis

In this section, we analyze the complexity of BREA with respect to the number of users, N, and model dimension d.
Complexity Analysis of the Users: User i's computation cost can be broken into three parts: 1) generating the secret shares $\mathbf{s}_{ij}$ in (12) for $j \in [N]$, 2) computing the pairwise distances $d_{jk}^{(i)}$ in (16) for $j, k \in [N]$, and 3) aggregating the secret shares belonging to the selected users from (26). First, generating N secret shares of a vector with dimension d has a computation cost of $O(dN\log^2 N)$ [18]. Second, as there are $O(N^2)$ pairwise distances, computing the pairwise distances has a computation cost of $O(dN^2)$ in total. Third, when the number of selected users $m = |\mathcal{S}|$ is $O(N)$, aggregating the secret shares belonging to the selected users has a computation cost of $O(dN)$. Therefore, the overall computation cost of each user is $O(dN\log^2 N + dN^2)$.

User i's communication cost can be broken to three parts: 1) sending the secret share $\mathbf{s}_{ij}$ to user $j \in [N]$, 2) sending the secret shares of the pairwise distances $d_{jk}^{(i)}$ to the server for $j, k \in [N]$, and 3) sending the aggregate of secret the shares $\mathbf{s}_i$ in (26) to the server. The communication cost of the three parts are $O(dN)$, $O(N^2)$, and $O(d)$, respectively. Therefore, the overall communication cost of each user is $O(dN + N^2)$.
Complexity Analysis of the Server: Computation cost of the server can be broken into three parts: 1) decoding the pairwise distances by recovering $h_{jk}(0)$ in (17) for $j, k \in [N]$, 2) carrying out the multi-Krum algorithm to select the m users for aggregation, and 3) decoding the aggregate of the selected models by recovering $h(0)$ in (27). As described in Section V-D, recovering the polynomial h_{jk} corresponds to decoding an $[N, 2T + 1, N - 2T]_p$ Reed-Solomon code with at most D erasures and at most A errors, which has an $O(N\log^2 N \log\log N)$ computation cost [39]. As there are $O(N^2)$ pairs and each pair is embedded in a single polynomial, the computation cost of the first part is $O(N^3 \log^2 N \log\log N)$ in total. The computation cost of the second part is $O(dN^2 + N^2 \log N)$ [9]. As the dimension of $h(0)$ is d, the computation cost of the third part is $O(dN\log^2 N \log\log N)$. Overall, the computation cost of the server is $O((N^3 + dN)\log^2 N \log\log N)$.

Communication cost can be broken into two parts: 1) receiving the secret shares of the pairwise distances $d_{jk}^{(i)}$ from users $i \in [N]$ for $j, k \in [N]$ and 2) receiving the aggregate of secret shared models $\mathbf{s}_i$ from users $i \in [N]$. The communication cost of the two parts are $O(N^3)$ and $O(dN)$, respectively. Overall, the communication cost of the server is $O(dN + N^3)$.

We summarize the complexity analysis in Table I. As can be observed in Table I, the server has a communication cost of $O(N^3)$ and a computation cost of $O(N^3 \log^2 N \log\log N)$, which is due to the recovery of N^2 pairwise distances. Although the distances are scalar valued, the overhead can become a limitation for very large-scale networks. In the next subsection, we propose a generalized framework to reduce the communication cost from $O(N^3)$ to $O(N^2)$ as well as to reduce the computation cost from $O(N^3 \log^2 N \log\log N)$ to $O(N^2 \log^2 N \log\log N)$.

B. The Generalized BREA Framework

The key idea of the generalized framework is to partition the set of $N(N-1)/2$ pairwise distances into sets of size K, and embed the K distances in a single polynomial. Consequently, the number of polynomials to embed the $N(N-1)/2$ pairwise distances can be reduced from $O(N^2)$ to $O(N^2/K)$. Each user then sends a single evaluation point of each polynomial to the server, which has an $O(N^3/K)$ communication cost in total. By setting $K = O(N)$, the generalized BREA framework can achieve $O(N^2)$ communication complexity.

