Secure Submodel Aggregation for Resource-Aware 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 (SA) is a privacy-enhancing framework for federated learning, to aggregate the local gradient updates from the users without revealing them in the clear. Conventional SA frameworks are built under the assumption of homogeneous computational resources across the users, where users are bound to train a local model whose dimensions are as large as the global model, preventing resource-limited users from participating in training. In this work, we propose a novel secure submodel training framework to address this challenge, where users train and communicate partial submodels through an adaptable secure aggregation mechanism during training. Our framework enables the participation of all users with varying computation and communication resources, while ensuring formal information-theoretic privacy guarantees for the individual local updates.**

I. Introduction

Federated learning (FL) is a collaborative training architecture where multiple data-owners (users) jointly train a shared global model without sharing their private data [1]. Instead, users perform training locally on their private datasets, after which the local updates (e.g., gradients) are aggregated by a server to update the global model. By eliminating the need for sharing sensitive data, FL has found applications in a variety of privacy-sensitive fields, such as healthcare. On the other hand, recent works have shown that even the local updates may reveal extensive information about sensitive datasets, through what is known as gradient inversion attacks [2]–[4]. Secure aggregation (SA) protocols aim to mitigate such information leakage, by allowing the server to aggregate the local updates without observing them in the clear [5]–[10].

Existing SA protocols consider a homogeneous setting where all users have equal computational resources, and the size of the local model trained by all users is as large as the global model. This imposes a major challenge in real-world applications, where user devices have different computation capabilities, further exacerbated by the fact that model sizes continue to grow in practice, preventing resource-limited devices from participating in training. In this work, our goal is to address this challenge, by developing an SA framework for networks with heterogeneous compute capabilities.

Several recent works have considered FL (without SA) under heterogeneous compute resources [11]–[15]. References [11], [12] leverage bidirectional knowledge distillation where users train a smaller model than the server. References [13]–[16] train partial submodels extracted from the original global model, where the size of the submodels are adapted to the on-device computation capabilities of the users. These existing works consider training without SA, where the server has access to individual local model updates.

In contrast, our goal is to enable submodel training for SA, where the server can only learn the aggregate of the submodels, without learning any further information about the selected submodels. To that end, we first demonstrate the vulnerabilities of existing SA frameworks for submodel aggregation, and the necessity to hide not only the submodel parameters, but also the submodels selected by each user. We then propose a novel SA framework, SESA (Secure Submodel Aggregation), to enable secure gradient aggregation in networks with heterogeneous compute resources. SESA enables each user to train a smaller model extracted from the global model according to their resource availability, while hiding both the selected submodels by each user, and the individual parameters for the selected submodels, under formal information-theoretic privacy guarantees. In doing so, our framework enables adaptability for resource-limited users to train and communicate smaller partial models, reducing both the computation and communication overhead simultaneously.

Recent works [17]–[19] consider the privacy of submodel indices in the context of Private Information Retrieval (PIR). Different from our work, the PIR task builds on a multi-server setting, where the servers do not collude with the users. In contrast, our work is built on a single-server FL setup, where the goal is to ensure privacy against any collusions across up to T users and the server.

Our contributions are summarized as follows:

- We demonstrate the vulnerabilities of conventional SA protocols for submodel training, due to the heterogeneity between the submodels selected by different users.
- We propose the first SA framework that simultaneously tackles heterogeneous computation and communication resource limitations of the users.
- We demonstrate the formal information-theoretic privacy guarantees for hiding both the individual local model updates and the selected submodels by each user.
- Our theoretical analysis demonstrates the adaptability of SESA to resource heterogeneity across the users, in terms of both the computation and communication load.
- Our experiments demonstrate that our SA framework

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.

achieves comparable performance to state-of-the-art submodel training benchmarks (without SA).

II. PROBLEM FORMULATION

We consider a network of N users, where user $i \in [N]$ holds a local dataset $\mathcal{D}_i$. In conventional FL, users have equal compute and computational resources, and the goal is to train a global model $\mathbf{w} \in \mathbb{R}^d$ to minimize a (global) loss function,

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

where $\omega_i \triangleq \frac{|\mathcal{D}_i|}{\sum_{i=1}^{N}|\mathcal{D}_i|}$, and

$$F_i(\mathbf{w}) \triangleq \frac{1}{|\mathcal{D}_i|} \sum_{\xi \in \mathcal{D}_i} f(\xi, \mathbf{w}) \tag{2}$$

denotes the local loss function of user i where $f(\xi, \mathbf{w})$ is the loss computed on the data sample ξ.

