Sparsity-Based Secure Gradient Aggregation for Resource-Constrained Federated Learning

Hasin Us Sami Başak Güler
Department of Electrical and Computer Engineering
University of California, Riverside, CA
hsami003@ucr.edu, bguler@ece.ucr.edu

Abstract—**Secure aggregation is an information-theoretic mechanism for gradient aggregation in federated learning, to aggregate the local user gradients without revealing them in the clear. In this work, we study secure aggregation under gradient sparsification constraints, for resource-limited wireless networks, where only a small fraction of local parameters are aggregated from each user during training (as opposed to the full gradient). We demonstrate that conventional mechanisms can reveal sensitive user data when aggregating sparsified gradients, due to the auxiliary coordinate information shared during sparsification, even when the individual gradients are not disclosed in the clear. We then propose a coordinate-hiding sparsified secure aggregation mechanism to address this challenge, which hides both the gradient parameters and the associated coordinates under formal information-theoretic privacy guarantees. Our framework reduces the communication overhead of conventional secure aggregation baselines by an order of magnitude without compromising model accuracy.**

I. Introduction

Federated learning (FL) is a popular paradigm for distributed training, where data-owners (users) perform training on locally collected datasets, after which the local updates (e.g., local gradients) are aggregated by a server to form a global model [1]. While popular in a variety of privacy-sensitive applications (such as healthcare) due to its on-device training architecture (data never leaves the device), FL can still reveal sensitive information about the local data samples through what is known as gradient inversion attacks [2]–[5].

Secure aggregation (SA) is an information-theoretic mechanism to address this challenge without compromising model accuracy [6]–[11]. SA allows the server to learn the sum of the local gradients, but without learning any further information about the individual gradients (beyond their sum). In doing so, SA prevents the server from associating the aggregated gradients with any particular user, enhancing resilience against inversion attacks as the number of users increases [6]–[11]. While popular in enhancing user privacy in distributed settings, communication overhead is still a major challenge in SA, which can hinder scalability to larger networks.

Gradient sparsification is a widely adopted technique to reduce the communication overhead in FL. In this setting, each user shares only a small portion of their local gradient parameters with the server (as opposed to sending the entire gradient), along with their coordinates. The parameters are selected uniformly random (rand-K), or based on their magnitude (top-K), where K is the number of parameters sent from each user [12]–[19]. The server then aggregates the received parameters using the received coordinates, where parameters from different users are aggregated if their coordinates match, to update the global model for the next training round.

In this work, we study gradient sparsification in the context of SA. We first demonstrate the potential vulnerabilities associated with sharing coordinate information during sparsification. In particular, we show that the local data samples can be recovered from the aggregate of the gradient parameters using the coordinates shared over multiple training rounds, even if the gradients are securely aggregated (using SA) at each training round. We then propose a coordinate-hiding SA framework to address this challenge, which hides not only the gradient parameters but also their coordinates during aggregation. Our framework enables the server to aggregate the sparsified local gradients, but without learning any information about the gradient parameters (beyond their aggregate) or their coordinates, under formal information-theoretic privacy guarantees.

Our framework builds on an offline-online trade-off, where we offload the communication-intensive operations, such as randomness generation, to a data-independent offline phase. The offline phase can take place in advance when the network load is low, or can be parallelized with other components of training. The online phase depends on the local datasets, hence should be carried out during training. We then propose an efficient SA mechanism for the online phase, to aggregate the sparsified gradients in FL, while revealing no information about the individual parameters or their coordinates (beyond their aggregated information).

A related line of work is preserving the privacy of sparse updates and the corresponding coordinates for gradient sparsification in the context of Private Information Retrieval (PIR) [20], [21]. Different from our work, the PIR setting builds on a multi-server setup, where users do not collude with the servers. In contrast, we consider a single-server FL task where any set of up to T users may collude with the server.

Our contributions can be summarized as follows:

1) We identify the vulnerabilities of sharing coordinate information in SA, and the necessity of stronger, topology-hiding privacy notions to enhance adversary-resilience under sparsification constraints.

This 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 UCR OASIS Fellowship.

979-8-3503-8284-6/24/$31.00 ©2024 IEEE

2) We demonstrate successful reconstruction of local data samples from the aggregate of sparsified gradients, by utilizing only the knowledge of the rand-K/top-K coordinates of the users.
3) We propose a coordinate-hiding sparsified SA framework, TinySecAgg, to enhance the scalability of SA in large model settings, without revealing any information about the gradient parameters or their coordinates, under formal information-theoretic privacy guarantees.
4) Our experiments demonstrate that our framework cuts the communication cost by an order of magnitude ($22.5\times$) over conventional SA mechanisms.