We now present the details of the generalized BREA framework. In the stochastic quantization phase, the generalized BREA framework follows the same steps as in Section V-A. In the verifiable secret sharing phase, user $i \in [N]$ generates secret shares of the quantized model $\overline{\mathbf{w}}_i$ by modifying the random polynomial f_i in (12) as,

$$f_i(\theta) = \overline{\mathbf{w}}_i + \sum_{j=1}^{T} \mathbf{r}_{ij}\theta^{K+j-1} \qquad (52)$$

where the degree of f_i is increased from T to $K+T-1$. User i then sends a secret share of $\overline{\mathbf{w}}_i$ to user $j \in [N]$, denoted by $\mathbf{s}_{ij} = f_i(\theta_j)$.

In the secure distance computation phase, user i computes the pairwise distances $d_{jk}^{(i)}$ as in (16) for $j, k \in [N]$. Note that there are $\frac{N(N-1)}{2}$ unique distance values, due to the symmetry of the distance measure. The server and the users then agree on a partition $\mathcal{P}_1, \ldots, \mathcal{P}_G$ of the $\frac{N(N-1)}{2}$ distances into sets of size K, where $G = \lceil \frac{N(N-1)}{2K} \rceil$ and $|\mathcal{P}_k| = K$ for all $k \in [G]$. $\lceil x \rceil$ denotes the smallest integer greater or equal to x. Then, each user embeds the K pairwise distances in each $\mathcal{P}_k$ into a single polynomial, for every $k \in \{1, \ldots, G\}$. For instance, let the first set in the partition be $\mathcal{P}_1 = \{(1, 2), \ldots, (1, K+1)\}$, Then, one can define a univariate polynomial $u_{\mathcal{P}_1} : \mathbb{F}_p \to \mathbb{F}_p$ of degree $2(K+T-1)+K-1$ to embed the K pairwise distances of the pairs in $\mathcal{P}_1$ as

$$u_{\mathcal{P}_1}(\theta) = \|\mathbf{f}_1(\theta) - \mathbf{f}_2(\theta)\|^2 + \theta\|\mathbf{f}_1(\theta) - \mathbf{f}_3(\theta)\|^2$$
$$+ \cdots + \theta^{K-1}\|\mathbf{f}_1(\theta) - \mathbf{f}_{K+1}(\theta)\|^2. \qquad (53)$$

As the coefficients from the first degree to the $(K-1)$-th degree terms in (52) are zero, the coefficient of the $(k-1)$-th degree term in (53) corresponds to the pairwise distance of the k-th pair in $\mathcal{P}_1$, i.e., $\|\mathbf{f}_1(0) - \mathbf{f}_{k+1}(0)\|^2 = \|\overline{\mathbf{w}}_1 - \overline{\mathbf{w}}_{k+1}\|^2$ for all $k \in [K]$. User i then sends $u_{\mathcal{P}_1}(\theta_i)$ to the server. Upon receiving the computation results from a sufficient number of users, the server can decode the pairwise distances of all pairs in $\mathcal{P}_1$ by reconstructing the polynomial $u_{\mathcal{P}_1}(\theta_i)$. As the degree of $u_{\mathcal{P}_1}$ is $3K + 2T - 3$, the minimum number of results the server needs to collect from the users to recover the pairwise distances, i.e., the recovery threshold, is $3K + 2T - 2$. The decoding process of the polynomial corresponds to decoding an $[N, 3K + 2T - 3, N - 3K - 2T + 2]_p$ Reed-Solomon code with at most D erasures and at most A errors. In a similar way, we can define polynomials $u_{\mathcal{P}_k}$ for $k \in \{2, \ldots, G\}$ where user i computes and sends $u_{\mathcal{P}_k}(\theta_i)$ to the server. The server can then decode all of the $N(N-1)/2$ pairwise distances by reconstructing the G polynomials.

After the server learns the pairwise distances, the generalized protocol follows the same steps as in Sections V-D and V-E. The overall algorithm of the generalized BREA framework is presented in Algorithm 2.

The generalized BREA framework achieves a communication cost of $O(N^2)$ and computation cost of $O(N^2 \log^2 N \log \log N)$ by setting $K = O(N)$, which follows from the following observations. First, user i sends an evaluation point of each polynomial $u_{\mathcal{P}_k}$ to the server for $k \in [G]$, and there are N users and $G = O(N^2/K) = O(N)$ polynomials, which has $O(N^2)$ communication overhead in total. Second, the computation cost to decode the polynomial $u_{\mathcal{P}_k}$ is $O(N \log^2 N \log \log N)$ [39] and there are $G = O(N)$ polynomials, which has a computation cost of $O(N^2 \log^2 N \log \log N)$ in total. We summarize the complexity of the generalized BREA protocol in Table II.

