Communication-Efficient Secure Aggregation for Federated Learning

Irem Ergün*
Computer Science and Engineering
University of California, Riverside
Riverside, CA
iergu001@ucr.edu

Hasin Us Sami*
Electrical and Computer Engineering
University of California, Riverside
Riverside, CA
hsami003@ucr.edu

Başak Güler
Electrical and Computer Engineering
University of California, Riverside
Riverside, CA
bguler@ece.ucr.edu

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001481

***Abstract*—Secure aggregation is a privacy-aware protocol for model aggregation in federated learning. A major challenge of conventional secure aggregation protocols is their large communication overhead. Towards addressing this challenge, in this work we propose the first gradient sparsification framework for communication-efficient secure aggregation, which allows aggregation of sparsified local gradients from a large number of users, without revealing the individual local gradient parameters in the clear. We provide the theoretical performance guarantees of the proposed framework in terms of the communication efficiency, resilience to user dropouts, and model convergence. We further evaluate the performance of our framework through large-scale experiments in a distributed network with up to 100 users, and demonstrate a significant reduction in the communication overhead compared to conventional secure aggregation benchmarks.**

I. Introduction

Federated learning is a distributed training framework to train machine learning models over the data stored at a large number of wireless devices (users). In the conventional federated learning architecture proposed in [1], training is carried out in an iterative manner governed by a central server, who holds a global model. At each iteration, the server sends the current state of the global model to the users, who then update the global model by training locally on their datasets, generating a local model. The local models are sent from the users to the server, who then aggregates (sums) the local models to update the global model. Finally, the updated global model is sent back to the users for the next training round. Various generalizations of this initial architecture have been proposed in the literature, including decentralized or asynchronous federated learning [2].

Among the most distinctive features of federated learning is its *on-device* learning architecture, i.e., training is performed locally on the local dataset of the device. Due to this on-device learning architecture (data never leaves the device), federated learning is widely used in a variety of privacy-sensitive applications [3]–[6]. On the other hand, recent works have shown that the local models can still reveal privacy-sensitive information about the training data of the users by incorporating model inversion techniques [7]–[10].

Secure aggregation protocols have emerged as a counter-measure against such privacy threats, by enabling the server to *aggregate* the local models of millions of users, without observing the local models in the clear [11]–[13]. This is done by a process known as pairwise masking, where a random mask is generated upon agreement between each pair of users, and users mask their local models by combining them with the pairwise random masks. Users then only share the *masked local models* with the server, which hides the actual contents of the local models from the server. Once the masked models are aggregated at the server, the pairwise masks cancel out, allowing the server to learn the aggregate of the local models. On the other hand, no further information (in an information-theoretic sense) is revealed about the local models beyond their aggregate. Secure aggregation is complementary to and can be combined with other privacy-preserving computing techiques such as differential privacy [14].

A major challenge against the scalability of secure aggregation protocols to large networks is their communication overhead. Gradient sparsification is a popular approach to reduce the communication overhead in conventional (non-private) distributed learning, where each user only sends a small fraction of the model parameters to the server, instead of the full set of model parameters [15]–[19]. On the other hand, conventional gradient sparsification techniques can not be readily applied to secure aggregation. This is due to the fact that the locations of the sparsified gradient parameters often vary significantly from one user to another. As we demonstrate later, this prevents the pairwise masks from being cancelled out upon aggregation of the masked local models at the server.

Towards addressing this challenge, in this work we introduce a novel gradient sparsification framework for secure aggregation, `SparseSecAgg`, which securely aggregates a fraction of $\alpha \in (0, 1]$ model parameters from each user, allowing the server to learn the aggregate of the sparsified local models, but without learning their true values. To the best of our knowledge, `SparseSecAgg` is the first gradient sparsification framework that is compatible with the underlying cryptographic primitives of secure aggregation. We provide the theoretical performance guarantees of `SparseSecAgg` for the commu-

*Equal contribution.