Submodel training. Unlike conventional FL, in real-world settings users may have heterogeneous computational resources. Submodel training is a recent framework to address this challenge, where users can train models with varying sizes, adapted to their resource availability [14], [15]. To do so, each user $i \in [N]$ extracts a partial model $\mathbf{w}_i$ from the global model $\mathbf{w}$. Then, the local loss function of user i is defined as,

$$F_i(\mathbf{w}_i) \triangleq \frac{1}{|\mathcal{D}_i|} \sum_{\xi \in \mathcal{D}_i} f(\xi, \mathbf{w}_i), \tag{3}$$

The partial model $\mathbf{w}_i$ is determined by the resource availability of user i, quantified by a parameter B_i denoting the fraction of nodes extracted from each fully connected layer in the global model $\mathbf{w}$ [15]. For convolutional layers, this represents the fraction of the total number of kernels selected within each layer of the global model $\mathbf{w}$. In doing so, users with high computational resources can train larger models by extracting a larger partial model, whereas resource-limited users can train smaller models by choosing a smaller partial model.

At each training round $t \in [J]$, the server then sends the current state $\mathbf{w}^t$ of the global model $\mathbf{w}$ to the N users. After receiving the global model, user $i \in [N]$ extracts a partial model $\mathbf{w}_i^t$, consisting of the extracted parameters across different layers, according to its resource constraint B_i. Next, user i updates the partial model through local training,

$$\mathbf{w}_i^t \leftarrow \mathbf{w}_i^t - \eta \nabla F_i(\mathbf{w}_i^t) \tag{4}$$

where η is the learning rate and $\nabla F_i(\mathbf{w}_i^t)$ denotes the gradient computed with respect to the partial model $\mathbf{w}_i^t$. Let $\mathcal{I}_i^t$ denote the ordered set of the indices of the extracted parameters by user i at round t. After E rounds of local training, user i computes the model difference for the extracted parameters,

$$\mathbf{g}_i^t(k) \triangleq \mathbf{w}_i^t(k) - \mathbf{w}^t(k) \tag{5}$$

for all $k \in \mathcal{I}_i^t$, where $\mathbf{v}(k)$ denotes the k^{th} parameter of a vector $\mathbf{v}$. Then, user i sends $\mathbf{g}_i^t(k)$ for all $k \in \mathcal{I}_i^t$ to the server. In doing so, some users may drop out from the network due to various reasons, including poor channel conditions, limited battery, or device unavailability. After receiving the parameter updates in (5) from the surviving users $i \in \mathcal{U}^t \subseteq [N]$, the server updates the global model for the next training round,

$$\mathbf{w}^t(k) = \mathbf{w}^t(k) - \frac{1}{N} \sum_{i \in \mathcal{S}_k^t \cap \mathcal{U}^t} \mathbf{g}_i^t(k) \tag{6}$$

for all $k \in [d]$, where $\mathcal{S}_k^t \triangleq \{i \in [N] : k \in \mathcal{I}_i^t\}$ denotes the set of users who update the k^{th} parameter at round t.

Threat model. We consider an honest-but-curious adversary model along the line of conventional SA protocols, where adversaries follow the training protocol truthfully, but try to extract confidential information about honest users' local data through the messages exchanged [5]–[8]. We consider up to T adversarial users, who may collude with one another and/or the server. The set of adversarial and honest users are denoted as $\mathcal{T}$ and $\mathcal{H} \triangleq [N] \backslash \mathcal{T}$, respectively.