II. Problem Formulation

We consider a centralized FL architecture with N users, coordinated by a central server. User $i \in [N]$ holds a local dataset $\mathcal{D}_i$ with $D_i \triangleq |\mathcal{D}_i|$ samples. The goal is to train a global model $\mathbf{w} \in \mathbb{R}^d$ to minimize the global loss,

$$F(\mathbf{w}) \triangleq \sum_{i=1}^{N} F_i(\mathbf{w}) \tag{1}$$

where $F_i(\mathbf{w})$ denotes the local loss of user i. Training is performed iteratively through global and local training rounds. At the beginning of each global round t, the server sends the current state of the global model $\mathbf{w}^t$ to the users. User $i \in [N]$ generates a local model $\mathbf{w}_i^t \leftarrow \mathbf{w}^t$, which is updated locally through E local training rounds,

$$\mathbf{w}_i^t \leftarrow \mathbf{w}_i^t - \eta \nabla F_i(\mathbf{w}_i^t) \tag{2}$$

where $\nabla F_i(\mathbf{w}_i^t)$ is the gradient evaluated on $\mathcal{D}_i$, and η is the learning rate. After E local training rounds, user i sends the (cumulative) local gradient,

$$\Delta_i^t \triangleq \mathbf{w}_i^t - \mathbf{w}^t \in \mathbb{R}^d \tag{3}$$

to the server, who then aggregates the received gradients to update the global model for the next training round,

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \frac{1}{|\mathcal{U}(t)|} \sum_{i \in \mathcal{U}(t)} \Delta_i^t \tag{4}$$

where $\mathcal{U}(t)$ denotes the set of surviving users who succeed in sending their local updates to the server at round t, as some users may drop out from the protocol due to poor wireless connectivity, or device unavailability.

Gradient sparsification is a popular compression mechanism to improve the communication efficiency in FL, where, instead of sending the entire gradient Δ_i^t to the server, each user sends only $K \ll d$ selected parameters. Sparsification typically involves a process known as *error-accumulation*, to track the cumulative error resulting from the parameters that have not been sent in previous rounds. Accordingly, the sparsification operation is given by,

$$\mathbf{x}_i^t \triangleq \mathbf{b}_i^t \odot \widetilde{\Delta}_i^t \tag{5}$$

where $\odot$ is the Hadamard product, $\mathbf{x}_i^t$ denotes the sparsified local gradient, $\mathbf{b}_i^t \in \{0,1\}^d$ is a binary mask holding the coordinates of the K parameters selected by user i, where the ℓ^{th} element is given by,

$$\mathbf{b}_i^t(\ell) \triangleq \begin{cases} 1 & \text{if user } i \text{ selects coordinate } \ell \text{ at round } t \\ 0 & \text{otherwise} \end{cases}$$

such that $\|\mathbf{b}_i^t\|_1 = K$, where $\|\cdot\|_1$ denotes the L_1 norm and

$$\widetilde{\Delta}_i^t \triangleq \mathbf{w}_i^t - \mathbf{w}^t + \mathbf{e}_i^{t-1} = \Delta_i^t + \mathbf{e}_i^{t-1}, \tag{6}$$

where $\mathbf{e}_i^t$ denotes the error accumulated at round t,

$$\mathbf{e}_i^t \triangleq \Delta_i^t + \mathbf{e}_i^{t-1} - \mathbf{x}_i^t = \widetilde{\Delta}_i^t - \mathbf{x}_i^t. \tag{7}$$

After constructing the sparsified local gradient $\mathbf{x}_i^t$ from (5), user i then sends the selected K parameters from $\mathbf{x}_i^t$ to the server, along with the coordinates of the selected parameters. Using the received coordinates, the server aggregates the sparsified local gradients $\mathbf{x}_i^t$ to update the global model,

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \frac{1}{|\mathcal{U}(t)|} \sum_{i \in \mathcal{U}(t)} \mathbf{x}_i^t \tag{8}$$

for the next training round. The specific structure of the binary mask $\mathbf{b}_i^t$ depends on the sparsification methodology:

1) rand-K sparsification: In this setting, each user selects K parameters uniformly at random (without replacement) from $\widetilde{\Delta}_i^t$, and $\mathbf{b}_i^t$ is a uniformly random binary vector where $\|\mathbf{b}_i^t\|_1 = K$, generated independently for each user $i \in [N]$.