Research was sponsored in part by the OUSD (R&E)/RT&L under Cooperative Agreement Number W911NF-20-2-0267, NSF CAREER Award CCF-2144927, and the UC Regents Faculty Fellowship. The views and conclusions are those of the authors.

nication overhead, resilience to user dropouts, and model convergence. We further perform extensive experiments for image classification in a network of up to 100 users on CIFAR-10 and MNIST datasets, and demonstrate up to 7.6× reduction in the communication overhead. Our results demonstrate that the performance improvement over the benchmarks increases as the number of users increase, which further motivates the use of `SparseSecAgg` in large-scale applications.

II. Overview

A. Federated Learning

We consider a cross-device federated learning architecture consisting of a server and N users, where user $i \in [N]$ has a local dataset $\mathcal{D}_i$ [1]. The goal is to train a model $\mathbf{w} \in \mathbb{R}^d$ of size d, to minimize a global loss function $F(\mathbf{w})$,

$$\min_{\mathbf{w}} F(\mathbf{w}) \text{ s.t. } F(\mathbf{w}) = \sum_{i\in[N]} \beta_i F_i(\mathbf{w}), \tag{1}$$

where F_i represents the local loss function of user i and β_i represents a weight parameter assigned to user i, usually selected as $\beta_i = \frac{|\mathcal{D}_i|}{\sum_{i\in[N]}|\mathcal{D}_i|}$ [2].

Training is carried out in an iterative manner. At iteration t, users receive the current state of the global model, represented by $\mathbf{w}^{(t)}$, from the server. User $i \in [N]$ trains this received global model locally on the local dataset through multiple stochastic gradient descent steps,

$$\mathbf{w}_i^{(t,j+1)} = \mathbf{w}_i^{(t,j)} - \eta^{(t,j)} \nabla F_i(\mathbf{w}_i^{(t,j)}) \tag{2}$$

for $j = 0, \dots, E-1$, where E represents the number of local training epochs, $\mathbf{w}_i^{(t,0)} \triangleq \mathbf{w}^{(t)}$ and $\eta^{(t,j)}$ is the learning rate. After E local training epochs, user i forms a local model,

$$\mathbf{w}_i^{(t)} := \mathbf{w}_i^{(t,E)} = \mathbf{w}^{(t)} - \sum_{j=0}^{E-1} \eta^{(t,j)} \nabla F_i(\mathbf{w}_i^{(t,j)}) \tag{3}$$

and sends it to the server. Alternatively, instead of sending the local model $\mathbf{w}_i^{(t)}$, user i can send the scaled local gradient,

$$\mathbf{y}_i^{(t)} := \sum_{j=0}^{E-1} \eta^{(t,j)} \nabla F_i(\mathbf{w}_i^{(t,j)}) \tag{4}$$

to the server, since one can be computed from the other. Finally, the server aggregates the local models and updates the global model for the next iteration,

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

B. Secure Aggregation

Secure aggregation (`SecAgg`) protocols utilize secure multi-party computation principles [20] for model aggregation from a large number of users without letting the server learn the individual models in the clear [11]. These protocols are mainly based on a primitive known as *pairwise masking*, where each pair of users $i, j \in [N]$ first agree on a pairwise random seed $s_{ij}^{(t)}$ (unknown to the other users and the server) using a Diffie-Hellman type key exchange protocol [21]. User i then secret shares $\{s_{ij}^{(t)}\}_{j\in[N]}$ with every other user, using Shamir's $\frac{N}{2}$-out-of-N secret sharing protocol [22]. Secret sharing allows each $s_{ij}^{(t)}$ to be reconstructed by collecting any set of least $\frac{N}{2}+1$ shares, but any set of less than $\frac{N}{2}+1$ shares reveals no information (in an information-theoretic sense) about $s_{ij}^{(t)}$.

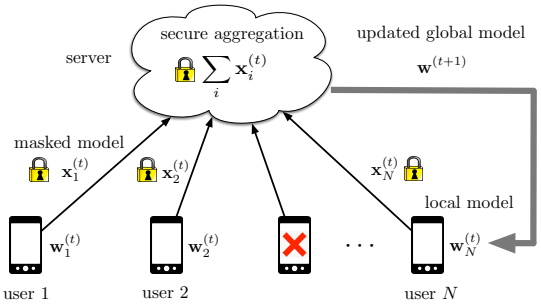


Fig. 1. Secure aggregation in federated learning. At each iteration, the server sends the current state of the global model to the users, who then perform local training on it using their local datasets, and send a masked model to the server. The server aggregates the masked models and learns the aggregate of the true models, which is then used to update the global model.