Secure aggregation. SA aims to aggregate the local updates $\mathbf{g}_i^t$, without revealing any information beyond their sum,

$$\mathbf{g}_{agg}^t \triangleq \sum_{i \in \mathcal{U}^t} \mathbf{g}_i^t \tag{7}$$

Formally, this condition is expressed as,

$$I(\{\mathbf{g}_i^t\}_{i \in \mathcal{H}}; \mathcal{M}_{\mathcal{T}}^t | \mathbf{g}_{agg}^t, \{\mathbf{g}_i^t\}_{i \in \mathcal{T}}, \mathcal{R}_{\mathcal{T}}^t) = 0 \tag{8}$$

where $\mathcal{M}_{\mathcal{T}}^t$ denotes the received messages, and $\mathcal{R}_{\mathcal{T}}^t$ is the randomness generated, by the adversaries and the server at round t. The correct recovery of (7) is ensured by the constraint,

$$H(\mathbf{g}_{agg}^t | \mathcal{M}_{\mathcal{U}^t}^t) = 0 \tag{9}$$

where $\mathcal{M}_{\mathcal{U}^t}^t$ encompasses the set of messages held by the surviving users $\mathcal{U}^t$ at round t. To compute (7) under the privacy guarantees from (8), users *encode* their local updates $\mathbf{g}_i^t$, sending instead the encoded version to the server. The encoding process obscures the true user updates from the server using locally generated random masks, while still allowing the server to decode their sum as in (7), without gaining any information about the individual updates $\mathbf{g}_i^t$. While doing so, SA protocols differ in their encoding/decoding process [5]–[8].

A major challenge of SA is its computational overhead; the dimensionality of the model trained by each user is as large as the global model, limiting scalability to resource-limited networks with heterogeneous compute capabilities. We aim to tackle this challenge by posing the question,

- *Can submodel training enhance the scalability of SA to networks with heterogeneous compute resources?*

In this work, we show that submodel training can enable SA in networks with heterogeneous compute resources, but additional care should be taken to anonymize the updated submodels, and naive approaches can be detrimental to the security premise of SA. As discussed later in Section III, SA is vulnerable to submodel training; varying submodel dimensions across the users allow adversaries to reconstruct local data samples of honest users even in the presence of SA. Our results emphasize the necessity for stronger security guarantees for submodel training, concealing not only the local submodels,

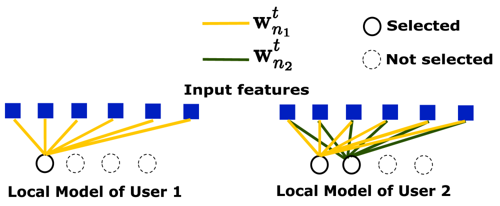


Fig. 1. Overview of an attack scenario with $N = 2$ users.

but also the indices of updated submodel parameters:

$$I(\{\mathbf{g}_i^t(k), k\}_{\substack{i\in\mathcal{H}\\ k\in\mathcal{I}_i^t}}; \mathcal{M}_\mathcal{T}^t | \{\sum_{i\in\mathcal{S}_k^t\cap\mathcal{U}^t} \mathbf{g}_i^t(k)\}_{k\in[d]}, \{\mathbf{g}_i^t(k), k\}_{\substack{i\in\mathcal{T}\\ k\in\mathcal{I}_i^t}}, \mathcal{R}_\mathcal{T}) = 0 \tag{10}$$

To address this challenge, we introduce SESA (Secure Submodel Aggregation), a secure aggregation framework for submodel training. Our framework enables users to flexibly train partial models based on their available system resources, while satisfying (10). In doing so, SESA hides both the trained submodels and indices of the updated submodel parameters from the server, ensuring the formal information-theoretic privacy guarantees from (10), while significantly enhancing both the compute and communication efficiency of SA.

III. Reconstruction Attacks

SA frameworks are built upon various information-theoretic primitives, while all are designed to achieve the same goal of recovering the sum of the local parameters without revealing the individual parameters. A naive approach to extend the existing frameworks to submodel training is to perform SA parameter-wise. On the other hand, as we demonstrate next, by leveraging the heterogeneity between the submodels, the server can perfectly reconstruct the sensitive training samples.

Consider a network of $N = 2$ users as shown in Fig. 1, with the resource constraints $B_1 = 1/4$ and $B_2 = 1/2$. Let $\mathbf{x}_i \in \mathbb{R}^m$ denote the training sample of user i, which is the input to a fully connected layer consisting of 4 nodes n_1, n_2, n_3, n_4. Without loss of generality, at a given training round t, assume that user 1 selects node n_1 and user 2 selects nodes n_1, n_2. Denote the global weight parameters corresponding to node n_j as $\mathbf{w}_{n_j}^t \in \mathbb{R}^m$ and the bias parameter as $z_{n_j}^t$. As the server knows the submodel indices of the users, it can identify that the aggregate of the gradient parameters corresponding to node n_2 is equal to the local gradient parameter of user 2. Similarly, the aggregate of the gradient parameters corresponding to node n_1 is sum of the local gradient parameters of users 1 and 2. Then, the server can recover the input training samples of both users as follows.

