Dual-Server Based Lightweight Privacy-Preserving Federated Learning

Liangyu Zhong, Lulu Wang, Lei Zhang, *Member, IEEE,* Josep Domingo-Ferrer, *Fellow, IEEE,* Burong Kang and Rui Zhang

***Abstract*—Federated learning (FL) allows multiple users to collaboratively train global machine learning models by keeping their data sets local. However, the existing privacy-preserving FL schemes suffer from several limitations, e.g., loss of accuracy, high communication/computation cost, failure to support dynamic users, and insecurity against collusion attacks. To solve these limitations, we propose a lightweight privacy-preserving FL scheme based on a dual-server architecture. Our scheme involves only lightweight cryptographic operations, i.e., hash and symmetric encryption operations, and it has low communication overhead. Thus, it is computationally lightweight and round-efficient. Further, it allows users to join/quit an FL task and it is accuracy-lossless. We formally prove that our scheme remains secure even in case of collusion attacks. In particular, if an attacker colludes with one of the servers and all the users who participate in an FL task except two, the privacy of user gradients stays unviolated.**

***Index Terms*—Privacy preservation, lightweight cryptography, secure aggregation, federated learning.**

I. Introduction

Machine learning (ML) is playing an increasingly important role in AI due to its widespread applicability to very diverse areas such as, e.g., medical research [32] or autonomous driving [10]. ML is used to make predictions, classifications and other analyses. The mainstream approach to obtaining a high-performance ML model is to train it on a centralized data set resulting from the collection of large amounts of data. However, in many scenarios, data are often distributed among different users who are unwilling to share their own data sets or cannot share them due to legal reasons. For example, hospitals are usually reluctant to share their data sets with a third party for model training, because they must protect the privacy of their patients. Thus, training ML models without centralizing data sets from multiple users is a very relevant endeavor.

For the above reasons, federated learning (FL) [20], [24], [28], [35] has gained great attention from both academia and industry. A typical FL architecture consists of a server (usually called aggregation server) and a set of users with their respective data sets whose aim is to collaboratively train a high-performance ML model, known as global model, through multiple rounds of joint local training by the users, who do not exchange their raw local data. In each training round, each user obtains the latest global model from the server and trains it on its local data set to obtain its local model and the corresponding local gradient. The local gradients of the users are also known as model updates and users send them to the server to generate an updated global model. Although FL avoids the need for users to share their local data, the local gradients sent by a user to the server may leak sensitive information on the user's data set [7], [21], [29], [39]. Thus, the gradients of honest users should be protected. This paper focuses on gradient privacy.

Designing a privacy-preserving FL scheme that protects gradient privacy is a challenging task, especially if one wants the scheme to be collusion-resistant and efficient in terms of computation and communications. We note that most of the existing proposals [4], [22], [25], [30], [31] are not computationally lightweight or round-efficient, since they rely on costly cryptographic techniques. Some schemes certainly achieve efficient privacy-preserving FL by adding noise to the users' local gradients, but this results in a loss of accuracy of the final global model. We also observe that a privacy-preserving FL scheme should be able to accommodate the dynamic nature of user participation: indeed, new users may join the FL task and existing users may quit.

A. Related Work

In existing privacy-preserving FL schemes, gradient privacy is usually addressed using differential privacy (DP) or cryptographic schemes, i.e., secure multi-party computation (MPC).

In DP-based privacy-preserving FL schemes [1], [2], [8], [12], [14], [18], [33], [36], each user adds random noise to its local gradient before sending it to a server for aggregation. As discussed above, noise injection in DP-based privacy-preserving FL causes loss of accuracy.

In MPC-based schemes [4], [11], [25], [37], users can jointly train a global model without revealing intermediate information during the whole training process. Most MPC-based schemes assume there exists a single server that iteratively collects users' local gradients and updates the global model. In existing proposals, costly cryptographic primitives are usually employed, e.g., garbled circuits [13], [23], [27], secret sharing [15], [26] or homomorphic encryption [37], [38]. For instance, the scheme in [4] applies secret sharing to ensure gradient privacy and it requires four rounds of interaction among the users and the server within each training round. This leads to high communication and computation costs. Another example is [25], which uses the Paillier additive homomorphic

Liangyu Zhong, Lulu Wang, Lei Zhang, Burong Kang and Rui Zhang are with the Engineering Research Center of Software/Hardware Co-design Technology and Application, Ministry of Education, East China Normal University, Shanghai 200062, China. Lei Zhang is the corresponding author.

Josep Domingo-Ferrer is with the Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Av. Països Catalans 26, 43007 Tarragona, Catalonia. E-mail josep.domingo@urv.cat

encryption scheme and incurs significant message expansion as well as high computation and communication overheads.

The dual-server privacy-preserving FL architecture was recently introduced to address poisoning attacks in FL [19]. This scheme assumes two non-colluding servers, but it is not round-efficient and it employs costly cryptographic techniques. Thus, it is not suited for large-scale FL. We notice that most existing privacy-preserving FL schemes are designed in the semi-trusted model, in which the users and the server(s) are assumed to be honest but curious. In this paper, we also focus on security in the semi-trusted model. We also note that a notion related to FL is collaborative model learning (CML). Recently, several dual-server privacy-preserving CML schemes have been proposed [30], [31]. However, unlike FL, they require the users' data to be uploaded to the servers in an encrypted form, which results in high communication cost. Further, these proposals rely on costly cryptographic primitives, e.g., homomorphic encryption, oblivious transfer and/or secret sharing. Therefore, these CML schemes are not lightweight.

B. Our Contribution

The existing privacy-preserving FL schemes suffer from high communication/computation cost, accuracy loss, failure to support dynamic users, and insecurity against collusion attacks. We propose a lightweight privacy-preserving FL scheme to address these limitations.

In our scheme, we apply a dual-server FL architecture with two servers S_α and S_β. These servers and the users are assumed to be semi-trusted. An FL task, aimed at training a global model and consisting of multiple training rounds, is initialized by S_β. Each user can decide to join or quit an FL task at any time. When a user joins the task for the first time, it should establish a shared secret key using a secure key agreement scheme with each of the servers. Then, in each training round, each user can train its local model and generate a local gradient based on its data set and the latest global model. The gradient of each user is partitioned into two shares that are then masked. A user's mask is generated using its shared secret key and a hash function, and the combined length of the two masked shares is the same as that of the gradient. The user sends one masked share to S_α and the other to S_β. Each server aggregates the masked shares into a masked aggregated share. S_α sends its masked aggregated share to S_β. With S_α's masked aggregated share and that generated by itself, S_β generates an updated global model via de-masking, which it securely sends to the users by protecting it under a secure symmetric encryption scheme.

Our scheme is lightweight, since it relies on hash functions and symmetric encryption to guarantee gradient privacy, and there is almost no message expansion. A key agreement scheme is certainly needed, but only used when a user joins an FL task for the first time. Further, multiple rounds of interactions among the users and the servers are *not* required within each training round. Therefore, our scheme is also round-efficient. Further, our scheme is dynamic because it allows users to join or quit an FL task at any training round. Our scheme is also lossless in terms of accuracy, because no noise is added to users' gradients. The above claimed properties are demonstrated through comparisons and experiments.

As to privacy, we use a simulation-based approach to prove that our scheme ensures gradient privacy even under collusion attacks. In particular, we prove that gradient privacy stays unviolated even if an attack involves a collusion of one of the two servers with all the users who participate in an FL task except two.

C. Organization

The remainder of this paper is organized as follows. Section II contains background. In Section III, we propose our lightweight privacy-preserving FL scheme. A comprehensive security analysis of the proposed scheme is shown in Section IV. The efficiency of our scheme is evaluated in Section V. Finally, Section VI gathers the conclusions.

II. Background

In this section, we introduce the architecture of our system, our design goals, the required cryptographic primitives, and our notation.