During training, user i expands each pairwise seed to a random vector of size d using a pseudorandom generator (PRG), and creates a *masked* version of its local model,

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

and sends this masked local model to the server. During this process, some users may drop out from the network due to poor channel quality or low battery, and fail to send (6) to the server. We let $\mathcal{D}^{(t)}$ and $\mathcal{S}^{(t)} = [N] \backslash \mathcal{D}^{(t)}$ denote the set of dropout and surviving users respectively at iteration t.

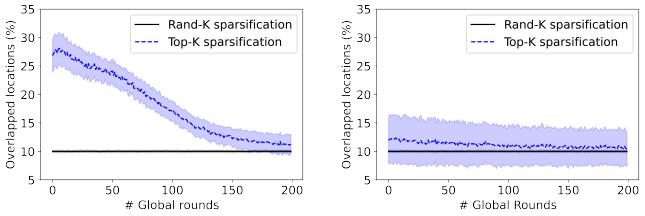
Upon receiving the masked models from the surviving users, the server computes their aggregate, $\sum_{i\in\mathcal{S}^{(t)}} \mathbf{x}_i^{(t)}$. Then, the server reconstructs the pairwise seeds corresponding to the dropped users (which do not cancel out upon aggregation), by collecting their secret shares (from the surviving users), and removes the corresponding random vectors from the aggregate,

$$\sum_{i\in\mathcal{S}^{(t)}} \mathbf{x}_i^{(t)} - \sum_{i\in\mathcal{D}^{(t)}} \Big(\sum_{j:i<j} \text{PRG}(s_{ij}^{(t)}) - \sum_{j:i>j} \text{PRG}(s_{ji}^{(t)}) \Big) = \sum_{i\in\mathcal{S}^{(t)}} \mathbf{w}_i^{(t)} \tag{7}$$

to learn the aggregate of the true local models of all surviving users. All operations are carried out in a finite field $\mathbb{F}_q$ of integers modulo prime q. This process is illustrated in Fig. 1.

III. System Model

The large communication overhead of sending the local models in (5) from the users to the server poses a major challenge in large-scale applications, where N and d can be in the order of millions. Gradient sparsification is a popular approach to address this challenge in distributed learning [15]–[19]. The most common sparsification techniques are random-K (rand-K) and top-K sparsification, where users select K parameters from their local gradients either randomly or in a magnitude-based manner, and send the corresponding parameters along with the location indices (coordinates) to the



(a) MNIST IID (b) MNIST nonIID

Fig. 2. Average percentage of overlappping gradient coordinates between each pair of users for rand-K and top-K sparsification, respectively. Under the non-IID data distribution, for both rand-K and top-K, the overlap between the selected coordinates is only around 10%. For the IID setting with top-K, the overlap starts around 30% and decreases to 12% throughout the training.

server. The server then aggregates the parameters according to their locations, and updates the global model.

However, none of these sparsification techniques can be applied to secure aggregation, as the coordinates of the K parameters often vary from one user to another. We illustrate this in Fig. 2, where we report the overlapping gradient coordinates in federated learning for an image classification task with rand-K and top-K gradient sparsification, for $K = 0.1d$ and $N = 30$ users, using the MNIST dataset [23]. The solid line represents the mean percentage of the overlapping gradient coordinates among each pair of users and the shaded areas represent one standard deviation from the mean. We consider both IID and non-IID data distributions across the users according to [1]. We observe that under the non-IID setting, for both rand-K and top-K, the overlap among the selected gradient coordinates of the users is only around 10%. For the IID setting with top-K, the overlap starts at around 30% and decreases to 12% as training progresses.