For user 2, the output of node n_2 can be written as,

$$y_{2,n_2}^t \triangleq (\mathbf{w}_{n_2}^t)^\mathrm{T}\mathbf{x}_2 + z_{n_2}^t \tag{11}$$

hence, the gradients with respect to the weight $\mathbf{w}_{n_2}^t$ and bias $z_{n_2}^t$ are given by,

$$\frac{\partial F_2}{\partial(\mathbf{w}_{n_2}^t)^\mathrm{T}} = \frac{\partial F_2}{\partial y_{2,n_2}^t}\frac{\partial y_{2,n_2}^t}{\partial(\mathbf{w}_{n_2}^t)^\mathrm{T}} = \frac{\partial F_2}{\partial y_{2,n_2}^t} \times \mathbf{x}_2^\mathrm{T} \tag{12}$$

$$\frac{\partial F_2}{\partial z_{n_2}^t} = \frac{\partial F_2}{\partial y_{2,n_2}^t}\frac{\partial y_{2,n_2}^t}{\partial z_{n_2}^t} = \frac{\partial F_2}{\partial y_{2,n_2}^t} \times 1 \tag{13}$$

By combining (12) and (13), one can recover the input sample,

$$\mathbf{x}_2^\mathrm{T} = \frac{\partial F_2}{\partial(\mathbf{w}_{n_2}^t)^\mathrm{T}} / \frac{\partial F_2}{\partial z_{n_2}^t}. \tag{14}$$

Therefore, by leveraging the gradient parameters corresponding to node n_2 sent by user 2, the server can reconstruct the training sample of user 2. Then, the server can use the recovered sample of user 2 to find the gradient parameters of user 2 corresponding to node n_1, i.e., $\frac{\partial F_2}{\partial(\mathbf{w}_{n_1}^t)^\mathrm{T}}$ and $\frac{\partial F_2}{\partial z_{n_1}^t}$. Using the aggregate of the gradients (of two users) obtained through SA, the server can recover the gradient of user 1,

$$\frac{\partial F_1}{\partial(\mathbf{w}_{n_1}^t)^\mathrm{T}} = \sum_{i\in[2]} \frac{\partial F_i}{\partial(\mathbf{w}_{n_1}^t)^\mathrm{T}} - \frac{\partial F_2}{\partial(\mathbf{w}_{n_1}^t)^\mathrm{T}} \tag{15}$$

$$\frac{\partial F_1}{\partial z_{n_1}^t} = \sum_{i\in[2]} \frac{\partial F_i}{\partial z_{n_1}^t} - \frac{\partial F_2}{\partial z_{n_1}^t} \tag{16}$$

Finally, by using the gradient obtained from (15) and (16), the server can also recover the training sample $\mathbf{x}_1$ of user 1 as in (14). Hence, disclosing the submodel parameter indices can reveal sensitive training samples, undermining the key premise of SA. In the following, we introduce a secure submodel aggregation framework, SESA, to address this challenge. Our framework enables submodel aggregation without disclosing neither the submodel parameters nor their indices.

IV. Secure Submodel Aggregation (SESA)

We next describe the individual steps of our framework.

Submodel Selection. We consider a random submodel extraction scheme, where the global model is divided into multiple submodels, and each user selects a number of submodels in accordance with their compute capability. To that end, we let L denote the total number of layers[2]. The nodes within each layer are divided into $1/B_{\text{min}}$ shards, where

$$B_{\text{min}} \triangleq min_{i\in[N]}\{B_i\} \tag{17}$$

denotes the resource constraint of the user with the lowest compute capability. A submodel is then defined as the set of all pairwise connections across two distinct shards between two consecutive layers. Accordingly, the total number of submodels between any two layers $(l, l+1)$ for $l \in [L]$ is,

$$K \triangleq 1/B_{\text{min}}^2 \tag{18}$$

From each layer $l \in [L]$, user $i \in [N]$ then selects B_i/B_{min} shards uniformly at random (without replacement), corresponding to a total number of,

$$K_i \triangleq B_i^2/B_{\text{min}}^2 \tag{19}$$

[2]For ease of exposition, we describe our framework using fully connected layers, while noting that the same principles can be applied also to convolutional layers, in which case kernels can be used to form submodels.