2) top-K sparsification: In this setting, users send only the top K parameters with the highest magnitude to the server, and $\mathbf{b}_i^t \in \{0,1\}^d$ is a binary vector indicating the coordinates of the top K parameters from $\widetilde{\Delta}_i^t$ with the highest magnitude.

The two mechanisms (rand-K/top-K) have complementary benefits. Rand-K is more memory and communication efficient; as the binary masks $\mathbf{b}_i^t$ are sampled independently and uniformly at random, they can be generated offline in advance when the network load is low. Top-K can speed up convergence, but the coordinates depend on the gradient magnitudes, which has to be sent online during training.

Threat model. In this work, our focus is on *honest-but-curious* adversaries (as is the most common threat model in SA), where adversaries do not poison the datasets, but try to reveal additional information about the local datasets of honest users using the information exchanged during protocol execution. Similar to [22], the server can freeze the global model parameters (albeit do not change them). Out of N users, up to $T < N$ users are adversarial, who may collude with each other and/or the server. The set of honest and adversarial users are denoted by $\mathcal{H}$ and $\mathcal{T} = [N] \backslash \mathcal{H}$, respectively.

Main problem. SA aims at aggregating the local updates $\mathbf{x}_i^t$,

$$\mathbf{x}_{agg}^t \triangleq \sum_{i \in \mathcal{U}(t)} \mathbf{x}_i^t \tag{9}$$

but without revealing any information about the individual updates (beyond their sum). Formally, this can be stated by the following mutual information condition,

$$I(\{\mathbf{x}_i^t\}_{i \in \mathcal{H}}; \mathcal{M}_{\mathcal{T}}^t | \mathbf{x}_{agg}^t, \{\mathbf{x}_i^t\}_{i \in \mathcal{T}}, \mathcal{R}_{\mathcal{T}}^t) = 0 \tag{10}$$

where $\mathcal{M}_{\mathcal{T}}^t$ denotes the set of all messages received, and $\mathcal{R}_{\mathcal{T}}^t$

denotes the randomness generated, by the adversaries and the server at round t. The correct recovery of the aggregate in (9) is formalized by the following entropy constraint,

$$H(\mathbf{x}_{agg}^t | \mathcal{M}_{\mathcal{U}(t)}^t) = 0 \tag{11}$$

where $\mathcal{M}_{\mathcal{U}(t)}^t$ denotes the set of all messages held by the surviving users $\mathcal{U}(t)$ at round t. To compute (9) under the information-theoretic privacy guarantees from (10), SA protocols enable users to *encode* their local updates $\mathbf{x}_i^t$ by using locally generated random secret masks, and send only an encoded version to the server. The encoding process hides the true value of the local updates from the server, while still allowing the server to decode their sum as in (9), without learning any information about the individual updates $\mathbf{x}_i^t$. In doing so, SA protocols differ in their encoding/decoding mechanism. A common challenge in conventional SA protocols is the communication overhead with large models, as the dimensionality of the encoded gradient sent from each user is as large as the true gradient, which prevents scalability to larger models in practice. Our goal in this work is to address this challenge, where we ask the question,

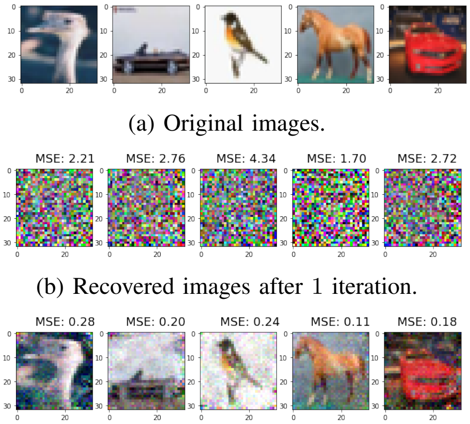
- Can gradient sparsification enhance the scalability of secure aggregation?

In this work, we answer this question in the affirmative. Sparsification can enhance the communication-efficiency of SA, but additional care should be taken to hide the coordinates, and naive approaches can do more harm than good. Specifically, as we demonstrate in the following section, the coordinate information exchanged during gradient sparsification can introduce new vulnerabilities to SA. This is due to the fact that the coordinates of sparsified local parameters vary across the users throughout the training, hence, by using the coordinates shared over multiple rounds, adversaries can reconstruct the local gradients of individual users from their sum, even when the gradients are aggregated using SA. To address this challenge, we then introduce TinySecAgg, a coordinate-hiding sparse SA framework. Our framework hides both the sparsified gradient parameters and their coordinates from the server, under formal information-theoretic privacy guarantees, while significantly enhancing the communication efficiency of SA.