VII. Experiments

In this section, we demonstrate the convergence and resilience properties of BREA compared to conventional federated learning, i.e., the federated averaging scheme from [1], which is termed *FedAvg* throughout the section. We measure the performance in terms of the cross entropy loss evaluated over the training samples and the model accuracy evaluated over the test samples, with respect to the iteration index, t.

Network architecture: We consider an image classification task with 10 classes on the MNIST dataset [41], and train a convolutional neural network with 6 layers [1] including two 5×5 convolutional layers with stride 1, where the first and the second layers have 32 and 64 channels, respectively, and each is followed by ReLu activation and 2×2 max pooling layer. It also includes a fully connected layer with 1024 units and ReLu activation followed by a final softmax output layer.

Experiment setup: We assume a network of $N = 40$ users where $T = 7$ users may collude and A users are malicious. We consider two cases for the number of Byzantine users: i) 0% Byzantine users ($A = 0$) and ii) 30% Byzantine users ($A = 12$). For i.i.d. data distribution, 60000 training samples are shuffled and then partitioned into $N = 40$ users each receiving 1500

Algorithm 2 Generalized BREA

input Local dataset $\mathcal{D}_i$ of users $i \in [N]$, number of iterations J.

output Global model $(\mathbf{w}^{(J)})$.

1: **for** iteration $t = 0, \ldots, J-1$ **do**
2:     **for** user $i = 1, \ldots, N$ **do**
3:       Download the global model $\mathbf{w}^{(t)}$ from the server.
4:       Create a local update $\mathbf{w}_i^{(t)}$ from (3).
5:       Create the quantized model $\overline{\mathbf{w}}_i = \phi(q \cdot Q_q(\mathbf{w}_i))$ from (10).
6:       Generate secret shares $\{\mathbf{s}_{ij} = f_i(\theta_j)\}_{j \in [N]}$ from (52) and send $\mathbf{s}_{ij}$ to user $j \in [N]$.
7:       Generate commitments $\{\mathbf{c}_{ij}\}_{j \in [T]}$ from (14) and broadcast $\{\mathbf{c}_{ij}\}_{j \in [T]}$ to all users.
8:       Verify the secret shares $\{\mathbf{s}_{ji}\}_{j \in [N]}$ by testing (15).
9:       Compute $\{d_{jk}^{(i)}\}_{j,k \in [N]}$ from (16) and send the results to the server.
10:     Server recovers $\{\overline{d}_{jk}\}_{j,k \in [N]}$ in (18) from the coefficients of polynomials $u_{\mathcal{P}_1}, \ldots, u_{\mathcal{P}_G}$ in (53) by utilizing Reed-Solomon decoding.
11:     Server converts $\{\overline{d}_{jk}\}_{j,k \in [N]}$ from the finite field to the real domain to obtain the pairwise distances $\{d_{jk}\}_{j,k \in [N]}$ from (19).
12:     Server selects the set $\mathcal{S}$ by utilizing the multi-Krum algorithm [9] based on the pairwise distances $\{d_{jk}\}_{j,k \in [N]}$.
13:     Server broadcasts the set $\mathcal{S}$ to all users.
14:     **for** user $i = 1, \ldots, N$ **do**
15:       Compute $\mathbf{s}_i = \sum_{j \in \mathcal{S}} \mathbf{s}_{ji}$ from (26) and send the result to the server.
16:     Server recovers $\sum_{i \in \mathcal{S}} \mathbf{w}_i^{(t)}$ from the computation results $\{\mathbf{s}_i\}_{i \in [N]}$ by utilizing Reed-Solomon decoding.
17:     Server updates the global model, $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\gamma^{(t)}}{q} \phi^{-1}\left(\sum_{i \in \mathcal{S}} \overline{\mathbf{w}}_i^{(t)}\right)$.