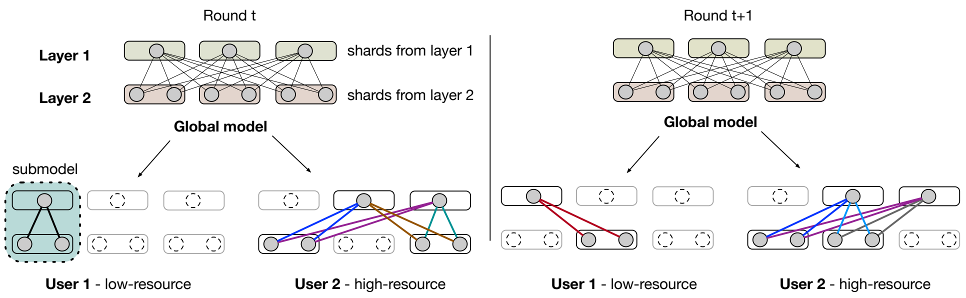


Fig. 2. Overview of submodel extraction with $N = 2$ users. The resource constraints of the users are given as $B_1 = 1/3, B_2 = 2/3$. The nodes in each layer are divided into $1/B_1 = 3$ shards according to (17). The user with low computational power (user 1) selects $B_1/B_1 = 1$ shard, whereas the user with higher computational power (user 2) selects $B_2/B_1 = 2$ shards from each layer. The pairwise connections between any two shards selected across two consecutive layers represents a submodel selected by the user (each submodel is represented by a distinct color).

submodels. After submodel selection, user i extracts the selected submodels from the global model $\mathbf{w}^t$, updates them through local training as shown in (4), and forms the submodel update $\mathbf{g}_i^t$ as in (5). We provide an illustrative example in Fig. 2 for a network of $N = 2$ users.

Secure Aggregation. After submodel selection, our goal is to sum the submodel updates as in (6), while ensuring that the server cannot learn anything about which submodels are selected by the individual users as shown in (10).

Our framework consists of an offline and an online phase. The former is data-independent such as randomness generation, which can be executed in advance when the network load is low. The latter is data-dependent, including local training and model update. Then, the following operations are performed layer-wise (independently) to aggregate the submodels selected within each pair of consecutive layers $(l, l+1)$ for $l \in [L]$. For simplicity, in the following we omit the layer index l, while noting that a new set of randomness is generated for each layer. All operations are performed in a finite field $\mathbb{F}_p$ of integers modulo a large prime p.

Offline. Initially, users agree on $N + K + T$ distinct public parameters $\{\alpha_i\}_{i\in[N]}$, $\{\beta_n\}_{n\in[K+T]}$ from $\mathbb{F}_p$. Let $\mathcal{K}_i^t$ denote the ordered set of the submodel indices selected by user i, where $|\mathcal{K}_i^t| = K_i$, and $\mathcal{K}_i^t(k) \in [K]$ denotes the index of the k^{th} submodel selected. Then, user i defines a binary vector $\mathbf{b}_{ik}^t \in \{0,1\}^K$ for $k \in [K_i]$, with the n^{th} element given by,

$$\mathbf{b}_{ik}^t(n) = \begin{cases} 1 & \text{if} \quad n = \mathcal{K}_i^t(k) \\ 0 & \text{otherwise} \end{cases}, \tag{20}$$

and constructs a Lagrange polynomial of degree $K + T - 1$,

$$\phi_{ik}^t(\alpha) \triangleq \sum_{n\in[K]} \mathbf{b}_{ik}^t(n) \prod_{n'\in[K+T]\setminus\{n\}} \frac{\alpha - \beta_{n'}}{\beta_n - \beta_{n'}} + \sum_{n=K+1}^{K+T} u_{ikn}^t \prod_{n'\in[K+T]\setminus\{n\}} \frac{\alpha - \beta_{n'}}{\beta_n - \beta_{n'}}, \tag{21}$$