III. Reconstruction from Compressed Gradients

In this section, we discuss the naive application of gradient sparsification with SA, to present the associated risks. Gradient sparsification, as described in Section II, is compatible with most well-known SA protocols [6], [7], where the key premise is to learn the sum $\mathbf{x}_{agg}^t = \sum_{i \in \mathcal{U}(t)} \mathbf{x}_i^t$ from (9), without disclosing the local updates $\mathbf{x}_i^t$. In doing so, users hide their selected parameters with additive random masks, and send the encoded parameters (and their coordinates) to the server. The additive masks are constructed in a way that they cancel out upon aggregation at the server, thus allowing the server to learn the sum $\mathbf{x}_{agg}^t$ of the local gradients, but without revealing any further information about the individual gradients $\mathbf{x}_i^t$.

In the following, we consider the frozen-model attack from [22], where the server freezes the model parameters sent to the users, as a result the local gradients Δ_i^t are stable throughout the iterations. This setting can also emerge (without the attack) when the model is close to convergence. We next demonstrate a gradient reconstruction mechanism that allows the server to recover the individual gradients Δ_i^t for all $i \in [N]$, using only the sum of the sparsified gradients $\mathbf{x}_{agg}^t$ and their coordinates over multiple training rounds.



(a) Original images.

(b) Recovered images after 1 iteration.

(c) Recovered images after 500 iterations.

Fig. 1: (rand-K) Image reconstruction quality with varying number of training rounds.

Let $\tau_i^t(\ell) < t$ be the most recent training round (prior to t) in which user i shares coordinate ℓ with the server. From the error accumulated up to round t, one can rewrite (6) as,

$$\widetilde{\Delta}_i^t(\ell) = (t - \tau_i^t(\ell))\Delta_i(\ell) \tag{12}$$

where $\Delta_i(\ell)$ is the ℓ^{th} element of the gradient[2] Δ_i of user i as defined in (3), using which the sparsified gradient from (5) can be written as,

$$\mathbf{x}_i^t(\ell) = \mathbf{b}_i^t(\ell)(t - \tau_i^t(\ell))\Delta_i(\ell) \tag{13}$$

where $\mathbf{x}_i^t(\ell)$ denotes the ℓ^{th} element of $\mathbf{x}_i^t$. After SA, the server learns the aggregate of the sparsified gradients for each coordinate $\ell \in [d]$,

$$\mathbf{x}_{agg}^t(\ell) = \sum_{i \in \mathcal{U}(t)} \mathbf{x}_i^t(\ell) \tag{14}$$

$$= \sum_{i \in \mathcal{U}(t)} \mathbf{b}_i^t(\ell)(t - \tau_i^t(\ell))\Delta_i(\ell) \tag{15}$$

From (15), along with the selected coordinates $\mathbf{b}_i^t$ from users $i \in \mathcal{U}(t)$, the server can construct the following,

$$\mathbf{A} \begin{bmatrix} \Delta_1(\ell) \\ \vdots \\ \Delta_N(\ell) \end{bmatrix} + \mathbf{n} = \begin{bmatrix} \mathbf{x}_{agg}^1(\ell) \\ \vdots \\ \mathbf{x}_{agg}^J(\ell) \end{bmatrix} \quad \forall \ell \in [d], \tag{16}$$

where J is the total number of training rounds, $\mathbf{n}$ denotes the noise incurred due to local training across the rounds, and $\mathbf{A}$

[2] One can omit the time index t from the local gradient Δ_i^t as the local gradients are stable throughout the iterations.

is a $J \times N$ matrix defined as,

$$\mathbf{A} \triangleq \begin{bmatrix} \mathbf{b}_1^1(\ell)(1-\tau_1^1(\ell)) & \cdots & \mathbf{b}_N^1(\ell)(1-\tau_N^1(\ell)) \\ \vdots & \vdots & \vdots \\ \mathbf{b}_1^J(\ell)(J-\tau_1^J(\ell)) & \cdots & \mathbf{b}_N^J(\ell)(J-\tau_N^J(\ell)) \end{bmatrix} \tag{17}$$

by letting $\mathbf{b}_i^t(\ell) = 0$ for the dropout users $i \in [N]\backslash\mathcal{U}(t)$ without loss of generality. By using $\mathbf{A}$ and the aggregate of the sparsified gradients $\{\mathbf{x}_{agg}^t\}_{t\in[J]}$, the server can finally recover the local gradients $\Delta^*(\ell) \triangleq \begin{bmatrix}\Delta_1(\ell) & \cdots & \Delta_N(\ell)\end{bmatrix}^{\mathrm{T}}$ for each coordinate $\ell \in [d]$, by solving a least squares problem,

$$\Delta^*(\ell) = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\begin{bmatrix}\mathbf{x}_{agg}^1(\ell) & \cdots & \mathbf{x}_{agg}^J(\ell)\end{bmatrix}^{\mathrm{T}}$$