A. System Architecture

Figure 1 illustrates the system architecture, which involves the certification authority (CA), server S_α, server S_β and the users. These entities can be described as follows:

- CA is a trusted authority. It is used to generate the system parameters and issue public-key certificates to the rest of entities in the system (i.e., users and servers), so that they can communicate securely.
- Users hold their respective local data sets. When an FL task is first initialized (by S_β), any user can join the FL task for the joint training of a global model. In each round of an FL task, a user trains its local model and generates a local gradient based on its own data set and the latest global model. Further, the user generates two masked shares of its gradient, one of which it sends to S_α and the other to S_β. Users can join or quit an FL task at any training round.
- S_α collects the masked shares of the gradients from the users at each round of an FL task to generate a masked aggregated share, which it sends to S_β.
- S_β initializes an FL task, i.e., it generates the initial global model and the related parameters. It is also responsible for updating the global model. To do this, S_β aggregates the masked shares of gradients from the users in each round to generate a masked aggregated share. With the masked aggregated share generated by itself and the masked aggregated share from S_α, S_β generates an updated global model. For each round an FL task, the users participating in the task have to obtain the latest global model from S_β.

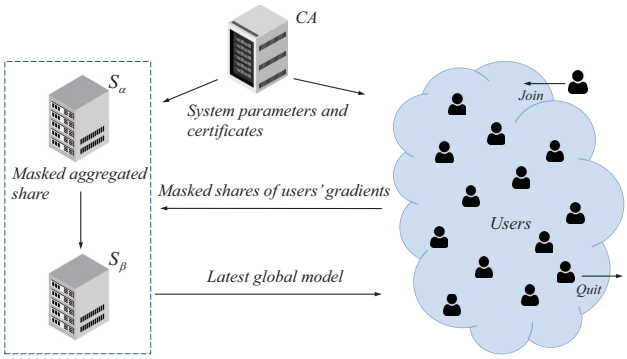


Fig. 1: System architecture

B. Design Goals

We assume the entities (i.e., users and servers) in the system to be semi-trusted and the two servers *not* to collude with each other. However, we do consider collusions among one server and a set of users. The worst attack we consider is one in which an attacker can corrupt one of the two servers and all the users who participate in an FL task except two. Our goal is to design a privacy-preserving FL scheme that is secure against such a large collusion. The scheme has to ensure the following properties:

- *Gradient privacy.* Even in the large collusion attack mentioned above, an attacker should be unable to learn the content of an uncorrupted user's local gradient that is sent to the servers.
- *Lightweight computation.* The efficiency of our scheme should be comparable with that of a plain FL scheme, that is, a scenario where there is a single server and all entities are fully honest/trusted, in which case no security mechanisms are needed.
- *Dynamic user behavior.* An FL task involves multiple rounds to train a global model. In real-word applications, users may join or quit at any moment. A privacy-preserving FL scheme should support user joining or quitting.
- *Round efficiency.* The users and the servers should not be required to run multiple rounds of interaction within each FL training round.
- *Lossless accuracy.* The accuracy of a final global model trained using our scheme should be comparable with that attained using plain FL without security mechanisms among trusted entities.

C. Key Agreement

Our scheme involves a secure key agreement scheme $\mathrm{KA} = (\mathrm{KA.setup}(\lambda), \mathrm{KA.gen}(params, p_i), \mathrm{KA.agree}(p_i, p_j))$ whose algorithms are as follows:

- $\mathrm{KA.setup}(\lambda)$: On input a security parameter λ, this algorithm generates the public parameters $params$.
- $\mathrm{KA.gen}(params, p_i)$: This algorithm allows an entity p_i to generate its private-public key pair $(d_{p_i}^{sk}, d_{p_i}^{pk})$.
- $\mathrm{KA.agree}(p_i, p_j)$: This algorithm allows an entity p_i with private-public key pair $(d_{p_i}^{sk}, d_{p_i}^{pk})$ and an entity p_j with private-public key pair $(d_{p_j}^{sk}, d_{p_j}^{pk})$ to negotiate a shared secret key.

As stated in [6], a secure key agreement scheme should satisfy the key indistinguishability property which guarantees that, given a key in the secret key space of the key agreement scheme, an attacker cannot distinguish whether the key was generated using the above $\mathrm{KA.agree}$ algorithm or is a random value. In other words, the distribution of a shared secret key generated by a secure key agreement scheme is random. In our scheme, a secure key agreement scheme will be used by a user and a server to generate a shared secret key.

D. Hash Functions

Let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\tau$ be a hash function that maps elements x from $\{0,1\}^*$ (binary strings of arbitrary length) into elements y from $\{0,1\}^\tau$ (binary strings of length τ). A secure hash function should satisfy collision resistance.

Definition 1 (Collision Resistance): For any probabilistic polynomial-time adversary $\mathcal{A}$, a hash function $\mathcal{H}$ is collision-resistant if $\mathcal{A}$'s advantage

$$\mathbf{Adv}^{\mathrm{CR}}_{\mathcal{H},\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c}(x, x') \leftarrow \mathcal{A}(\mathcal{H}) : \\ x \neq x' \wedge \mathcal{H}(x) = \mathcal{H}(x')\end{array}\right]$$

is negligible.

It is stated in [3] that cryptographic hash functions are commonly referred to as the ideal realization of random oracles whose output distributions are uniform ones. Therefore, we assume that the output distribution of a hash function is also uniform.

E. Symmetric Encryption Scheme

A symmetric encryption scheme $\mathcal{SE} = (\mathcal{E}_k, \mathcal{D}_k)$ consists of an encryption algorithm $\mathcal{E}_k$ and a decryption algorithm $\mathcal{D}_k$ which are defined as follows:

- $\mathcal{E}$ takes as input a secret key k and a message $m \in \{0,1\}^*$, and outputs a ciphertext c. We denote this by $c \leftarrow \mathcal{E}_k(m)$.
- $\mathcal{D}$ takes as input a secret key k and a ciphertext c, and outputs a message m in the message space or an error symbol $\perp$. We write this as $m = \mathcal{D}(c)$.

A symmetric encryption scheme should satisfy correctness, that is, $\mathcal{D}_k(\mathcal{E}_k(m)) = m$. Further, the ciphertext distribution of a secure symmetric encryption scheme $\mathcal{SE}$ is assumed to be uniformly random.

F. Notations

Let M_1 and M_2 be two matrices that have the same row dimension. We define $M_1 \sqcup M_2$ to be the horizontal matrix concatenation operation.

Let M be a $m \times n$ matrix. $(M_1, M_2) \leftarrow P^l(M)$ is defined to be the matrix partition operation. It partitions M into an $m \times l$ matrix M_1 and an $m \times (n-l)$ matrix M_2 such that $M = M_1 \sqcup M_2$.

Assume the range of the elements in an $m \times n$ matrix is $\mathbb{R}$. We use the notation $M \leftarrow \mathbb{R}^{m \times n}$ to define the operation of choosing a random $m \times n$ matrix M such that each element of it is randomly selected from $\mathbb{R}$. We write $M \in \mathbb{R}^{m \times n}$ if M is an $m \times n$ matrix with each element in $\mathbb{R}$.

III. The Proposal

In a typical FL task, each user has to upload its local gradient to the server(s) for global model updating in each training round. As noticed in Section I, an adversary can make inferences on a user's local data if he can get the local gradient of the user. Thus, the privacy of gradients should be protected. The scheme proposed in this section addresses this issue.

A. High-Level Description

As discussed in Section I-A, DP-based schemes for the protection of gradient privacy in FL cause a loss of model accuracy. Since we aim to achieve *lossless* protection, we do not employ DP in our scheme. Our scheme can be viewed as an MPC-based one and it can be described in terms of three stages: system setup, FL initialization, and training. These stages are graphically represented in Figure 2 and sketched next:

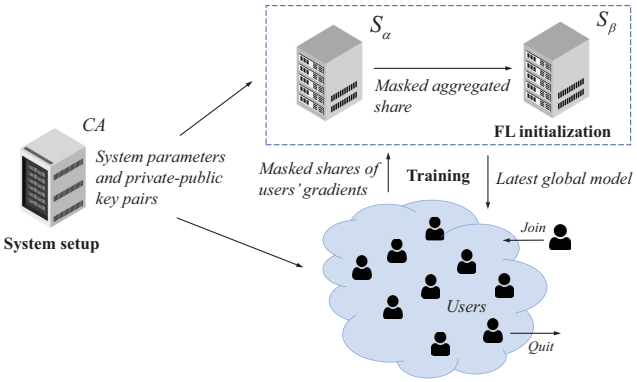


Fig. 2: High-level description of our scheme

- **System Setup**: The CA generates and publishes the system parameters. In addition, the CA also issues and certifies private-public key pairs for the servers and users in the system.
- **FL Initialization**: S_β initializes an FL task which aims to train a global model, i.e., it generates the initial global model and the related parameters. Each user that wants to join the task has to obtain the initial global model and the related parameters, and also establish a shared secret key with each of the servers using a secure key agreement scheme.
- **Training**: The users and servers train a global model in multiple training rounds. Each round consists of three steps, one for user initialization, another for training and masking by the user, and yet another for masked gradient aggregation and de-masking by the server.