for all $k \in [K_i]$, and $u_{ikn}^t \in \mathbb{F}_p$ for all $k \in [K_i], n \in \{K + 1, \dots, K + T\}$ are selected uniformly at random. Then, user i generates K_i uniformly random masks $\mathbf{r}_{ik}^t \in \mathbb{F}_p^{\frac{d'}{K}}$ for $k \in [K_i]$, where d' is the dimension of the model for the given layer, and constructs a Lagrange polynomial of degree $K + T - 1$,

$$\varphi_{ik}^t(\alpha) \triangleq \sum_{n\in[K]} \mathbf{b}_{ik}^t(n)\mathbf{r}_{ik}^t \prod_{n'\in[K+T]\setminus\{n\}} \frac{\alpha - \beta_{n'}}{\beta_n - \beta_{n'}} + \sum_{n=K+1}^{K+T} \mathbf{v}_{ikn}^t \prod_{n'\in[K+T]\setminus\{n\}} \frac{\alpha - \beta_{n'}}{\beta_n - \beta_{n'}}, \tag{22}$$

for each $k \in [K_i]$ where $\mathbf{v}_{ikn}^t \in \mathbb{F}_p^{d'/K}$ for all $k \in [K_i]$ and $n \in \{K+1, \dots, K+T\}$ are selected uniformly random. Finally, user i sends a distinct evaluation of the polynomials $\{\phi_{ik}^t(\alpha_j), \varphi_{ik}^t(\alpha_j)\}_{k\in[K_i]}$ to each user $j \in [N]$.

Online. In the online phase, after local training, each user $i \in [N]$ initially generates a finite field representation of their K_i local submodel updates, denoted by $\overline{\mathbf{g}}_{ik}^t \in \mathbb{F}_p^{d'/K}$ for $k \in \mathcal{K}_i^t$. This finite field transformation is a standard primitive in secure multi-party computing frameworks [9], [10], [20]–[23]. For the details of this transformation, we refer to [10], [21], [24], [25]. User i then broadcasts a masked submodel update,

$$\mathbf{x}_{ik}^t \triangleq \overline{\mathbf{g}}_{i,\mathcal{K}_i^t(k)}^t - \mathbf{r}_{ik}^t \tag{23}$$

for all $k \in [K_i]$. After receiving the masked submodels from the set of surviving users $i \in \mathcal{U}^t$, each user $i \in \mathcal{U}^t$ computes,

$$\psi^t(\alpha_i) \triangleq \sum_{j\in\mathcal{U}^t} \sum_{k\in[K_j]} (\mathbf{x}_{jk}^t \phi_{jk}^t(\alpha_i) + \varphi_{jk}^t(\alpha_i)) \tag{24}$$

$$= \sum_{j\in\mathcal{U}^t} \sum_{k\in[K_j]} \Big(\overline{\mathbf{g}}_{j,\mathcal{K}_j^t(k)}^t \prod_{n'\in[K+T]\setminus\{\mathcal{K}_j^t(k)\}} \frac{\alpha_i - \beta_{n'}}{\beta_{\mathcal{K}_j^t(k)} - \beta_{n'}} + \sum_{n=K+1}^{K+T} (\overline{\mathbf{g}}_{j,\mathcal{K}_j^t(k)}^t u_{jkn}^t - \mathbf{r}_{jk}^t u_{jkn}^t + \mathbf{v}_{jkn}^t) \prod_{n'\in[K+T]\setminus\{n\}} \frac{\alpha_i - \beta_{n'}}{\beta_n - \beta_{n'}}\Big), \tag{25}$$

and sends $\psi^t(\alpha_i)$ to the server, which corresponds to a distinct evaluation of the degree $K + T - 1$ polynomial,

$$\psi^t(\alpha) = \sum_{j\in\mathcal{U}^t} \sum_{k\in[K_j]} \Big(\overline{\mathbf{g}}_{j,\mathcal{K}_j^t(k)}^t \prod_{n'\in[K+T]\setminus\{\mathcal{K}_j^t(k)\}} \frac{\alpha - \beta_{n'}}{\beta_{\mathcal{K}_j^t(k)} - \beta_{n'}} + \sum_{n=K+1}^{K+T} (\overline{\mathbf{g}}_{j,\mathcal{K}_j^t(k)}^t u_{jkn}^t - \mathbf{r}_{jk}^t u_{jkn}^t$$