Upon recovering the local gradients $\{\Delta^*(\ell)\}_{\ell\in[d]}$, the server can apply any gradient inversion attack (e.g., [3]) to reveal the local data samples from the local gradients.

In Fig. 1, we demonstrate the image reconstruction quality on a ResNet-18 model trained on the CIFAR-10 dataset across 5 users [23], [24], where each user holds a single random data sample in accordance with [3], [22]. The selected parameters are aggregated using the SA protocol SecAgg from [6], while we note that our results are indifferent to the specific SA protocol used (as the final aggregated gradient is the same). After reconstructing the local gradients, gradient inversion from [3] is applied to recover the images. The reconstruction quality is measured using the mean square error (MSE) between the recovered and original image. Fig. 1 demonstrates the recovered images for rand-K sparsification, with $K = 0.01d$, i.e., only 1% of the gradient parameters are aggregated from each user. We observe that the quality of the recovered images approaches the original images after a sufficiently large number of training rounds.

Our key observation is that the attack can only be launched when coordinate information is available. When coordinate information is not available, reconstruction is unsuccessful, as observed in Fig. 1(b). Motivated by these findings, in the following, we introduce a coordinate-hiding SA mechanism, to enhance the security of SA under sparsification.

IV. Coordinate-hiding Gradient Sparsification for Secure Aggregation

This section presents TinySecAgg, a coordinate-hiding rand-K gradient sparsification mechanism for SA. Our framework builds on an offline-online trade-off, where we offload the communication-intensive operations, such as randomness generation, to a data-independent offline phase, which can take place in advance prior to training. In the online phase, users only communicate an encoded version of the selected gradient parameters and their coordinates (as opposed to their true values). At the end, the server decodes the correct aggregate of the gradient parameters for each coordinate, but without learning the individual parameters *or their coordinates*. Our framework operates in a finite field $\mathbb{F}_p$ of integers modulo a large prime p, where all operations are carried out in $\mathbb{F}_p$. We next describe the details of the offline and online phases.

Offline Phase. Initially, users define a binary vector $\mathbf{a}_k \in \{0,1\}^d$ for each $k \in [d]$, where only the k^{th} element is equal to 1, and all other elements are 0, and then partition $\mathbf{a}_k$ into M equal-sized shards,

$$\mathbf{a}_k = \begin{bmatrix}\mathbf{a}_{k1}^{\mathrm{T}} & \ldots & \mathbf{a}_{kM}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}. \tag{18}$$

where parameter M controls a trade-off between communication-efficiency and adversary tolerance. Users then agree on $N + M + T$ distinct public parameters $\{\alpha_i\}_{i\in[N]}$, $\{\beta_n\}_{n\in[M+T]}$ from $\mathbb{F}_p$. User $i \in [N]$ generates a random binary mask $\mathbf{b}_i^t \in \{0,1\}^d$ for rand-K sparsification, where K out of d elements are set to 1 uniformly at random (without replacement). Let $\mathcal{K}_i^t \triangleq \{\ell : \mathbf{b}_i^t(\ell) = 1\}$ denote the (ordered) set of the K coordinates selected by user i. Then, user i generates two Lagrange interpolation polynomials,

$$\phi_{ik}(\alpha) \triangleq \sum_{n\in[M]} \mathbf{a}_{\mathcal{K}_i^t(k),n} \prod_{n'\in[M+T]\backslash\{n\}} \frac{\alpha-\beta_{n'}}{\beta_n-\beta_{n'}} + \sum_{n=M+1}^{M+T} \mathbf{v}_{ikn}^t \prod_{n'\in[M+T]\backslash\{n\}} \frac{\alpha-\beta_{n'}}{\beta_n-\beta_{n'}}, \tag{19}$$