As a result, both rand-K or top-K gradient sparsification, if applied to `SecAgg` (from Section II-B), will not ensure cancellation of the pairwise random vectors upon aggregation, requiring the server to reconstruct all of the pairwise seeds. Doing so will allow the server to eliminate the random vectors in (6), revealing the true content of the local gradients to the server, violating the core principle of secure aggregation. One could instead apply secure aggregation to the union of rand-K or top-K coordinates of *all users*. Doing so, however, is highly inefficient especially under heterogeneous (non-iid) data distributions. In the worst case, this leads to the number of model parameters sent from each user to scale with respect to the number of users ($\min\{NK, d\}$), as opposed to K parameters. Towards addressing this challenge, in this work we introduce `SparseSecAgg`, a novel gradient sparsification framework for secure aggregation, where the server learns the aggregate of *sparsified local gradients* from a large number of users, but without learning the individual gradient parameters. `SparseSecAgg` reduces the communication overhead by aggregating, instead of the entire gradient, only a small fraction of the local gradient parameters from each user, while ensuring privacy of individual users and convergence of training.

Threat model. As in conventional secure aggregation [11], [13], we consider an honest-but-curious adversary model, where adversaries follow the protocol truthfully, but try to extract additional information about the local models of honest users from the messages exchanged during the protocol. Out of N users, up to $A \leq \gamma N$ users are adversarial for some $\gamma \in (0, 0.5)$, who may collude with each other and/or the server. We evaluate the performance of `SparseSecAgg` according to the following key performance metrics:

Compression ratio: The compression ratio $\alpha \in (0, 1]$ is defined as the fraction of the entire gradient vector each user sends to the server (each user sends αd masked parameters, as opposed to the full gradient of size d), with probability approaching 1 as $d \to \infty$. A smaller α reduces the communication overhead, but it may also slow down the convergence.

Aggregation Cardinality: The aggregation cardinality, T, quantifies the number of honest users participating at any given location of the global model, with probability approaching to 1 as $N \to \infty$. In the context of secure aggregation, a larger value of T leads to better privacy [24].

IV. The SparseSecAgg Framework

We now provide the details of `SparseSecAgg`. For the sake of brevity, we describe the process for one training round and omit the round index t.

A. Random Mask Generation

Pairwise additive masks. `SparseSecAgg` uses additive random masks to hide the true content of the local gradient updates from the server during aggregation. To do so, each pair of users $i, j \in [N]$ first agree on a pairwise random seed s_{ij} via a Diffie-Hellman key agreement protocol [21]. Using s_{ij}, users i, j then construct a random vector,

$$\mathbf{r}_{ij} = \text{PRG}(s_{ij}) \in \mathbb{F}_q^d. \tag{8}$$

where each element is generated uniformly random from $\mathbb{F}_q$.

Pairwise binary masks. `SparseSecAgg` utilizes pairwise binary masks to construct the sparsification pattern of the local model updates. To do so, each pair of users $i, j \in [N]$ agree on a random binary vector $\mathbf{b}_{ij} \in \{0, 1\}^d$, where each element $\ell \in [d]$ is drawn IID from a Bernoulli distribution:

$$\mathbf{b}_{ij}(\ell) = \begin{cases} 1 & \text{with probability } \frac{\alpha}{N-1} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

for a given $\alpha \in (0, 1]$. Parameter α quantifies the number of parameters sent from each user (accordingly, the communication overhead) as detailed in Section V.

To generate the binary vectors, first, another instantiation of the process described above for additive mask generation is carried out, where a vector of size d is generated uniformly at random from $\mathbb{F}_q$. Then, the domain of the PRG is divided into two intervals, where the size of the intervals are proportional to $\frac{\alpha}{N-1}$ and $1 - \frac{\alpha}{N-1}$, respectively. The numbers that are in the first interval are mapped to 1 whereas the others are mapped to 0. In doing so, each pair of users $i, j \in [N]$ can agree on a binary vector $\mathbf{b}_{ij} = \mathbf{b}_{ji} \in \{0, 1\}^d$. As we show next, the binary masks specify the parameter coordinates selected by the users to be sent to the server, and ensure that the additive masks cancel out upon the aggregation of sparsified gradients.

