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

Hasin Us Sami Ba¸ sak 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 collide with the servers. In contrast, we consider a single-server FL task where any set of up to \( T \) users may collide 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.
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 \(D_i\), with \(D_i \triangleq |D_i|\) samples. The goal is to train a global model \(w \in \mathbb{R}^d\) to minimize the global loss,

\[
F(w) \triangleq \sum_{i=1}^{N} F_i(w)
\]

where \(F_i(w)\) 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 \(w^t\) to the users. User \(i \in [N]\) generates a local model \(w_i^t \leftarrow w^t\), which is updated locally through \(E\) local training rounds,

\[
w_i^t \leftarrow w_i^t - \eta \nabla F_i(w_i^t)
\]

where \(\nabla F_i(w_i^t)\) is the gradient evaluated on \(D_i\), and \(\eta\) is the learning rate. After \(E\) local training rounds, user \(i\) sends the (cumulative) local gradient,

\[
\Delta_i^t \triangleq w_i^t - w^t \in \mathbb{R}^d
\]

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

\[
w^{t+1} = w^t - \frac{1}{|\mathcal{U}(t)|} \sum_{i \in \mathcal{U}(t)} \Delta_i^t
\]

for the next training round. The specific structure of the binary mask \(b_i^t\) depends on the sparsification methodology:

1) rand-\(K\) sparsification: In this setting, each user selects \(K\) parameters uniformly at random (without replacement) from \(\Delta_i^t\), and \(b_i^t\) is a uniformly random binary vector where \(\|b_i^t\|_1 = K\), 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 \(b_i^t \in \{0,1\}^d\) is a binary vector indicating the coordinates of the top \(K\) parameters from \(\Delta_i^t\) 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 \(b_i^t\) 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 \(\mathcal{T} = [N] \setminus \mathcal{H}\), respectively.

Main problem. SA aims at aggregating the local updates \(x_i^t\),

\[
x_{agg}^t \triangleq \sum_{i \in \mathcal{U}(t)} x_i^t
\]

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

\[
I(\{x_i^t\}_{i \in \mathcal{H}}; M_T^t|x_{agg}^t, \{x_i^t\}_{i \in \mathcal{T}}, R_T^t) = 0
\]

where \(M_T^t\) denotes the set of all messages received, and \(R_T^t\)
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,

$$H(x_{agg}^t | M_{ut(t)}^t) = 0$$

(11)

where $M_{ut(t)}^t$ denotes the set of all messages held by the surviving users $U(t)$ at round $t$. To compute (9) under the information-theoretic privacy guarantees from (10), SA protocols enable users to encode their local updates $x_i^t$ 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 $x_i^t$. 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 $x_{agg}^t = \sum_{i \in U(t)} x_i^t$ from (9), without disclosing the local updates $x_i^t$. 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 $x_{agg}^t$ of the local gradients, but without revealing any further information about the individual gradients $x_i^t$.

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 $\Delta_i^t$ 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 $\Delta_i^t$ for all $i \in [N]$, using only the sum of the sparsified gradients $x_{agg}^t$ and their coordinates over multiple training rounds.

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

$$\Delta_i(\ell) = (t - \tau_i^t(\ell)) \Delta_i(\ell)$$

(12)

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

$$x_i^t(\ell) = b_i^t(\ell) (t - \tau_i^t(\ell)) \Delta_i(\ell)$$

(13)

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

$$x_{agg}^t(\ell) = \sum_{i \in U(t)} x_i^t(\ell)$$

(14)

$$= \sum_{i \in U(t)} b_i^t(\ell) (t - \tau_i^t(\ell)) \Delta_i(\ell)$$

(15)

From (15), along with the selected coordinates $b_i^t$ from users $i \in U(t)$, the server can construct the following,

$$A = \begin{bmatrix} \Delta_1(\ell) \\ \vdots \\ \Delta_N(\ell) \end{bmatrix} + n = \begin{bmatrix} x_{agg}^1(\ell) \\ \vdots \\ x_{agg}^J(\ell) \end{bmatrix} \forall \ell \in [d],$$

(16)

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

\footnote{One can omit the time index $t$ from the local gradient $\Delta_i^t$ as the local gradients are stable throughout the iterations.}