In the following sections, we describe the above three stages in detail.

B. System Setup

CA chooses the secure key agreement scheme $\mathrm{KA} = (\mathrm{KA.setup}(\lambda), \mathrm{KA.gen}(params, p_i), \mathrm{KA.agree}(p_i, p_j))$ and generates the system parameters as follows:

1) Execute $\mathrm{KA.setup}(\lambda)$ to generate the parameters $params$.
2) Select a secure symmetric encryption scheme $\mathcal{SE} = (\mathcal{E}_k, \mathcal{D}_k)$.
3) Set $pub = (\mathrm{KA}, \mathcal{SE}, params)$ as the system parameters.

Each entity p_i (a user or a server) in the system has to execute $\mathrm{KA.gen}(params, p_i)$ to obtain its private-public key pair $d_{p_i}^{sk}, d_{p_i}^{pk}$.

C. FL Initialization

A model/gradient in FL can be represented as multiple matrices with different dimensions. Further, a high-dimensional matrix can be expressed as multiple two-dimensional matrices. For simplicity, in our scheme we assume that a model/gradient is represented as a two-dimensional matrix. However, our protocol can be easily extended to the case where multiple matrices are used to represent a model/gradient.

For an FL task, suppose that the global model is represented as an $m \times n$ matrix $\mathcal{M}_{fed}\langle t\rangle$ with each element (usually a floating point number) in $\mathbb{R}$, where t is initially set to be 1. When the FL task is first initialized, server S_β has to select an initial global model $\mathcal{M}_{fed}\langle t\rangle \leftarrow \mathbb{R}^{m\times n}$, secure hash functions $\mathcal{HS} : \{0,1\}^* \rightarrow \{0,1\}^\iota$, $\mathcal{HM} : \{0,1\}^* \rightarrow \mathbb{R}^{m\times l}$, $\mathcal{HM}' : \{0,1\}^* \rightarrow \mathbb{R}^{m\times(n-l)}$ and the related training parameters $tparams$ (e.g., the learning rate, the number of the local epochs and the training batch size), where $\iota \geq \lambda, 1 < l < n$.

Let the local data set of user u_i be $\mathcal{D}_i$ and let $\mathcal{M}_i\langle t\rangle \leftarrow \mathrm{LT}(\mathcal{M}_{fed}\langle t\rangle, \mathcal{D}_i, tparams)$ be the training algorithm whereby u_i obtains its local model $\mathcal{M}_i\langle t\rangle$.

D. Training

This stage consists of multiple rounds. Each round goes through three steps: i) user initialization; ii) user training and masking; and iii) server aggregation and de-masking. Figure 3 depicts a training round.

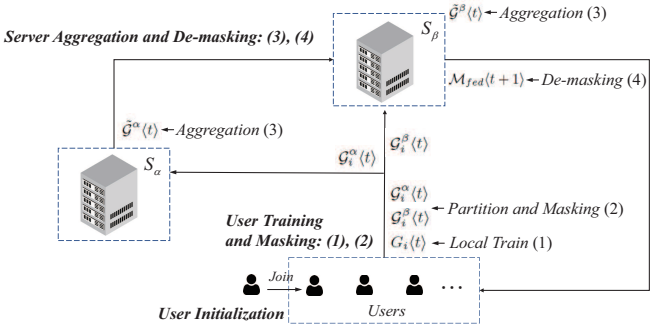


Fig. 3: Steps of a training round: (i) user initialization, (ii) user training and masking (with sub-steps (1) and (2)) and (iii) server aggregation and de-masking (with sub-steps (3) and (4)).

Step 1: User Initialization. This step is only run by users who join an FL task for the first time. A new user u_i needs to establish two shared secret keys with S_α and S_β.

Specifically, u_i and S_α run $\mathrm{KA.agree}(u_i, S_\alpha)$ to generate the key $k_i^\alpha\langle t\rangle$, while u_i and S_β run $\mathrm{KA.agree}(u_i, S_\beta)$ to generate the key $k_i^\beta\langle t\rangle$. These two shared secret keys will be used by u_i to generate its masks. Then S_β sends $\mathcal{E}_{k_i^\beta\langle t\rangle}(\mathcal{M}_{fed}\langle t\rangle||\mathcal{HS}||\mathcal{HM}||\mathcal{HM}'||tparams)$ to u_i, who decrypts this ciphertext to obtain $\mathcal{M}_{fed}\langle t\rangle||\mathcal{HS}||\mathcal{HM}||\mathcal{HM}'||tparams$.

Step 2: User Training and Masking. This step is run by the users to generate masked gradients and the corresponding shares of the masked gradients. Each user u_i runs the following two sub-steps: (1) local training, and (2) partition and masking. Let the current global model received from S_β be $\mathcal{M}_{fed}\langle t\rangle$, let $k_i^\alpha\langle t\rangle$ be the current secret key shared by u_i and S_α, and let $k_i^\beta\langle t\rangle$ be the current secret key shared by u_i and S_β. Then u_i runs the two sub-steps as follows:

1) *Local Train*: u_i runs

$$\mathcal{M}_i\langle t\rangle \leftarrow \mathrm{LT}(\mathcal{M}_{fed}\langle t\rangle, \mathcal{D}_i, tparams)$$

to obtain its local model $\mathcal{M}_i\langle t\rangle$. Then the user computes its local gradient

$$G_i\langle t\rangle \leftarrow \mathcal{M}_i\langle t\rangle - \mathcal{M}_{fed}\langle t\rangle.$$

By our definition, $G_i\langle t\rangle$ is an $m \times n$ matrix.

2) *Partition and Masking*: u_i first partitions its local gradient into two shares and then masks them. Specifically, u_i does:
   a) Run
   $$(G_i^\alpha\langle t\rangle, G_i^\beta\langle t\rangle) \leftarrow P^l(G_i\langle t\rangle)$$
   to generate the shares
   $$G_i^\alpha\langle t\rangle \in \mathbb{R}^{m\times l}$$
   and
   $$G_i^\beta\langle t\rangle \in \mathbb{R}^{m\times(n-l)}.$$
   b) Generate masks
   $$K_i^\alpha\langle t\rangle \in \mathbb{R}^{m\times(n-l)}$$
   and
   $$K_i^\beta\langle t\rangle \in \mathbb{R}^{m\times l},$$
   where $K_i^\alpha\langle t\rangle = \mathcal{HM}'(k_i^\alpha\langle t\rangle)$, $K_i^\beta\langle t\rangle = \mathcal{HM}(k_i^\beta\langle t\rangle)$ can be pre-computed by u_i. Note that the mask $K_i^\alpha\langle t\rangle$, resp. $K_i^\beta\langle t\rangle$, can be also pre-computed by S_α, resp. S_β, since the server knows the shared key $k_i^\alpha\langle t\rangle$, resp. $k_i^\beta\langle t\rangle$.
   c) Generate the masked shares of the gradient as
   $$\mathcal{G}_i^\alpha\langle t\rangle = G_i^\alpha\langle t\rangle + K_i^\beta\langle t\rangle$$
   and
   $$\mathcal{G}_i^\beta\langle t\rangle = G_i^\beta\langle t\rangle + K_i^\alpha\langle t\rangle.$$
   d) Send the masked share $\mathcal{G}_i^\alpha\langle t\rangle$ to S_α, and send $\mathcal{G}_i^\beta\langle t\rangle$ to S_β. The data set size $\xi_i\langle t\rangle$ of u_i is sent to both servers.
   e) Update the shared secret keys by computing
   $$k_i^\alpha\langle t+1\rangle \leftarrow \mathcal{HS}(k_i^\alpha\langle t\rangle)$$
   and
   $$k_i^\beta\langle t+1\rangle \leftarrow \mathcal{HS}(k_i^\beta\langle t\rangle).$$

   Observe that, by our setting, S_α cannot get $G_i^\alpha\langle t\rangle$ by removing $K_i^\beta\langle t\rangle$ since it does not know $K_i^\beta\langle t\rangle$. Similarly, S_β cannot get $G_i^\beta\langle t\rangle$.