Secret sharing. Users secret share the seeds of their pairwise binary and additive masks with every other user, using Shamir's $\frac{N}{2}$-out-of-N secret sharing [22], which embeds each random seed in a random polynomial of degree $\frac{N}{2}$ over the finite field $\mathbb{F}_q$. As we show next, if any user drops out during gradient aggregation, the secret shares are used to recover the pairwise binary and additive seeds belonging to those users.

B. Mapping from Real Numbers to Finite Field

Users then convert their local gradients from the domain of real numbers to the finite field $\mathbb{F}_q$. This is handled by a stochastic rounding function $Q_c(z)$ where $Q_c(z) = \lfloor cz \rfloor / c$ with probability $1 - (cz - \lfloor cz \rfloor)$, and $Q_c(z) = (\lfloor cz \rfloor + 1)/c$ otherwise, where $\lfloor z \rfloor$ is the largest integer that is less than or equal to z and c is a parameter that determines the quantization level. Next, user i forms a quantized local gradient as follows,

$$\overline{\mathbf{y}}_i = \phi\Big(c \cdot Q_c\Big(\beta_i \mathbf{y}_i\Big)\Big), \tag{10}$$

where the function $\phi : \mathbb{R} \to \mathbb{F}_q$ is given as,

$$\phi(z) = \begin{cases} z & \text{if } z \geq 0 \\ q + z & \text{if } z < 0 \end{cases} \tag{11}$$

to represent the positive and negative numbers using the first and second half of the finite field, respectively. Functions $Q_c(.)$ and $\phi(.)$ are applied element-wise in (10).

C. Sparsified Gradient Construction

Using the pairwise binary and additive masks, user $i \in [N]$ forms their sparsified masked gradient $\mathbf{x}_i$, where

$$\mathbf{x}_i(\ell) = \Big(1 - \prod_{j \in [N]: j \neq i} (1 - \mathbf{b}_{ij}(\ell))\Big)(\overline{\mathbf{y}}_i(\ell)) + \sum_{j \in [N]: i < j} \mathbf{b}_{ij}(\ell)\mathbf{r}_{ij}(\ell) - \sum_{j \in [N]: i > j} \mathbf{b}_{ij}(\ell)\mathbf{r}_{ij}(\ell) \tag{12}$$

for $\ell \in [d]$. Specifically, for each non-zero element of $\mathbf{b}_{ij}(\ell)$, user i subtracts $\mathbf{r}_{ij}(\ell)$ from its quantized gradient $\overline{\mathbf{y}}_i(\ell)$ if $i > j$, and adds it if $i < j$. The key intuition behind (12) is that the sparsified pairwise additive masks cancel out upon aggregation at the server. Next, we define a set $\mathcal{U}_i$ such that,

$$\mathcal{U}_i = \{\ell : \mathbf{b}_{ij}(\ell) = 1 \text{ for some } j \in [N] \backslash \{i\}, \ell \in [d]\}, \tag{13}$$

which contains the coordinates of the gradient parameters selected by user i. User i then sends all $\mathbf{x}_i(\ell)$ for which $\ell \in \mathcal{U}_i$, along with $\mathcal{U}_i$ (represented as a vector), to the server.

D. Secure Aggregation of Sparsified Gradients

Next, the server aggregates the sparsified masked gradients received from the surviving users,

$$\mathbf{x} := \sum_{i \in \mathcal{S}} \mathbf{x}_i \tag{14}$$

Note that the pairwise additive masks corresponding to the dropout users will not be cancelled out in (14), which prevents the server from learning the true sum of the local gradients. To handle this, from the surviving users, the server requests the secret shares of the pairwise seeds corresponding to dropout users. Upon receiving a sufficient number of (i.e., $N/2 + 1$) secret shares from the surviving users, the server reconstructs the pairwise binary and additive masks of the dropped users, and removes them from the sum of the masked gradients,