---

samples. Honest users utilize the ADAM optimizer [42] to update the local update by setting the size of the local mini-batch sample to $|\xi_i^{(t)}| = 500$ for all $i \in [N], t \in [J]$ where J is the total number of iterations. Byzantine users generate vectors uniformly at random from $\mathbb{F}_p^d$ where we set the field size $p = 2^{32} - 5$, which is the largest prime within 32 bits. For both schemes, BREA and FedAvg, the number of models to be aggregated is set to $m = |\mathcal{S}| = 13 < N - 2A - 2$. FedAvg randomly selects m models at each iteration while BREA selects m users from (24).

Convergence and robustness against Byzantine users: Figure 2 shows the test accuracy of BREA and FedAvg for different number of Byzantine users. We can observe that BREA with 0% and 30% Byzantine users is as efficient as FedAvg with 0% Byzantine users, while FedAvg does not tolerate Byzantine users. Figure 3 presents the cross entropy loss for BREA versus FedAvg for different number of Byzantine users. We omit the FedAvg with 30% Byzantine users as it diverges. We observe that BREA with 30% Byzantine users achieves convergence with comparable rate to FedAvg
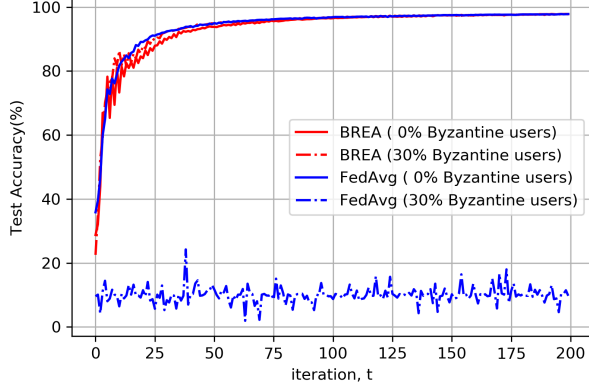
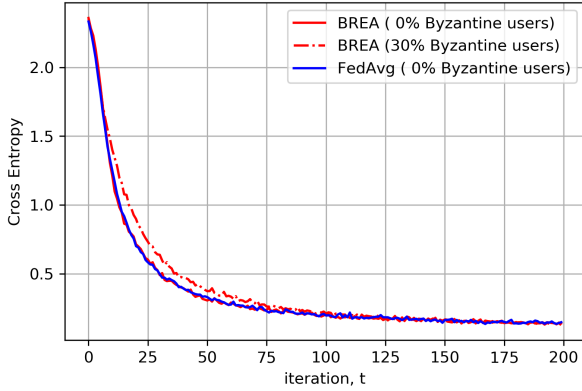Fig. 2. Test accuracy of BREA and FedAvg [1] for different number of Byzantine users with i.i.d. MNIST dataset.

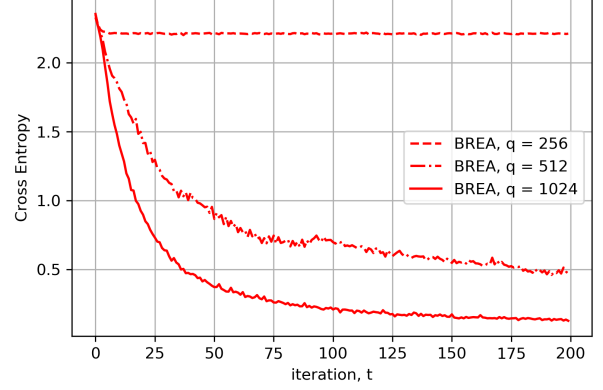Fig. 4. Convergence of BREA for different values of the quantization parameter q in (9) with 30% Byzantine users and i.i.d. MNIST dataset.
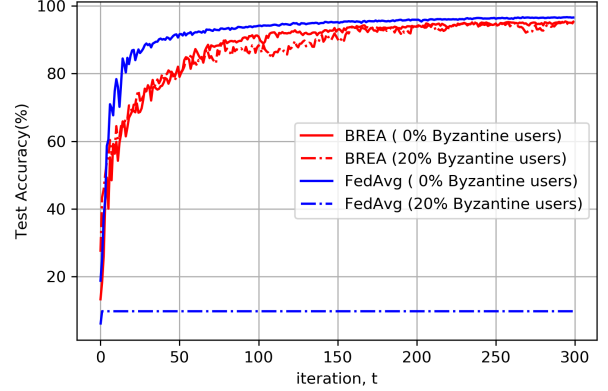

Fig. 3. Convergence of BREA and FedAvg [1] for different number of Byzantine users with i.i.d. MNIST dataset.