Step 3: Server Aggregation and De-masking. This step is performed by the servers and consists of two sub-steps: (3) aggregation, and (4) de-masking. The two servers run as follows:

3) *Aggregation*: In this sub-step, the two servers aggregate the masked shares from the users to obtain their respective masked aggregated shares. Since we assume users may quit an FL task, the two servers have to agree on a user set. Let $\mathcal{Z}\langle t\rangle$ be the set of users that send masked shares and data set sizes to the servers. Let the data set size of user $u_i \in \mathcal{Z}\langle t\rangle$ be $\xi_i\langle t\rangle$. Both servers compute
$$\xi\langle t\rangle = \sum_{u_i\in\mathcal{Z}\langle t\rangle} \xi_i\langle t\rangle.$$
   - If the server is S_α, it first computes
   $$\mathcal{G}^\alpha\langle t\rangle = \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}\mathcal{G}_i^\alpha\langle t\rangle,$$
   $$K^\alpha\langle t\rangle = \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}(-K_i^\alpha\langle t\rangle),$$
   and then generates the masked aggregated share
   $$\tilde{\mathcal{G}}^\alpha\langle t\rangle \leftarrow \mathcal{G}^\alpha\langle t\rangle \sqcup K^\alpha\langle t\rangle.$$
   S_α sends $\tilde{\mathcal{G}}^\alpha\langle t\rangle$ to S_β. Then S_α updates its shared secret key to be $k_i^\alpha\langle t+1\rangle \leftarrow \mathcal{HS}(k_i^\alpha\langle t\rangle)$.
   - Else if the server is S_β, it first computes
   $$\mathcal{G}^\beta\langle t\rangle = \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}\mathcal{G}_i^\beta\langle t\rangle,$$
   $$K^\beta\langle t\rangle = \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}(-K_i^\beta\langle t\rangle),$$
   and then generates the masked aggregated share
   $$\tilde{\mathcal{G}}^\beta\langle t\rangle \leftarrow K^\beta\langle t\rangle \sqcup \mathcal{G}^\beta\langle t\rangle.$$
   S_β updates its shared secret key to be
   $$k_i^\beta\langle t+1\rangle \leftarrow \mathcal{HS}(k_i^\beta\langle t\rangle).$$

4) *De-Masking*: With $\tilde{\mathcal{G}}^\alpha\langle t\rangle$ generated by S_α and $\tilde{\mathcal{G}}^\beta\langle t\rangle$, S_β can obtain an updated global model. It first computes

$G^{\Sigma}\langle t\rangle = \tilde{\mathcal{G}}^{\alpha}\langle t\rangle + \tilde{\mathcal{G}}^{\beta}\langle t\rangle$. It is easy to verify that

$$\begin{aligned}
G^{\Sigma}\langle t\rangle &= \tilde{\mathcal{G}}^{\alpha}\langle t\rangle + \tilde{\mathcal{G}}^{\beta}\langle t\rangle \\
&= \mathcal{G}^{\alpha}\langle t\rangle \sqcup K^{\alpha}\langle t\rangle + K^{\beta}\langle t\rangle \sqcup \mathcal{G}^{\beta}\langle t\rangle \\
&= \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}\mathcal{G}_i^{\alpha}\langle t\rangle \sqcup \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}(-K_i^{\alpha}\langle t\rangle) \\
&+ \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}(-K_i^{\beta}\langle t\rangle) \sqcup \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}\mathcal{G}_i^{\beta}\langle t\rangle \\
&= \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}((G_i^{\alpha}\langle t\rangle + K_i^{\beta}\langle t\rangle) \sqcup (-K_i^{\alpha}\langle t\rangle)) \\
&+ \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}((-K_i^{\beta}\langle t\rangle) \sqcup (G_i^{\beta}\langle t\rangle + K_i^{\alpha}\langle t\rangle) \\
&= \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}(G_i^{\alpha}\langle t\rangle \sqcup G_i^{\beta}\langle t\rangle) \\
&= \sum_{u_i\in\mathcal{Z}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}G_i\langle t\rangle.
\end{aligned}$$

Then S_{β} can compute the updated global model as

$$\mathcal{M}_{fed}\langle t+1\rangle \leftarrow \mathcal{M}_{fed}\langle t\rangle + G^{\Sigma}\langle t\rangle.$$

For $u_i \in \mathcal{Z}\langle t\rangle$, S_{β} encrypts $\mathcal{M}_{fed}\langle t+1\rangle$ under $k_i^{\beta}\langle t+1\rangle$ and sends it to u_i, who can decrypt and obtain $\mathcal{M}_{fed}\langle t+1\rangle$.

The above three steps will be repeated until the accuracy of the global model is high enough or the maximum number of rounds/training time is reached.

Whereas the existing MPC-based schemes for protection of gradient privacy in FL result in large communication and/or computational overheads, it is easy to see that our scheme incurs no message expansion and involves only lightweight operations. Using two servers does not entail a significant overhead increase. See Section V for experimental evidence on performance.

IV. Security Analysis

In this section, we examine the security of our scheme. We first prove an auxiliary theorem, and then we use it in the proof of the main security result.

A. An Auxiliary Theorem

In our scheme, the local gradient of each user is partitioned into two shares which are then masked to generate two masked shares one of which is sent to S_{α} and the other to S_{β}. The following auxiliary theorem proves that if the gradients of the users are partitioned into two matrices (shares) and respective uniformly random masks are added, then the masked gradient shares obtained by an attacker look uniformly random, conditioned on the aggregate result of these masked gradient shares being equal to the aggregate result of the users' gradient shares. In other words, the masks hide the information about the users' individual gradients, except for their aggregation.

Theorem 1: Suppose $\mathcal{U}\langle t\rangle$ is a set that consists of honest users, the data set size of user $u_i \in \mathcal{U}\langle t\rangle$ is $\xi_i\langle t\rangle$, $G_i\langle t\rangle = G_i^{\alpha}\langle t\rangle \sqcup G_i^{\beta}\langle t\rangle$ (as defined in Section III-D), where $G_i^{\alpha}\langle t\rangle \in \mathbb{R}^{m\times l}$, $G_i^{\beta}\langle t\rangle \in \mathbb{R}^{m\times(n-l)}$. Let $\zeta_{|\mathcal{U}\langle t\rangle|}\langle t\rangle = \sum_{i=1}^{|\mathcal{U}\langle t\rangle|}\xi_i\langle t\rangle$; $*, \circ \in \{\alpha,\beta\}$, $* \neq \circ$; $\widetilde{l} \in \{l, n-l\}$; $G_i^{*}\langle t\rangle \in \mathbb{R}^{m\times\widetilde{l}}$ be a matrix generated by a user $u_i \in \mathcal{U}\langle t\rangle$ and sent to S_{*}; $\{K_i^{\circ}\langle t\rangle \in \mathbb{R}^{m\times\widetilde{l}}\}_{1\le i\le|\mathcal{U}\langle t\rangle|}$ be uniformly random masks owned by $S_{\circ}$ and $\{R_i^{*}\langle t\rangle \in \mathbb{R}^{m\times\widetilde{l}}\}_{1\le i\le|\mathcal{U}\langle t\rangle|}$ be uniformly random matrices s.t.

$$\begin{aligned}
&\sum_{u_i\in\mathcal{U}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\zeta_{|\mathcal{U}\langle t\rangle|}\langle t\rangle}(R_i^{*}\langle t\rangle + (-K_i^{\circ}\langle t\rangle)) \\
&= \sum_{u_i\in\mathcal{U}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\zeta_{|\mathcal{U}\langle t\rangle|}\langle t\rangle}G_i^{*}\langle t\rangle.
\end{aligned} \tag{1}$$

For each user $u_i \in \mathcal{U}\langle t\rangle$, we have

$$G_i^{*}\langle t\rangle + K_i^{\circ}\langle t\rangle \equiv R_i^{*}\langle t\rangle, \tag{2}$$

which denotes that the distributions of $G_i^{*}\langle t\rangle + K_i^{\circ}\langle t\rangle$ and $R_i^{*}\langle t\rangle$ are identical.