$$+ \mathbf{v}^t_{jkn}\Big) \prod_{n' \in [K+T] \setminus \{n\}} \frac{\alpha - \beta_{n'}}{\beta_n - \beta_{n'}}\Big) \quad (26)$$

$$= \sum_{k \in [K]} \sum_{j \in \mathcal{U}^t \cap \mathcal{S}^t_k} \Big(\bar{\mathbf{g}}^t_{jk} \prod_{n' \in [K+T] \setminus \{k\}} \frac{\alpha - \beta_{n'}}{\beta_k - \beta_{n'}} + \sum_{n=K+1}^{K+T} (\bar{\mathbf{g}}^t_{jk} u^t_{jkn} - \mathbf{r}^t_{jk} u^t_{jkn} + \mathbf{v}^t_{jkn}) \prod_{n' \in [K+T] \setminus \{n\}} \frac{\alpha - \beta_{n'}}{\beta_n - \beta_{n'}}\Big), \quad (27)$$

at $\alpha = \alpha_i$, where $\mathcal{S}^t_k \triangleq \{i \in [N] : k \in \mathcal{K}^t_i\}$ denotes the set of all users that have selected submodel k between layers $(l, l+1)$. Therefore, after receiving at least $K+T$ evaluations, the server can reconstruct $\psi^t(\alpha)$ using polynomial interpolation, and recover the aggregate of the submodel updates,

$$\sum_{i \in \mathcal{S}^t_k \cap \mathcal{U}^t} \bar{\mathbf{g}}^t_{ik} = \psi^t(\beta_k) \qquad \forall k \in [K]. \quad (28)$$

After aggregating the submodels for each pair of layers $(l, l+1) \in [L]$, the server updates the global model as in (6).

V. Theoretical Analysis

We next present the communication and computation complexity, dropout tolerance, and privacy guarantees of SESA. Dropout tolerance is defined by the maximum number of dropped users the system can tolerate while still ensuring correct recovery of the aggregate of the submodels.

Theorem 1. *(Communication complexity) The communication overhead of SESA for user $i \in [N]$ is $O(K_i \frac{d}{K})$ online, and $O(NK_i \frac{d}{K})$ offline.*

Remark 1. *The online communication overhead of user i scales with the number of submodels K_i, which is controlled by the resource constraint B_i from* (19)*. Hence, SESA can reduce not only the training load, but also the communication overhead for resource-limited users, which can be useful in practice as such users often have limited communication capabilities also (e.g., edge devices vs. data centers).*

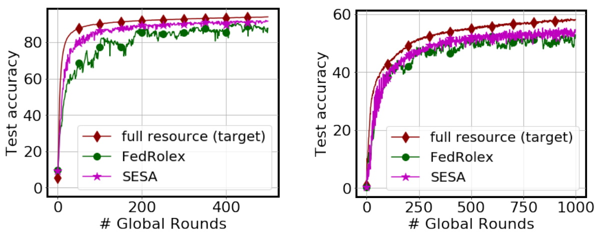
Theorem 2. *(Computation complexity) The computation overhead of SESA for user i is $O(Nd)$ online, $O(NK_i \frac{d}{K} \log^2(K+T) \log\log(K+T))$ offline.*

Theorem 3. *(Dropout tolerance) The maximum number of user dropouts that SESA can tolerate is $D \leq N - (K+T)$.*

Theorem 4. *(Privacy) SESA ensures the information-theoretic privacy of the local submodels selected by the honest users against any set of up to T adversarial users and the server,*

$$I\Big(\{\bar{\mathbf{g}}^t_{ik}\}_{\substack{i \in \mathcal{H} \\ k \in \mathcal{K}^t_i}}, \{\mathcal{K}^t_i\}_{i \in \mathcal{H}}; \mathcal{M}^t_{\mathcal{T}} \Big| \Big\{ \sum_{i \in \mathcal{S}^t_k \cap \mathcal{U}^t} \bar{\mathbf{g}}^t_{ik} \Big\}_{k \in [K]}, \{\bar{\mathbf{g}}^t_{ik}\}_{\substack{i \in \mathcal{T} \\ k \in \mathcal{K}^t_i}}, \{\mathcal{K}^t_i\}_{i \in \mathcal{T}}, \mathcal{R}_{\mathcal{T}}\Big) = 0 \quad (29)$$

where $\mathcal{M}_{\mathcal{T}}$ and $\mathcal{R}_{\mathcal{T}}$ denote the set of messages received and randomness generated by the adversaries, respectively.