Fig. 5. Test accuracy of BREA and FedAvg [1] for different number of Byzantine users with non-i.i.d MNIST dataset.

with 0% Byzantine users, while providing robustness against Byzantine users and being privacy-preserving. For all cases of BREA in Figures 2 and 3, we set the quantization value in (9) to $q = 1024$. Figure 4 further illustrates the cross entropy loss of BREA for different values of quantization parameter q. We can observe that BREA with a larger value of q has better performance because the variance caused by the quantization function Q_q defined in (9) gets smaller as q increases. On the other hand, given the field size p, the quantization parameter q should be less than a certain threshold to ensure (23) holds.

Non-i.i.d dataset: To investigate the performance of BREA with non-i.i.d dataset, we partition the MNIST data over $N = 100$ users by using the same non-i.i.d setting of [1]. In this setting, 60000 training samples are sorted by their digit labels and partitioned into 200 shards of size 300. Then, $N = 100$ users are considered, where each user receives two shards with different digit labels. We assume $T = 20$ users may collude and consider two cases for the number of Byzantine users: i) 0% Byzantine users ($A = 0$) and ii) 20% Byzantine users ($A = 20$). Honest users utilize the ADAM optimizer to update the model by setting the size of the local mini-batch sample to $|\xi_i^{(t)}| = 10$ and local iterations to 5 (i.e., each user locally updates the model 5 times before sending to the server

at each iteration). Byzantine users generate vectors uniformly at random from $\mathbb{F}_p^d$. For both schemes, BREA and FedAvg, the number of models to be aggregated is set to $m = 50$. Figure 5 shows the test accuracy of BREA and FedAvg for different number of Byzantine users. Even for the non-i.i.d. dataset, BREA with 0% and 20% Byzantine users empirically demonstrates comparable test accuracy to FedAvg with 0% Byzantine users.

CIFAR-10 dataset: To investigate the performance of BREA with a larger dataset, we additionally experiment with CIFAR-10 [43], by using the same model architecture of [3] (about 10^6 model parameters). We use the same setting for parameters N, A, m, p as the setting of i.i.d MNIST dataset. Figure 6 shows that the BREA algorithm (with 30% Byzantine users) has comparable test accuracy to the FedAvg algorithm (with no Byzantine users).

VIII. CONCLUSION

This paper presents the first single-server solution for Byzantine-resilient secure federated learning. Our framework is based on a verifiable secure outlier detection strategy to guarantee robustness of the trained model against Byzantine faults, while protecting the privacy of the individual users. We
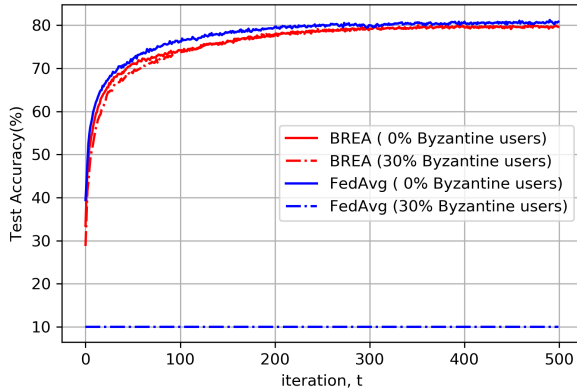
Fig. 6. Test accuracy of BREA and FedAvg [1] for different number of Byzantine users with CIFAR-10 dataset.

provide the theoretical convergence guarantees and the fundamental performance trade-offs of our framework, in terms of the number of Byzantine adversaries and the user dropouts the system can tolerate. In our experiments, we have implemented our system in a distributed network by using two ways of partitioning the MNIST dataset over up to $N = 100$ users: i.i.d. and non-i.i.d. data distribution. For both settings, we numerically demonstrated the convergence behaviour while providing robustness against Byzantine users and being privacy-preserving. Future directions include developing single-server Byzantine-resilient secure learning architectures considering the heterogeneous environments in terms of computation and communication resources, developing efficient communication architectures, and quantization techniques.