Proof. Define $LHS_i = G_i^{*}\langle t\rangle + K_i^{\circ}\langle t\rangle$, $RHS_i = R_i^{*}\langle t\rangle$ for $i \in \{1,..,|\mathcal{U}\langle t\rangle|\}$, $LS_{|\mathcal{U}\langle t\rangle|} = \sum_{u_i\in\mathcal{U}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\zeta_{|\mathcal{U}\langle t\rangle|}\langle t\rangle}(R_i^{*}\langle t\rangle + (-K_i^{\circ}\langle t\rangle))$, $RS_{|\mathcal{U}\langle t\rangle|} = \sum_{u_i\in\mathcal{U}\langle t\rangle} \frac{\xi_i\langle t\rangle}{\zeta_{|\mathcal{U}\langle t\rangle|}\langle t\rangle}G_i^{*}\langle t\rangle$. We prove the theorem by induction.

Base Case: $|\mathcal{U}\langle t\rangle| = 1$. According to Equation (1), we have $R_1^{*}\langle t\rangle = G_1^{*}\langle t\rangle + K_1^{\circ}\langle t\rangle$, i.e., $LHS_1 = RHS_1$. Obviously, the distribution of LHS_1 is identical to the distribution of RHS_1. Thus, we have $LHS_1 \equiv RHS_1$.

Induction Step: Assume the theorem holds when $|\mathcal{U}\langle t\rangle| = \omega$. In other words, we have $LHS_i \equiv RHS_i$ for $i \in \{1,..,\omega\}$ when $LS_{\omega} = RS_{\omega}$.

Define

$$\begin{cases}
\mathcal{U}'\langle t\rangle = \mathcal{U}\langle t\rangle; \\
\mathcal{U}\langle t\rangle = \mathcal{U}'\langle t\rangle \cup \{u_{\omega+1}\}; \\
RN_{|\mathcal{U}'\langle t\rangle|} = \sum_{u_i\in\mathcal{U}'\langle t\rangle} \frac{\xi_i\langle t\rangle}{\zeta_{|\mathcal{U}\langle t\rangle|}\langle t\rangle}G_i^{*}\langle t\rangle; \\
LN_{|\mathcal{U}'\langle t\rangle|} = \sum_{u_i\in\mathcal{U}'\langle t\rangle} \frac{\xi_i\langle t\rangle}{\zeta_{|\mathcal{U}\langle t\rangle|}\langle t\rangle}(R_i^{*}\langle t\rangle + (-K_i^{\circ}\langle t\rangle)); \\
\delta_1 = RN_{\omega} - RS_{\omega}; \\
\delta_2 = LN_{\omega} - LS_{\omega}.
\end{cases}$$

According to Equation (1), we require $LS_{\omega+1} = RS_{\omega+1}$, i.e., $LN_{\omega} + \frac{\xi_{\omega+1}\langle t\rangle}{\zeta_{\omega+1}\langle t\rangle}(R_{\omega+1}^{*}\langle t\rangle + (-K_{\omega+1}^{\circ}\langle t\rangle)) = RN_{\omega} + \frac{\xi_{\omega+1}\langle t\rangle}{\zeta_{\omega+1}\langle t\rangle}G_i^{*}\langle t\rangle$. Since $LN_{\omega} = \delta_2 + LS_{\omega}$, $RN_{\omega} = \delta_2 + RS_{\omega}$, we have

$$\begin{aligned}
&\delta_2 + LS_{\omega} + \frac{\xi_{\omega+1}\langle t\rangle}{\zeta_{\omega+1}\langle t\rangle}(R_{\omega+1}^{*}\langle t\rangle + (-K_{\omega+1}^{\circ}\langle t\rangle)) \\
&= \delta_1 + RS_{\omega} + \frac{\xi_{\omega+1}\langle t\rangle}{\zeta_{\omega+1}\langle t\rangle}G_{\omega+1}^{*}\langle t\rangle.
\end{aligned}$$

Since we assume $LS_{\omega} = RS_{\omega}$, we have

$$\delta_2 + \frac{\xi_{\omega+1}\langle t\rangle}{\zeta_{\omega+1}\langle t\rangle}(R_{\omega+1}^{*}\langle t\rangle + (-K_{\omega+1}^{\circ}\langle t\rangle)) = \delta_1 + \frac{\xi_{\omega+1}\langle t\rangle}{\zeta_{\omega+1}\langle t\rangle}G_{\omega+1}^{*}\langle t\rangle.$$

Let $\delta = \frac{\zeta_{\omega+1}\langle t\rangle}{\xi_{\omega+1}\langle t\rangle}(\delta_2 - \delta_1)$. We have

$$R_{\omega+1}^{*}\langle t\rangle + \delta = G_{\omega+1}^{*}\langle t\rangle + K_{\omega+1}^{\circ}\langle t\rangle.$$

Thus, the distribution of $R^*_{\omega+1}\langle t\rangle + \delta$ is identical to the distribution of $G^*_{\omega+1}\langle t\rangle + K^\circ_{\omega+1}\langle t\rangle$. Since δ is a fixed matrix and $R^*_{\omega+1}\langle t\rangle$ is a uniformly random matrix, the distributions of $R^*_{\omega+1}\langle t\rangle$ and $R^*_{\omega+1}\langle t\rangle + \delta$ are the same. Therefore, we have

$$G^*_{\omega+1}\langle t\rangle + K^\circ_{\omega+1}\langle t\rangle \equiv R^*_{\omega+1}\langle t\rangle,$$

and $LHS_i \equiv RHS_i$ for $i \in \{1, .., \omega+1\}$.

Conclusion: By induction, the theorem follows.

B. Security Proof

In this section, we prove the security of our scheme via the simulation-based approach, also called the real-ideal world paradigm [5], [9], [17]. For a scheme in the real world, where an attacker can corrupt a participant, the attacker's view contains the input and output of the corrupted participant and all the messages received from other participants. In the ideal world, although an attacker (simulator) can also corrupt a participant, its view only contains the input and output of the corrupted participant since ideally the attacker cannot obtain any other information. If the attacker achieves the same attack effect in the real world as it achieves in the ideal world, then we say the execution of our scheme in the real world is indistinguishable from the execution in the ideal world. In other words, the scheme in the real world is secure.

Theorem 2: Assuming that at most one server and all users participating FL but two can be corrupted, the joint view of corrupted participants in the real world is indistinguishable from the joint view of participants in the ideal world (limited to their own inputs and outputs). Hence, our scheme is secure.

Proof: Let $\mathcal{Z}$ be the set of potential users and $\{S_\alpha, S_\beta\}$ be the set of servers. Since we assume the two servers will not collude, an adversary can corrupt at most one of them. Let $\mathcal{P}\langle t\rangle \subseteq \mathcal{Z}$ be the subset of users participating in the t-th round and let $\mathcal{C}\langle t\rangle$ be the subset of users who have been corrupted until the t-th round of the FL task. Here if a user is corrupted in the t-th round of an FL task, then it is also corrupted in the following rounds. Further, for the t-th round of an FL task, we require that at least two of the users in $\mathcal{P}\langle t\rangle$ should be honest, i.e, $|\mathcal{P}\langle t\rangle \setminus \mathcal{C}\langle t\rangle| \geq 2$.

We construct a simulator that makes a series of successive modifications (from Hyb_0 to Hyb_6) to our scheme in the real world such that the joint views of participants in any two successive modified schemes are computationally indistinguishable.