(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.
is a $J \times N$ matrix defined as,
\[
A = \begin{bmatrix}
    b^1_1(\ell) (1 - \tau^1_1(\ell)) & \cdots & b^1_N(\ell) (1 - \tau^1_N(\ell)) \\
    \vdots & \ddots & \vdots \\
    b^J_1(\ell) (J - \tau^J_1(\ell)) & \cdots & b^J_N(\ell) (J - \tau^J_N(\ell))
\end{bmatrix}
\]
(17)
by letting $b^i_\ell(\ell) = 0$ for the dropout users $i \in [N] \setminus \mathcal{U}(\ell)$ without loss of generality. Using $A$ and the aggregate of the sparsified gradients $\{x^t_{agg}\}_{t \in [J]}$, the server can finally recover the local gradients $\Delta^t(\ell) \triangleq [\Delta_1(\ell) \cdots \Delta_N(\ell)]^T$ for each coordinate $\ell \in [d]$, by solving a least squares problem,
\[
\Delta^t(\ell) = (A^T A)^{-1} A^T [x^t_{agg}(\ell) \cdots x^J_{agg}(\ell)]^T
\]
Upon recovering the local gradients $\{\Delta^t(\ell)\}_{t \in [d]}$, 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. The specific SA protocol used (as the final aggregated gradient is random data sample in accordance with [3], [22], the selected 5\% on a ResNet-18 model trained on the CIFAR-10 dataset local data samples from the local gradients.

Our key observation is that the attack can only be launched for each $d \in [K]$, where $K^t\{k\}$ denotes the $k^{th}$ element of $K^t\{k\}$. \{r^t_{ik}\}_{k \in [K]}$ are $K$ random masks generated uniformly at random from $\mathbb{F}_p$; and \{v^t_{ikn}, u^t_{ikn}\}_{k \in [K], n \in [M+1,...,M+T]} are generated uniformly at random from $\mathbb{F}_p^{M+T}$. The random masks \{r^t_{ik}\}_{k \in [K]} will later be used to hide the true value of the gradient parameters in the online phase, whereas the random vectors \{v^t_{ikn}, u^t_{ikn}\}_{k \in [K], n \in [M+1,...,M+T]} will hide the contents of the masks \{r^t_{ik}\}_{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 $x^t_i \in \mathbb{R}^d$ to $\mathbb{F}_p$,
\[
x^t_i(\ell) \triangleq f(x^t_i(\ell)) \quad \forall \ell \in K^t_i
\]
where the finite field transform $f(\cdot) : \mathbb{R}^d \rightarrow \mathbb{F}_p^d$ 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,
\[
\tilde{x}^t_{ik} \triangleq x^t_i(K^t_i(k)) - r^t_{ik}
\]
for each $k \in [K]$, where the true content of the $K$ selected parameters $K^t_i$ are hidden by the $K$ random masks $s^t_{ik1}, s^t_{ik2}, \ldots, s^t_{ikK}$ generated in the offline phase. After receiving (22), each user $i$ sends a local aggregate of the encoded gradients:
\[
\varphi(\alpha_i) \triangleq \sum_{j \in \mathcal{U}(i)} \sum_{k \in [K]} (\tilde{x}^t_{jk} \phi_{jk}(\alpha_i) + \psi_{jk}(\alpha_i)),
\]
to the server. The local computations $\varphi(\alpha_i)$ in (23) can be
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,

$$x_{agg}^t = \sum_{i \in U(t)} x_i^t = [\varphi^T(\beta_1) \cdots \varphi^T(\beta_M)]^T$$ (24)

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

$$w^{t+1} - w^t - \frac{1}{|U(t)|} f^{-1}(x_{agg}^t)$$ (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 $O(K + d/M)$ (online) and $O(KNd/M)$ (offline).

Theorem 2. (Computation Complexity) The per-user computation complexity of TinySecAgg is $O(KN^2d/M)$ (online) and $O(NKd^2/M)$ (offline).

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

$$H(\sum_{i \in U(t)} x_i^t, \{x_i^t\}_{i \in U(t)}; \{\varphi(\alpha_i)\}_{i \in U(t)}) = 0$$ (26)

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

$$I(\{x_i^t, \kappa_i^t\}_{i \in U(t)}; T; M_T^t) = \sum_{i \in U(t)} x_i^t, \kappa_i^t; R_T^t) = 0$$

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.

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