$$\begin{aligned} \overline{\mathbf{y}}(\ell) &\triangleq \mathbf{x}(\ell) - \sum_{i \in \mathcal{D}} \sum_{\substack{j: i < j \\ j \in [N] \backslash \mathcal{D}}} \mathbf{r}_{ij}(\ell) \mathbb{1}_i^\ell + \sum_{i \in \mathcal{D}} \sum_{\substack{j: i > j \\ j \in [N] \backslash \mathcal{D}}} \mathbf{r}_{ij}(\ell) \mathbb{1}_i^\ell \\ &= \sum_{i \in \mathcal{S}} \overline{\mathbf{y}}_i(\ell) \end{aligned} \tag{15}$$

to learn the sum of the true gradients of all surviving users, where $\mathbb{1}_i^\ell$ is an indicator random variable that is equal to 1 if and only if $\ell \subseteq \mathcal{U}_i$, and 0 otherwise.

Then, the sum of the quantized gradients from (15) are mapped from the finite field back to the real domain,

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{1}{c} \cdot \phi^{-1}(\overline{\mathbf{y}}), \tag{16}$$

where ϕ^{-1} is applied element-wise to $\overline{\mathbf{y}}$. Finally, the server sends the new global model, $\mathbf{w}$, to the users for the next round.

V. Theoretical Results

We now provide the theoretical performance guarantees of `SparseSecAgg`. according to the key performance metrics from Section III.

Theorem 1 (Compression ratio)**.** *`SparseSecAgg` guarantees a compression ratio of α as $d \to \infty$, in other words, each user sends αd model parameters to the server.*

Proof (Sketch). The proof follows from Hoeffding-Chernoff bounds for the tail probabilities of the sum of independent random variables for the proposed pairwise sparsification framework from Section IV. Due to space constraints, the detailed steps of the proof are provided in [25].

Theorem 2 (Aggregation Cardinality)**.** *In a system of N users where each user drops from the system with probability $\theta \in [0, 0.5)$, `SparseSecAgg` provides an aggregation cardinality of $T = (1 - e^{-\alpha})(1 - \theta)(1 - \gamma)N$ as the number of users $N \to \infty$. For $\alpha \ll 1$, the aggregation cardinality approaches $T = \alpha(1 - \theta)(1 - \gamma)N$.*

Proof (Sketch). The proof follows from a relation between sampling with and without replacement for tail probabilities, and the detailed steps are provided in [25].

Corollary 1 (Robustness to user dropouts)**.** *`SparseSecAgg` is robust to up to a user dropout rate of $\theta < 0.5$ as $N \to \infty$.*

Proof. This result follows directly from Shamir's $\frac{N}{2}$-out-of-N secret sharing [22], which ensures that as long as there are at least $\frac{N}{2} + 1$ surviving users, the server can collect the secret shares to reconstruct the pairwise seeds of the dropout users, reconstruct the corresponding binary and additive masks, and remove them from the aggregated gradients. Finally, with a dropout rate $\theta < 0.5$, the number of surviving users approaches $\frac{N}{2} + 1$ as $N \to \infty$. □

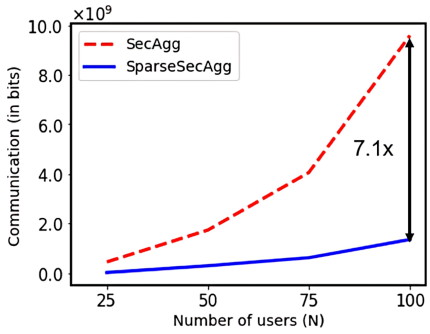


Fig. 3. Total communication overhead to reach the target test accuracy for the CIFAR-10 dataset under the IID setting.

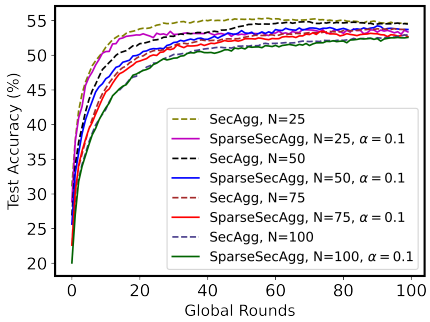


Fig. 4. Test accuracy vs number of global training rounds for the CIFAR-10 dataset under the IID setting.