Hyb_0: The execution of this hybrid scheme is exactly the same as the real execution of our scheme, which means that the joint view of corrupted participants in Hyb_0 is identical to the joint view of corrupted participants in the real world.

Hyb_1: In our scheme, we assume that server S_α and server S_β are not colluding. In other words, an adversary cannot corrupt both of them. We first consider that the server S_α is corrupted by an adversary, while S_β is not. Let the honest entities form the set $\{S_\beta\} \cup \mathcal{P}\langle t\rangle \setminus \mathcal{C}\langle t\rangle$. This hybrid is the same as Hyb_0, except that the simulator will not run the $\mathrm{KA.agree}(u_i, S_\beta)$ algorithm to generate the secret $k_i^\beta\langle t\rangle$ $(t = 1)$ for a honest user $u_i \in \mathcal{P}\langle t\rangle \setminus \mathcal{C}\langle t\rangle$ and the server S_β in the user initialization step. Instead, the simulator chooses a uniformly random value $r_i^\beta\langle t\rangle$ as the secret. Since the key agreement scheme chosen is provably secure, the shared key generated by this scheme is indistinguishable from a uniformly random value. In other words, the distributions of $k_i^\beta\langle t\rangle$ and the uniformly random value $r_i^\beta\langle t\rangle$ are identical. The security of the selected key agreement scheme guarantees that this hybrid is indistinguishable from the previous one.

If S_β is corrupted by an adversary but S_α is not, instead of running the $\mathrm{KA.agree}(u_i, S_\alpha)$ algorithm with S_α in the user initialization step to generate the secret $k_i^\alpha\langle t\rangle$, the simulator chooses a uniformly random value $r_i^\alpha\langle t\rangle$ as the secret. Similar to the above case, the security of the chosen key agreement scheme guarantees that this hybrid is indistinguishable from the previous one.

Hyb_2: We first consider the case in which S_α is corrupted while S_β is not. This hybrid is the same as Hyb_1, except that, for the user initialization step, the simulator will not send the ciphertext $\mathcal{E}_{r_i^\beta\langle t\rangle}(\mathcal{M}_{fed}\langle t\rangle||\mathcal{HS}||\mathcal{HM}||\mathcal{HM}'||tparams)$ $(t = 1)$ to each user. Instead, it will choose a uniformly random value $R^r_{\mathcal{E}}\langle 1\rangle$ as the ciphertext. Since the ciphertext obtained by a secure symmetric encryption scheme is uniformly random, $\mathcal{E}_{r_i^\beta\langle t\rangle}(\mathcal{M}_{fed}\langle t\rangle||\mathcal{HS}||\mathcal{HM}||\mathcal{HM}'||tparams)$ and $R^r_{\mathcal{E}}\langle 1\rangle$ have identical distributions. Therefore, this hybrid is indistinguishable from the previous one.

In case S_β is corrupted by an adversary but S_α is not, the simulator will select a random value $R^k_{\mathcal{E}}\langle 1\rangle$ as the ciphertext, instead of sending the ciphertext $\mathcal{E}_{k_i^\beta\langle t\rangle}(\mathcal{M}_{fed}\langle t\rangle||\mathcal{HS}||\mathcal{HM}||\mathcal{HM}'||tparams)$ $(t = 1)$ to each user. Similar to the above case, the security of the chosen symmetric encryption scheme guarantees that this hybrid is indistinguishable from the previous one.

Hyb_3: We first analyze the case in which S_α is corrupted but S_β is not. Note that, in Hyb_1, we only replace $k_i^\beta\langle 1\rangle$ (i.e., $t = 1$) with a random value $r_i^\alpha\langle 1\rangle$ in the user initialization step. In this hybrid, when $t \geq 2$, the simulator will not generate the secret $k_i^\beta\langle t\rangle$ shared between $u_i \in \mathcal{P}\langle t\rangle \setminus \mathcal{C}\langle t\rangle$ and S_β in the $(t-1)$-th round of the training stage by using the hash function $\mathcal{HS}$. Instead, the simulator chooses a uniformly random value $r_i^\beta\langle t\rangle$ and sets $k_i^\beta\langle t\rangle = r_i^\beta\langle t\rangle$. Since the output of a secure hash function is uniformly random, $k_i^\beta\langle t\rangle$ is indistinguishable from a uniformly random value. Thus, the distributions of $k_i^\beta\langle t\rangle$ and the uniformly random value $r_i^\beta\langle t\rangle$ are the same. Therefore, this hybrid is indistinguishable from the previous hybrid.

In case S_β is corrupted by an adversary but S_α is not, the simulator behaves analogously as described in the previous paragraph. It chooses a uniformly random value $r_i^\alpha\langle t\rangle$ as the secret held by $u_i \in \mathcal{P}\langle t\rangle \setminus \mathcal{C}\langle t\rangle$ and S_β in the $(t-1)$-th round of the training stage. By an argument similar to the one in the previous paragraph, the security of the hash function $\mathcal{HS}$ being used guarantees that this hybrid is indistinguishable from the previous hybrid.

Hyb_4: We first analyze the case in which S_α is corrupted while S_β is not. This hybrid is the same as Hyb_3, except that, for the user initialization step, the simulator will not send the ciphertext $\mathcal{E}_{r_i^\beta\langle t\rangle}(\mathcal{M}_{fed}\langle t\rangle||\mathcal{HS}||\mathcal{HM}||\mathcal{HM}'||tparams)$ $(t \geq 2)$ to each user. Instead, it will choose a uniformly random value $R^r_{\mathcal{E}}\langle t\rangle$ as the ciphertext. Since the ciphertext obtained by

a secure symmetric encryption scheme is uniformly random, $R^r_{\mathcal{E}}\langle t\rangle$ and $\mathcal{E}_{r^\beta_i\langle t\rangle}(\mathcal{M}_{fed}\langle t\rangle||\mathcal{HS}||\mathcal{HM}||\mathcal{HM}'||tparams)$ have identical distributions. Therefore, this hybrid is indistinguishable from the previous one.

In case S_β is corrupted by an adversary and S_α is not, the simulator will send a random value $R^k_{\mathcal{E}}\langle t\rangle$ instead of sending the ciphertext $\mathcal{E}_{k^\beta_i\langle t\rangle}(\mathcal{M}_{fed}\langle t\rangle||\mathcal{HS}||\mathcal{HM}||\mathcal{HM}'||tparams)$ ($t \geq 2$) to each user. Similarly as argued above, the security of the chosen symmetric encryption scheme guarantees that this hybrid is indistinguishable from the previous one.

Hyb_5: We first consider that S_α is corrupted while S_β is not. This hybrid is the same as Hyb_4, except that, for the t-th round of training stage, the simulator will not generate the mask $K^\beta_i\langle t\rangle$ shared between S_β and $u_i \in \mathcal{P}\langle t\rangle \setminus \mathcal{C}\langle t\rangle$ in the user training and masking step by using hash function $\mathcal{HM}$. Instead, the simulator selects a uniformly random matrix $\gamma^\beta_i\langle t\rangle$ and sets $K^\beta_i\langle t\rangle = \gamma^\beta_i\langle t\rangle$. Since the output of a secure hash function is uniformly random, the mask $K^\beta_i\langle t\rangle$ is indistinguishable from a uniformly random matrix. Hence, the distributions of $K^\beta_i\langle t\rangle$ and $\gamma^\beta_i\langle t\rangle$ are identical. Therefore, this hybrid is indistinguishable from the previous one.

The case in which S_β is corrupted but S_α is not is similar to the above one. Instead of using the hash function $\mathcal{HM}'$ to obtain the mask $K^\alpha_i\langle t\rangle$, the simulator selects a uniformly random matrix $\gamma^\alpha_i\langle t\rangle$ as the mask. Using analogous arguments as above, this hybrid is indistinguishable from the previous one.