(a) MNIST (b) FEMNIST

Fig. 3. Performance evaluation of SESA with respect to the benchmarks.

VI. Experiments

We evaluate the performance of SESA compared to both the state-of-the-art resource-aware submodel training protocol, FedRolex [15], as well as conventional FL without any resource limitations, where all users can train the entire global model (which serves as our target accuracy). For best baseline accuracy, both benchmarks are applied without SA, as the finite field conversion in SA can degrade model performance.

We consider a FL task in a network of $N = 100$ users for image classification on the MNIST [26] and FEMNIST [27] datasets, distributed across the users with respect to the non-i.i.d. data distributions from [1] and [27], respectively. The model architecture includes two fully connected layers with one hidden layer with 200 nodes between the input and output layer. For resource heterogeneity, users are partitioned into 3 groups $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$ uniformly random (without replacement), where $|\mathcal{G}_1| = |\mathcal{G}_2| = \lfloor \frac{N}{3} \rfloor$ and $|\mathcal{G}_3| = \lfloor \frac{N}{3} \rfloor + 1$. Each group of users has a different resource constraint, given as $B_i = 1$ for $i \in \mathcal{G}_1$, $B_i = 1/2$ for $i \in \mathcal{G}_2$, and $B_i = 1/4$ for $i \in \mathcal{G}_3$. At each training round, $D = 10\%$ of the users drop out uniformly random, as user dropout rates vary between 0.06 and 0.1 in real-world settings [28]. The total number of adversarial users is $T = \frac{N}{2}$ [5]. The size of the finite field is set to $p = 2^{32} - 5$.

In Fig. 3, we present the test accuracy of SESA and the benchmarks. For a fair comparison, we use the same resource allocation and hyperparameters for both FedRolex and SESA. We observe that both frameworks reach a comparable accuracy at convergence, while SESA has a faster convergence rate. This could be due to the cyclic submodel selection mechanism employed by the former, where users select a fraction of nodes which are advanced by one node at each training round. For instance, a user with resource constraint $B_i = 1/4$ initially selects 50-out-of-200 nodes from the hidden layer and then rolls over through all the nodes by advancing one node at a time (per round). As a result, covering the updates from all 200 nodes requires 150 training rounds. Finally, SESA achieves comparable accuracy to the target benchmark with no resource limitations, which is denoted as "*full resource (target)*".

VII. Conclusion

In this work, we propose SA for submodel training for networks with heterogeneous computation and communication capabilities. Our framework ensures a personalized training computation and communication load, adapted to the resource availability of the individual users, while guaranteeing information-theoretic privacy for the local submodels.

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, pp. 1273–1282.

[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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 332–16 341.

[5] 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 (CCS)*, 2017, pp. 1175–1191.

[6] 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.

[7] 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," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 694–720, 2022.

[8] 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.

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

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

[11] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large cnns at the edge," in *Advances in Neural Information Processing Systems (NeurIPS)*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[12] Y. J. Cho, A. Manoel, G. Joshi, R. Sim, and D. Dimitriadis, "Heterogeneous ensemble knowledge transfer for training large models in federated learning," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, 2022, pp. 2881–2887.

[13] S. Caldas, J. Konečný, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *Arxiv*, 2018.

[14] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," in *International Conference on Learning Representations, ICLR*, 2021.

[15] S. Alam, L. Liu, M. Yan, and M. Zhang, "Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[16] S. Horváth, S. Laskaridis, M. Almeida, I. Leontiadis, S. I. Venieris, and N. D. Lane, "Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout," in *Advances in Neural Information Processing Systems (Neurips)*, 2021, pp. 12 876–12 889.

[17] M. Kim and J. Lee, "Information-theoretic privacy in federated submodel learning," *ICT Express*, vol. 9, no. 3, pp. 415–419, 2023.

[18] 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.

[19] Z. Jia and S. A. Jafar, "X-secure t-private federated submodel learning with elastic dropout resilience," *IEEE Trans. Inf. Theory*, 2022.

[20] 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.

[21] 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.

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

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

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

[25] 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.

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

[27] 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.

[28] 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.