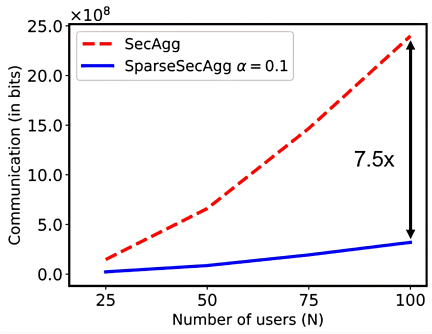


Fig. 5. Total communication overhead to reach the target test accuracy for the MNIST dataset distributed IID across the users.

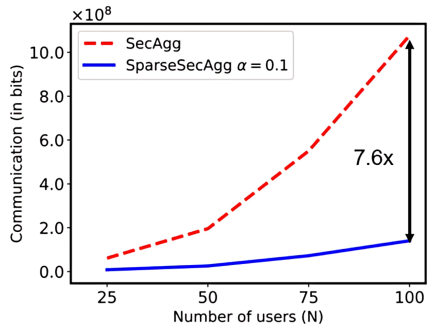


Fig. 6. Total communication overhead to reach the target test accuracy for the MNIST dataset distributed non-IID across the users.

We note that in real-world settings, the drop-out rates vary between 0.06 and 0.1 [26]. For completeness, we also provide the convergence guarantees of training in [25].

VI. Experiments

In our experiments, we compare the performance of `SparseSecAgg` with the conventional secure aggregation benchmark from [11], termed `SecAgg`, in terms of the communication overhead to reach convergence.

Setup. We consider image classification tasks on the CIFAR-10 [27] and MNIST [23] datasets, using the CNN architectures from [1]. The compression ratio is set to $\alpha = 0.1$, and the dropout rate is set to $\theta = 0.3$, to evaluate the performance under severe network conditions. We then compare the performance of `SparseSecAgg` versus `SecAgg` to reach a target test accuracy. Note that both schemes guarantee unbiased training, hence can reach the same highest accuracy, but potentially in different number of training rounds. In particular, due to sparsification, `SparseSecAgg` may take longer to reach the target accuracy. In the following, we show that even in such scenarios, `SparseSecAgg` provides significant benefits in reducing the communication overhead. Specifically, `SparseSecAgg` reduces not only the communication overhead per training round (which is our main goal, to enable secure aggregation in bandwidth limited environments), but also the total communication overhead throughout the entire training (i.e., to reach the target test accuracy). We set the size of the finite field to $q = 2^{32} - 5$, which is the largest prime within 32 bits.

Performance evaluation. We compare the performance of `SparseSecAgg` in terms of both the communication overhead per training round, and the total communication overhead (over the entire network) to reach the target test accuracy. We consider $N = 25,\ 50,\ 75,\ 100$ users, where the local datasets are distributed under both IID and non-IID settings from [1]. For the IID setting, the dataset is shuffled and distributed uniformly across N users. For the non-IID setting, the dataset is first sorted according to the labels of each data point, and then divided into 300 shards, each shard containing samples from at most two classes. Then, each user is randomly given $300/N$ shards. Hence, the local dataset sizes of the users are the same in both the IID and non-IID cases under the same number of users. For both frameworks, the number of local training epochs is E=5, batch size is 28, and the momentum parameter is 0.5. The learning rate is set to 0.01.

We first consider CIFAR-10 training, where the target test accuracy is 52% using the CNN architecture from [1] under the IID setting. In Table I, we report the communication overhead per user per training round for `SparseSecAgg` versus `SecAgg`. For `SparseSecAgg`, we report the maximum (worst-case) across all users and training rounds. We observe that the per-user communication overhead of `SparseSecAgg` at any given training round is 7.7× smaller than `SecAgg`. This is consistent with our theoretical findings, and the small increase in communication is due to sending the coordinates of the model parameters to the server. We use 32 bits to represent each parameter, and one bit per parameter coordinate (which represents whether a given parameter is selected).