$$\psi_{ik}(\alpha) \triangleq \sum_{n\in[M]} \mathbf{a}_{\mathcal{K}_i^t(k),n} r_{ik}^t \prod_{n'\in[M+T]\backslash\{n\}} \frac{\alpha-\beta_{n'}}{\beta_n-\beta_{n'}} + \sum_{n=M+1}^{M+T} \mathbf{u}_{ikn}^t \prod_{n'\in[M+T]\backslash\{n\}} \frac{\alpha-\beta_{n'}}{\beta_n-\beta_{n'}}, \tag{20}$$

for all $k \in [K]$, where $\mathcal{K}_i^t(k)$ denotes the k^{th} element of $\mathcal{K}_i^t$; $\{r_{ik}^t\}_{k\in[K]}$ are K random masks generated uniformly at random from $\mathbb{F}_p$; and $\{\mathbf{v}_{ikn}^t, \mathbf{u}_{ikn}^t\}_{k\in[K],n\in\{M+1,\ldots,M+T\}}$ are generated uniformly at random from $\mathbb{F}_p^{d/M}$. The random masks $\{r_{ik}^t\}_{k\in[K]}$ will later be used to hide the true value of the gradient parameters in the online phase, whereas the random vectors $\{\mathbf{v}_{ikn}^t, \mathbf{u}_{ikn}^t\}_{k\in[K],n\in\{M+1,\ldots,M+T\}}$ will hide the contents of the masks $\{r_{ik}^t\}_{k\in[K]}$ as well as the selected coordinates. Finally, user i sends the encoded vectors $\phi_{ik}(\alpha_j)$ and $\psi_{ik}(\alpha_j)$ to user $j \in [N]$, which will later be used in the online phase to ensure the correct matching of the local gradient parameters within the global model, while preventing the server from gaining explicit access to the coordinates.

Online Phase. After local training and sparsification, user $i \in [N]$ transforms its sparsified local gradient $\mathbf{x}_i^t \in \mathbb{R}^d$ to $\mathbb{F}_p$,

$$\overline{\mathbf{x}}_i^t(\ell) \triangleq f(\mathbf{x}_i^t(\ell)) \quad \forall \ell \in \mathcal{K}_i^t \tag{21}$$

where the finite field transform $f(\cdot): \mathbb{R}^d \to \mathbb{F}_p^d$ is common to secure multi-party computing frameworks [25]–[28]. For the details of this transformation, we refer to [11], [26], [29], [30]. Next, user i broadcasts a masked gradient parameter,

$$\widehat{\mathbf{x}}_{ik}^t \triangleq \overline{\mathbf{x}}_i^t(\mathcal{K}_i^t(k)) - r_{ik}^t \tag{22}$$

for each $k \in [K]$, where the true content of the K selected parameters $\mathcal{K}_i^t$ are hidden by the K random masks $r_{i1}^t, \ldots, r_{iK}^t$ generated in the offline phase. After receiving (22), each user i sends a local aggregate of the encoded gradients:

$$\varphi(\alpha_i) \triangleq \sum_{j\in\mathcal{U}(t)} \sum_{k\in[K]} (\widehat{\mathbf{x}}_{jk}^t \phi_{jk}(\alpha_i) + \psi_{jk}(\alpha_i)), \tag{23}$$

to the server. The local computations $\varphi(\alpha_i)$ in (23) can be

	Online	Offline
SecAgg	$O(N+d)$	$O(N)$
SecAgg+	$O(\log(N)+d)$	$O(\log(N))$
LightSecAgg	$O(d+d/M)$	$O(Nd/M)$
TinySecAgg	$O(K+d/M)$	$O(KNd/M)$

TABLE I: Comparison of per-round communication complexity of TinySecAgg with SA baselines.

viewed as evaluations of a degree $M+T-1$ polynomial $\varphi(\alpha)$ at $\alpha=\alpha_i$. As a result, after collecting $\varphi(\alpha_i)$ from any set of $M+T$ users, the server can reconstruct the polynomial $\varphi(\alpha)$ via polynomial interpolation, and recover,

$$\mathbf{x}^t_{agg} = \sum_{i\in\mathcal{U}(t)} \overline{\mathbf{x}}^t_i = \left[\varphi^{\mathrm{T}}(\beta_1) \quad \cdots \quad \varphi^{\mathrm{T}}(\beta_M)\right]^{\mathrm{T}} \tag{24}$$

which corresponds to the *true (desired) sum of the sparsified local gradients*. After aggregating the local gradients, the server updates the global model,

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \frac{1}{|\mathcal{U}(t)|} f^{-1}(\mathbf{x}^t_{agg}) \tag{25}$$

In the following, we demonstrate the theoretical guarantees and performance trade-offs of TinySecAgg.