In this paper, we utilize the distance based outlier detection approach to guarantee the robustness against Byzantine users. There have been many outlier detection approaches for Byzantine resilient SGD methods (or federated learning). Existing works can be roughly classified into two main categories: 1) geometric median [9], [10] and 2) coordinate-wise median [11]–[13]. Our framework utilizes the geometric median based approach because it is more efficient in the secure domain. In order to preserve the privacy of local updates, the coordinate-wise median based approaches require a multiparty secure comparison protocol [44] to find the median value of each coordinate. On the other hand, in our framework, the server compares the pairwise distances in plaintext, which is much more efficient than the secure comparison protocol. Extending the coordinate-wise median based algorithms to the secure domain would be a very interesting future direction.

Acknowledgement

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 54, Fort Lauderdale, FL, USA, Apr 2017, pp. 1273–1282.

[2] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *ACM SIGSAC Conf. on Comp. and Comm. Security*.   ACM, 2017, pp. 1175–1191.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Int. Conf. on Artificial Int. and Stat. (AISTATS)*, 2017, pp. 1273–1282.

[4] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," in *2nd SysML Conf.*, 2019.

[5] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[6] J. So, B. Guler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," *arXiv preprint arXiv:2002.04156*, 2020.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[8] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[9] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 119–129.

[10] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[11] 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*, vol. 80, Stockholm Sweden, 10–15 Jul 2018, pp. 5650–5659.

[12] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2018, pp. 4613–4623.

[13] H. Yang, X. Zhang, M. Fang, and J. Liu, "Byzantine-resilient stochastic gradient descent for distributed learning: A lipschitz-inspired coordinate-wise median approach," in *2019 IEEE 58th Conference on Decision and Control (CDC)*.   IEEE, 2019, pp. 5832–5837.

[14] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 774–14 784.

[15] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE Conference on Computer Communications (INFOCOM'2019)*, 2019, pp. 2512–2520.

[16] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients – How easy is it to break privacy in federated learning?" *arXiv preprint arXiv:2003.14053*, 2020.

[17] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *28th Annual Symposium on Foundations of Computer Science*.   IEEE, 1987, pp. 427–438.

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

[19] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1544–1551.

[20] D. Data and S. Diggavi, "Byzantine-resilient sgd in high dimensions on heterogeneous data," *arXiv preprint arXiv:2005.07866*, 2020.

[21] G. Ács and C. Castelluccia, "I have a dream! (differentially private smart metering)," in *International Workshop on Information Hiding*. Springer, 2011, pp. 118–132.

[22] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. on Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[23] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Conference on Neural Information Processing Systems: Workshop on Private Multi-Party Machine Learning*, 2016.

[24] K. Bonawitz, F. Salehi, J. Konečnỳ, B. McMahan, and M. Gruteser, "Federated learning with autotuned communication-efficient secure aggregation," *arXiv preprint arXiv:1912.00131*, 2019.

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

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

[27] C. Dwork, "Differential privacy: A survey of results," in *International conference on theory and applications of models of computation*. Springer, 2008, pp. 1–19.

[28] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," *arXiv preprint arXiv:1902.00146*, 2019.

[29] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," *arXiv preprint arXiv:1905.10497*, 2019.

[30] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[31] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," *arXiv preprint arXiv:1811.12470*, 2018.

[32] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," *arXiv preprint arXiv:1911.11815*, 2019.

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

[34] J. Bell, K. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly)logarithmic overhead," *IACR Cryptol. ePrint Arch.*, 2020. [Online]. Available: https://eprint.iacr.org/2020/704

[35] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1709–1720.

[36] D. Dolev and H. R. Strong, "Authenticated algorithms for Byzantine agreement," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 656–666, 1983.

[37] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed systems: concepts and design*. pearson education, 2005.

[38] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Byzantine-tolerant machine learning," *arXiv preprint arXiv:1703.02757*, 2017.

[39] S. Gao, "A new algorithm for decoding reed-solomon codes," in *Communications, information and network security*. Springer, 2003, pp. 55–68.

[40] L. Bottou, "Online learning and stochastic approximations," *Online learning in neural networks*, vol. 17, no. 9, p. 142, 1998.

[41] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *[Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[43] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[44] F. Kerschbaum, D. Biswas, and S. de Hoogh, "Performance comparison of secure comparison protocols," in *2009 20th International Workshop on Database and Expert Systems Application*. IEEE, 2009, pp. 133–136.