Hyb_6: We first consider that S_α is corrupted by an adversary whereas S_β is not. This hybrid is the same as Hyb_5, except that the simulator will not send $\{G^\alpha_i\langle t\rangle + K^\beta_i\langle t\rangle\}_{u_i\in\mathcal{P}\langle t\rangle\setminus\mathcal{C}\langle t\rangle}$ to S_α. Instead, the simulator chooses uniformly random matrices $\{R^\alpha_i\langle t\rangle\}_{u_i\in\mathcal{P}\langle t\rangle\setminus\mathcal{C}\langle t\rangle}$ that satisfy

$$\sum_{u_i\in\mathcal{P}\langle t\rangle\setminus\mathcal{C}\langle t\rangle}\frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}(R^\alpha_i\langle t\rangle - K^\beta_i\langle t\rangle) = \sum_{u_i\in\mathcal{P}\langle t\rangle\setminus\mathcal{C}\langle t\rangle}\frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}G^\alpha_i\langle t\rangle,$$

and sends them to S_α. We have to prove that the distribution of $R^\alpha_i\langle t\rangle$ is indistinguishable from that of $G^\alpha_i\langle t\rangle + K^\beta_i\langle t\rangle$ in Hyb_3.

Let $\mathcal{U}\langle t\rangle \leftarrow \mathcal{P}\langle t\rangle \setminus \mathcal{C}\langle t\rangle$, $* \leftarrow \alpha$, $\circ \leftarrow \beta$. According to Theorem 1, we have

$$G^\alpha_i\langle t\rangle + K^\beta_i\langle t\rangle \equiv R^\alpha_i\langle t\rangle,$$

which means that $R^\alpha_i\langle t\rangle$'s distribution is identical to the distribution of $G^\alpha_i\langle t\rangle + K^\beta_i\langle t\rangle$ in Hyb_5. Therefore, this hybrid is indistinguishable from the previous one.

If the server S_β is corrupted but S_α is not, instead of sending $\{G^\beta_i\langle t\rangle + K^\alpha_i\langle t\rangle\}_{u_i\in\mathcal{P}\langle t\rangle\setminus\mathcal{C}\langle t\rangle}$ to S_β, the simulator selects uniformly random matrices $\{R^\beta_i\langle t\rangle\}_{u_i\in\mathcal{P}\langle t\rangle\setminus\mathcal{C}\langle t\rangle}$ that satisfy

$$\sum_{u_i\in\mathcal{P}\langle t\rangle\setminus\mathcal{C}\langle t\rangle}\frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}(R^\beta_i\langle t\rangle - K^\alpha_i\langle t\rangle) = \sum_{u_i\in\mathcal{P}\langle t\rangle\setminus\mathcal{C}\langle t\rangle}\frac{\xi_i\langle t\rangle}{\xi\langle t\rangle}G^\beta_i\langle t\rangle,$$

and sends them to S_β.

Let $\mathcal{U}\langle t\rangle \leftarrow \mathcal{P}\langle t\rangle \setminus \mathcal{C}\langle t\rangle$, $* \leftarrow \beta$, $\circ \leftarrow \alpha$. According to Theorem 1, we have

$$G^\beta_i\langle t\rangle + K^\alpha_i\langle t\rangle \equiv R^\beta_i\langle t\rangle,$$

Hence, this hybrid is indistinguishable from the previous one.

Since the simulator completes the simulation in Hyb_6 by invoking only the input and output of the corrupted participants, the execution of our scheme in Hyb_6 is the same as in the ideal world. Thus, we have completed the construction of the simulator and performed the modification of our scheme from the real world to the ideal world. Therefore, we can say that the joint view of corrupted participants in the real world is indistinguishable from the joint view of corrupted participants in the ideal world. This completes the security proof.

V. Performance Evaluation

In this section, we evaluate the performance of our scheme through comparison and experiments. For the experiments, we employed PyTorch as our platform. We used a PC with an Intel(R) Xeon(R) Silver 4114 CPU @2.2GHz, 64G RAM and an Titan-V graphics card. For the sake of reproducibility, the code of our experiments is available at https://github.com/Don-Chung/Lightweight-Privacy-Preserving-Federated-Learning.

A. Comparison of Properties

In this section, we compare our scheme with several state-of-the-art privacy-preserving FL schemes with regard to the properties listed in Section II-B. Since our focus in on the semi-trusted model, we limit our comparison to schemes in that model (see Table I).

The scheme in [1] is DP-based. Although it is lightweight, round-efficient and dynamic, it entails loss of model accuracy. Further, no formal security proof is provided for this scheme. The schemes in [4], [25] and ours are lossless in terms of accuracy, because they are all MPC-based. Yet, [4], [25] are not lightweight, because the former relies on a costly secret sharing technique, whereas the latter uses additive homomorphic encryption to guarantee gradient privacy. Thus, both of these schemes incur high computation and communication overheads. Further, [25] does not include a security proof and [4] is neither fully dynamic nor round-efficient. In conclusion, only our scheme achieves all the properties defined in Section II-B. Note that, although our scheme is MPC-based, it remains lightweight because it only relies on hash functions and symmetric cryptography to protect the privacy of a user's local gradient.

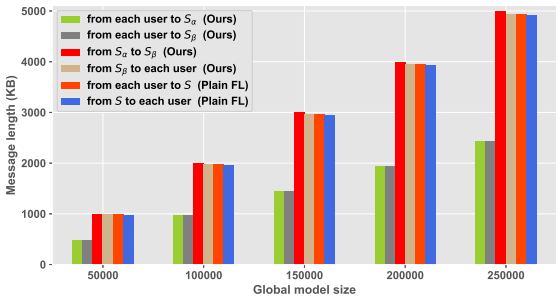


Fig. 4: Comparison of communication costs

TABLE I: Comparison of properties

Schemes	Scheme in [1]	Scheme in [4]	Scheme in [25]	Our scheme
Architecture	Single-server	Single-server	Single-server	Dual-server
Privacy-preserving approach	DP-based	MPC-based	MPC-based	MPC-based
Lossless accuracy	✗	✓	✓	✓
Lightweight computation	✓	✗	✗	✓
Dynamic user behavior	Join + Quit	Quit	Join + Quit	Join + Quit
Round-efficient	✓	✗	✓	✓
Security proof	-	Simulation-based	-	Simulation-based

B. Comparison of Computation and Communication Costs

In this section, we experimentally evaluate the computation and communication costs of our scheme. We take as baseline for comparison the plain FL scheme (trusted scenario, no privacy-preserving countermeasures needed), because it is the most efficient. Since a global model (and hence the local gradient of a user) consists of multiple matrices, in what follows we use the total number of elements in these matrices as the model size. Note that each element is a floating point number. In our experiments, the global model size ranged from 50,000 to 250,000.

Figure 4 shows the communication costs of our scheme and the plain FL, where S denotes the server in the plain FL. In our scheme, in each training round, each user has to generate two masked shares of a local gradient which are sent to S_α and S_β, respectively. However, we note that the total length of the two masked shares is the same as that of the corresponding local gradient. Further, S_α has to send a masked aggregated share to S_β and S_β has to send an encrypted global model to the users. Compared with the plain FL, the cost of sending a masked aggregated share from S_α to S_β represents a communication overhead. It is easy to see in Figure 4 that the total communication overhead of our scheme (first four bars for each model size) is slightly higher than that of the plain FL (last two bars for each model size).

Regarding the computational cost, we assume that the cost of each user to generate a local gradient in our scheme is the same as that in the plain FL. Therefore, we will not show that cost here. We concentrate on the additional computation overhead of our scheme due to the adoption of cryptographic primitives. We note that the shared secret keys can be pre-updated by the servers/users in our scheme and that our key update mechanism involves only lightweight operations. Hence the cost of updating the shared secret keys is not considered in our experiments, where the number of users was set to be 10, 40, 70 and 100.

Figure 5 shows the relationship between the global model size and the average computation overhead of a user/server in each round, where S(plain FL) denotes the computation overhead of a server in the plain FL. We do not represent the computation cost for a user in the plain FL, because the user's only cost is to generate its local gradient, which is not counted, as justified above. As shown in Figure 5, as the global model size grows, the computation cost grows linearly. Also, as the number of users grows, the computation cost at the servers also grows. In contrast, as it could be expected, the number of users does not influence the cost for an individual user. The important feature is that the computation cost for the servers in our scheme is only slightly higher than the computation cost of a server in the plain FL.