V. Theoretical Analysis

In this section, we present the formal privacy, complexity, and correctness guarantees.

Theorem 1. *(Communication Complexity) The per-user communication complexity of TinySecAgg is $O(K+\frac{d}{M})$ (online) and $O(KN\frac{d}{M})$ (offline).*

Theorem 2. *(Computation Complexity) The per-user computation complexity of TinySecAgg is $O(NK\frac{d}{M})$ (online) and $O(NK\frac{d}{M}\log^2(M+T)\log\log(M+T))$ (offline).*

Theorem 3. *(Correctness) TinySecAgg ensures the correct recovery of the aggregate $\mathbf{x}^t_{agg} = \sum_{i\in\mathcal{U}(t)} \overline{\mathbf{x}}^t_i$ of sparsified gradients from* (24)*, from the messages of any set $\mathcal{U}(t)$ of $|\mathcal{U}(t)| \geq M+T$ surviving users,*

$$H\Big(\sum_{i\in\mathcal{U}(t)} \overline{\mathbf{x}}^t_i \big| \{\widehat{\mathbf{x}}^t_i\}_{i\in\mathcal{U}(t)}, \{\varphi(\alpha_i)\}_{i\in\mathcal{U}(t)}\Big) = 0 \tag{26}$$

Theorem 4. *(Privacy) TinySecAgg ensures information-theoretic privacy for both the gradients and their coordinates against up to T adversaries, for any $|\mathcal{U}(t)| \geq T+M$,*

$$I(\{\overline{\mathbf{x}}^t_i, \mathcal{K}^t_i\}_{[N]\setminus\mathcal{T}}; \mathcal{M}^t_{\mathcal{T}} | \sum_{i\in\mathcal{U}(t)} \overline{\mathbf{x}}^t_i, \{\overline{\mathbf{x}}^t_i, \mathcal{K}^t_i\}_{i\in\mathcal{T}}, \mathcal{R}^t_{\mathcal{T}}) = 0$$

Finally, the convergence guarantees follow using the standard assumptions from [12], [14], [19], [31].

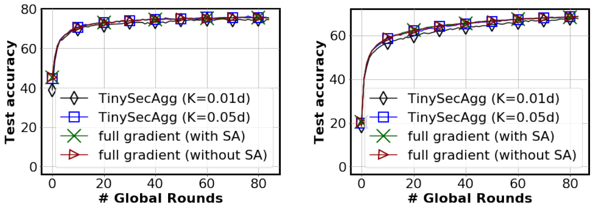
VI. Experiments

We evaluate the performance of TinySecAgg with respect to conventional SA baselines SecAgg [6], SecAgg+ [7], and LightSecAgg [8], where users aggregate the full gradient.

Setup. We consider a distributed network with $N=100$ users, where 10% of the users (randomly) drop out at each round. The number of adversarial users is $T=\frac{N}{2}$ [6]. We consider image classification tasks on the following datasets: 1) FEMNIST, a non-IID dataset by design, using the CNN model and data distribution from [32], 2) CIFAR-10, using the CNN model and data distribution (IID) from [1]. We evaluate the performance for two sparsification levels, $K=0.05d$, and $K=0.01d$, with $M=40$, which is the maximum number of shards allowed by Theorem 4.

Framework	FEMNIST	CIFAR-10
SecAgg	738813	172304
SecAgg+	738810	172300
LightSecAgg	757277	176601
TinySecAgg ($K=0.05d$)	56831	13150
TinySecAgg ($K=0.01d$)	43760	7645

TABLE II: Total communication overhead (Mbits) to reach convergence test accuracy (75%-FEMNIST, 67%-CIFAR-10).



(a) FEMNIST.

(b) CIFAR-10.

Fig. 2: Test accuracy of TinySecAgg vs. full gradient aggregation (with and without SA).

Table I compares the per-round communication complexity of TinySecAgg with the SA baselines. In Table II, we present the total online communication overhead to reach the target test accuracy at convergence. As the model size d is large (in the order of 10^6) compared to the number of users N, all SA baselines incur similar communication overhead (as they communicate the full gradient), whereas TinySecAgg reduces the cost by up to $22.5\times$ over the best baseline.

In Fig. 2, we further compare the convergence of TinySecAgg (SA with sparsification) with respect to SA without sparsification (i.e., full gradient aggregation with SA). All SA baselines carry out the same training operations and hence have the same test accuracy, as they only differ in their encoding mechanism, not in training. We also demonstrate the convergence of the same model architecture, but without sparsification or SA (i.e., full gradient aggregation without SA). This represents our target accuracy, as the finite-field conversion in SA can degrade accuracy. We observe that TinySecAgg achieves comparable accuracy to all baselines (with or without SA).