APPENDIX

A. The BREA Framework with Local Updating Schemes

In this section, we present how BREA framework can work with local updating schemes where each user locally takes multiple-steps of stochastic gradient descent using its local data before sending the updated model to server.

The BREA framework with local updating schemes follows the same steps from Section V-A to Section V-E by changing the local update $\mathbf{w}_i^{(t)}$ of user i from a gradient estimate to the updated model by using multiple-steps of stochastic gradient descent (SGD). Then, the update equation of a benign user i can be expressed as follows,

$$\mathbf{v}_i^{(t+1)} = \mathbf{w}_i^{(t)} - \gamma_t g(\mathbf{w}_i^{(t)}, \xi_i^{(t)}) \tag{54}$$

$$\mathbf{w}_i^{(t+1)} = \begin{cases} \frac{1}{m} \sum_{i \in \mathcal{S}^{(t+1)}} \mathbf{v}_i^{(t+1)} & \text{if } t+1 \in \mathcal{J}_E \\ \mathbf{v}_i^{(t+1)} & \text{if } t+1 \notin \mathcal{J}_E \end{cases} \tag{55}$$

where $\mathcal{J}_E$ denotes the set of global synchronization steps, i.e., $\mathcal{J}_E = \{nE \mid n = 1, 2, \cdots\}$, E is the number of local iterations, and $\mathcal{S}^{(t+1)}$ is the index set of selected users at iteration $t+1$ by utilizing the multi-Krum algorithm described in Section V-D. An additional vector $\mathbf{v}_i^{(t+1)}$ is introduced to represent the immediate result of one step SGD update from $\mathbf{w}_i^{(t)}$. Malicious user i can send any random vector $\mathbf{b}_i^{(t+1)} \in \mathbb{R}^d$, which we represent as $\mathbf{v}_i^{(t+1)} = \mathbf{b}_i^{(t+1)}$. We note that the distance-based outlier detection algorithm of which inputs are the local models instead of local gradient works effectively as the distance between the local models is equal to the multiplication of the learning rate γ and the distance between the local gradients. Consequently, the aggregation step at the server side can be represented as

$$\mathbf{w}^{(t)} = \sum_{i \in \mathcal{S}^{(t)}} \mathbf{w}_i^{(t)} \quad \text{if } t \in \mathcal{J}_E. \tag{56}$$

Jinhyun So received the B.S. and M.S. degrees in electrical and computer engineering from KAIST. He is currently pursuing the Ph.D. degree in electrical and computer engineering at University of Southern California (USC). He received the Annenberg Graduate Fellowship in 2017. His research interests include information theory, large-scale distributed machine learning, and secure and private computing.

Başak Güler is an Assistant Professor in the Department of Electrical and Computer Engineering at University of California, Riverside. She has received her PhD from the Pennsylvania State university in 2017, and was a postdoctoral scholar at University of Southern California between 2018-2020. Her research interests include machine learning in wireless networks, information theory, distributed computing, and signal processing.

A. Salman Avestimehr is a Professor and director of the Information Theory and Machine Learning (vITAL) research lab at the Electrical and Computer Engineering Department of University of Southern California. He received his Ph.D. in 2008 and M.S. degree in 2005 in Electrical Engineering and Computer Science, both from the University of California, Berkeley. Prior to that, he obtained his B.S. in Electrical Engineering from Sharif University of Technology in 2003. His research interests include information theory and coding theory, and large-scale distributed computing and machine learning, secure and private computing, and blockchain systems.

Dr. Avestimehr has received a number of awards for his research, including the James L. Massey Research & Teaching Award from IEEE Information Theory Society, an Information Theory Society and Communication Society Joint Paper Award, a Presidential Early Career Award for Scientists and Engineers (PECASE) from the White House (President Obama), a Young Investigator Program (YIP) award from the U. S. Air Force Office of Scientific Research, a National Science Foundation CAREER award, the David J. Sakrison Memorial Prize, and several Best Paper Awards at Conferences. He has been an Associate Editor for IEEE Transactions on Information Theory and a general Co-Chair of the 2020 International Symposium on Information Theory (ISIT). He is a fellow of IEEE.