In Figure 3, we demonstrate the total communication

TABLE I
COMMUNICATION OVERHEAD PER USER PER ROUND (CIFAR-10).

N \ Protocol	SecAgg	SparseSecAgg
25	19.8×10^5 bits	2.48×10^5 bits
50	19.8×10^5 bits	2.54×10^5 bits
75	19.8×10^5 bits	2.56×10^5 bits
100	19.8×10^5 bits	2.57×10^5 bits

overhead to reach the target accuracy, and observe that SparseSecAgg reduces the communication overhead by $7.1\times$ compared to SecAgg. In Figure 4, we demonstrate the convergence behavior of SparseSecAgg versus SecAgg. We observe that even though SparseSecAgg shares only a small fraction of the gradient parameters per round, the convergence behavior of the two protocols are comparable, with SecAgg reaching the target accuracy only a few iterations before SparseSecAgg for the same number of users[1].

For the MNIST experiments, we consider both IID and non-IID data distribution settings, along with the CNN architecture from [1]. The target accuracy is 97.5% for the IID setting and 94% for the non-IID setting, respectively. For both settings, we compare the communication overhead for SparseSecAgg and SecAgg, to reach the target accuracy. In Figure 5, we demonstrate that SparseSecAgg reduces the communication overhead by $7.5\times$ with respect to SecAgg. Finally, we present the results for the MNIST dataset under the non-IID setup with a target test accuracy 94%. In Figure 6, we observe that in this case SparseSecAgg reduces the communication overhead by $7.6\times$ compared to SecAgg.

VII. CONCLUSION

This work presents the first gradient sparsification framework for secure aggregation. The proposed framework ensures efficient aggregation of sparsified gradients from a large number of users, without revealing the gradients parameters in the clear. Our experiments demonstrate a significant reduction in the communication overhead compared to conventional secure aggregation benchmarks. Future directions include integrating our secure aggregation framework with different federated learning topologies such as asynchronous learning.

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 *Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, ser. Proceedings of Machine Learning Research, vol. 54, Fort Lauderdale, FL, USA, Apr 2017, pp. 1273–1282.

[2] P. Kairouz and H. B. McMahan, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1, 2021.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[1]The increase in the number of iterations as N increases is expected. The local dataset sizes of the clients shrink as N grows as the dataset is divided equally to N users [1].

[4] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, no. 1, pp. 1–19, 2021.

[5] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "Fedhealth: A federated transfer learning framework for wearable healthcare," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 83–93, 2020.

[6] W. Y. B. Lim, S. Garg, Z. Xiong, D. Niyato, C. Leung, C. Miao, and M. Guizani, "Dynamic contract design for federated learning in smart healthcare applications," *IEEE Internet of Things Journal*, 2020.

[7] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.

[8] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 739–753.

[9] L. Zhu and S. Han, "Deep leakage from gradients," in *Federated Learning*. Springer, 2020, pp. 17–31.

[10] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" in *Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

[11] 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 *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[12] J. So, B. Güler, and A. S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," *IEEE Journal on Selected Areas in Information Theory*, 2021.

[13] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1253–1269.

[14] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.

[15] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv preprint arXiv:1704.05021*, 2017.

[16] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[17] P. Jiang and G. Agrawal, "A linear speedup analysis of distributed deep learning with sparse and quantized communication," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 2530–2541.

[18] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," *Annual Conference on Neural Information Processing Systems, NeurIPS*, 2018.

[19] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," *arXiv preprint arXiv:1710.09854*, 2017.

[20] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Foundations and Trends in Privacy and Security*, vol. 2, no. 2-3, 2017.

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

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

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

[24] J. So, R. E. Ali, B. Guler, J. Jiao, and S. Avestimehr, "Securing secure aggregation: Mitigating multi-round privacy leakage in federated learning," *arXiv preprint arXiv:2106.03328*, 2021.

[25] I. Ergun, H. U. Sami, and B. Guler, "Sparsified secure aggregation for privacy-preserving federated learning," *CoRR*, vol. abs/2112.12872, 2021. [Online]. Available: https://arxiv.org/abs/2112.12872

[26] 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.

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