VII. Conclusion

In this work, we study gradient sparsification in the context of SA. We identify the vulnerabilities of sharing coordinate information in gradient sparsification, and introduce a coordinate-hiding SA mechanism, TinySecAgg, to enhance the reliability of SA in resource-limited settings. Our framework provides an order of magnitude improvement in communication-efficiency without degrading model accuracy.

References

[1] 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.

[2] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

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

[4] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2021.

[5] Y. Wen, J. Geiping, L. Fowl, M. Goldblum, and T. Goldstein, "Fishing for user data in large-batch federated learning via gradient magnification," in *International Conference on Machine Learning, ICML*, 2022.

[6] 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 ACM SIGSAC Conf. on Computer and Communications Security (CCS)*, 2017.

[7] 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 (CCS)*, 2020, pp. 1253–1269.

[8] J. So, C.-S. Yang, S. Li, Q. Yu, R. E Ali, B. Guler, and S. Avestimehr, "Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning," *Proc. of Machine Learning and Systems*, 2022.

[9] Y. Zhao and H. Sun, "Information theoretic secure aggregation with user dropouts," *IEEE Transactions on Information Theory*, vol. 68, no. 11, pp. 7471–7484, 2022.

[10] H. U. Sami and B. Güler, "Secure aggregation for clustered federated learning," in *IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 186–191.

[11] ——, "Secure aggregation for clustered federated learning with passive adversaries," *IEEE Transactions on Communications*, 2024.

[12] S. U. Stich, J. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Annual Conf. on Neural Information Processing Systems*, 2018.

[13] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[14] P. Jiang and G. Agrawal, "A linear speedup analysis of distributed deep learning with sparse and quantized communication," in *Advances in Neural Information Processing Systems (Neurips)*, 2018, pp. 2530–2541.

[15] H. Xu, C.-Y. Ho, A. M. Abdelmoniem, A. Dutta, E. H. Bergou, K. Karatsenidis, M. Canini, and P. Kalnis, "Grace: A compressed communication framework for distributed machine learning," in *IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, 2021.

[16] A. Xu and H. Huang, "Detached error feedback for distributed sgd with random sparsification," in *Int. Conf. on Machine Learning (ICML)*, 2020.

[17] H. U. Sami and B. Güler, "Over-the-air personalized federated learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 8777–8781.

[18] J. Wang, Y. Lu, B. Yuan, B. Chen, P. Liang, C. De Sa, C. Re, and C. Zhang, "CocktailSGD: Fine-tuning foundation models over 500Mbps networks," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023, pp. 36 058–36 076.

[19] A. Xu, Z. Huo, and H. Huang, "Step-ahead error feedback for distributed training with compressed gradient," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, 2021, pp. 10 478–10 486.

[20] S. Vithana and S. Ulukus, "Private read update write (PRUW) in federated submodel learning (FSL): communication efficient schemes with and without sparsification," *IEEE Trans. Inf. Theory*, 2024.

[21] ——, "Private read-update-write with controllable information leakage for storage-efficient federated learning with top r sparsification," *IEEE Trans. Inf. Theory*, 2023.

[22] M. Lam, G. Wei, D. Brooks, V. J. Reddi, and M. Mitzenmacher, "Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix," in *Proceedings of the 38th International Conference on Machine Learning, (ICML)*, 2021.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016, pp. 770–778.

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

[25] X. Lu and B. Güler, "Breaking the quadratic communication overhead of secure multi-party neural network training," in *IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 921–926.

[26] X. Lu, H. U. Sami, and B. Güler, "Dropout-resilient secure multi-party collaborative learning with linear communication complexity," in *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 206, 2023, pp. 10 566–10 593.

[27] X. Lu, H. U. Sami, and B. Güler, "Privacy-preserving collaborative learning with linear communication complexity," *IEEE Transactions on Information Theory*, 2023.

[28] ——, "Scalr: Communication-efficient secure multi-party logistic regression," *IEEE Transactions on Communications*, vol. 72, no. 1, pp. 162–178, 2024.

[29] J. So, B. Guler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Comm.*, 2020.

[30] J. So, B. Güler, and S. Avestimehr, "A scalable approach for privacy-preserving collaborative machine learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[31] H. U. Sami and B. Güler, "Over-the-air clustered federated learning," *IEEE Transactions on Wireless Communications*, 2023.

[32] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.