C. Model Accuracy

In this section, we evaluate the model accuracy of our scheme. In our experiments, we used the FMNIST[1] and CIFAR10[2] data sets, from which we randomly drew the local data sets of users. We trained a 4-layer fully-connected neural network model on the FMNIST data set [34], and a VGG-16 neural network model (composed of 13 convolutional layers and 3 fully-connected layers) on the CIFAR10 data set [16]. For the experiments on FMNIST, we set the training batch size to 128, the learning rate to 0.01, the number of local epochs to 5, and we adopted the SGD optimizer. For the experiments on CIFAR10, we set the training batch size to 128, the learning rate to 0.001, the number of local epochs to 5 and we adopted the Adam optimizer.

We first compare the model accuracy of our scheme in the *static* case with that of the plain FL scheme. In the experiments, the number of training rounds ranged from 1 to 50, and the number of users was set to 10, 40, 70 and 100. Figure 6 shows the results. It can be seen that the accuracy of our scheme is almost identical to that of the plain FL. This also implies that our scheme is lossless.

Next, we evaluated the impact of accommodating a dynamic user behavior on the model accuracy. In our experiments, the number of training rounds ranged from 1 to 50, and the average number of users in a training round was set to 10, 40, 70 and 100. In each training round, we allowed up to 10% of the users to join or quit the FL task.

The results of our experiments are reported in Figure 7. It turns out that the model accuracy of our scheme in the dynamic case is very similar to the model accuracy in the static case. Thus the impact of users joining or quitting on the accuracy is very limited.

VI. Conclusions

In this paper, we have proposed a privacy-preserving FL scheme using a dual-server FL architecture. Thanks to this architecture, our scheme only needs to employ lightweight cryptographic techniques to protect the privacy of a user's

[1]https://www.kaggle.com/code/digvijayyadav/fmnist-nn-and-cnn/notebook
[2]https://www.kaggle.com/c/cifar-10

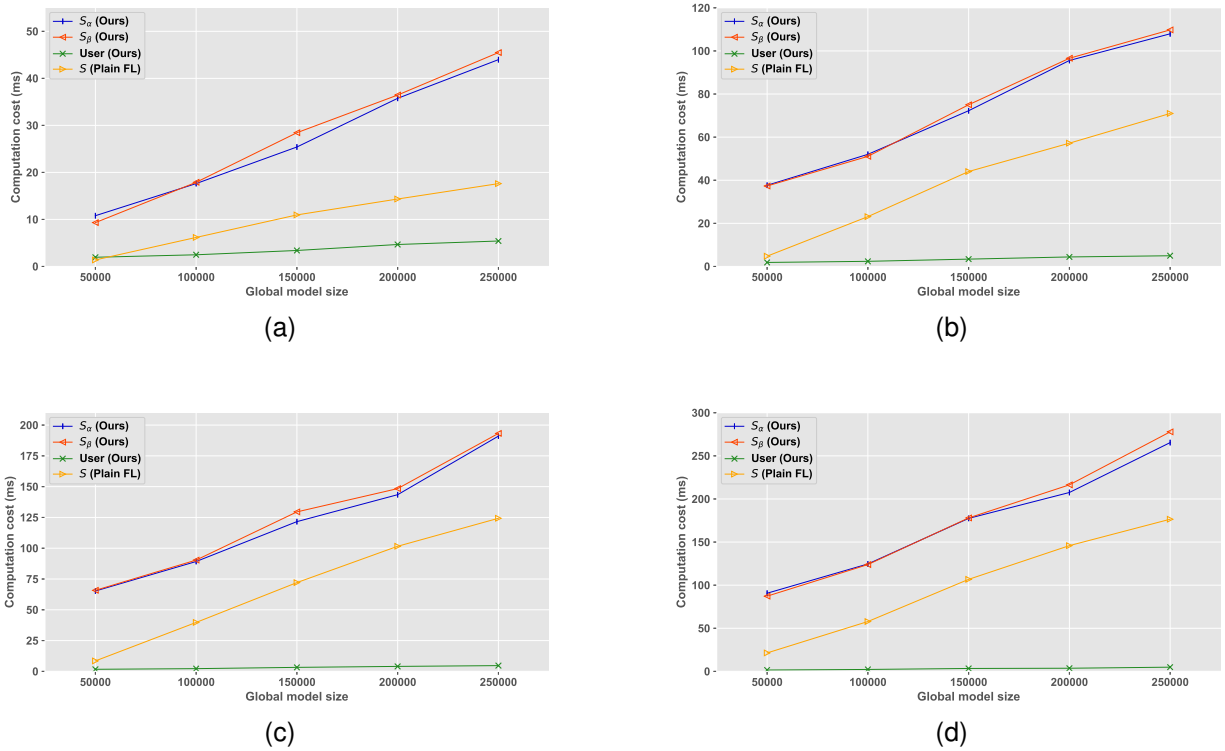


Fig. 5: Computation overheads of the users and servers in the plain FL and in our scheme. (a) 10 users. (b) 40 users. (c) 70 users. (d) 100 users.

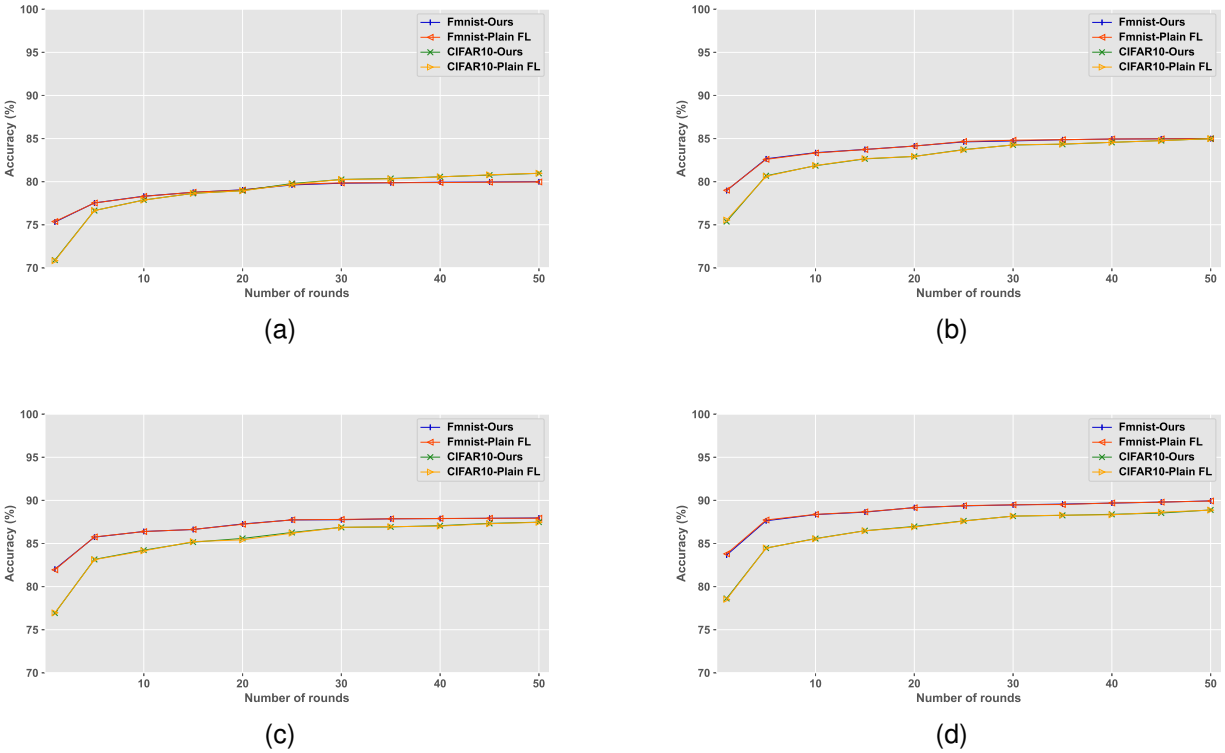


Fig. 6: Global model accuracy of the plain FL and our scheme. (a) 10 users. (b) 40 users. (c) 70 users. (d) 100 users.

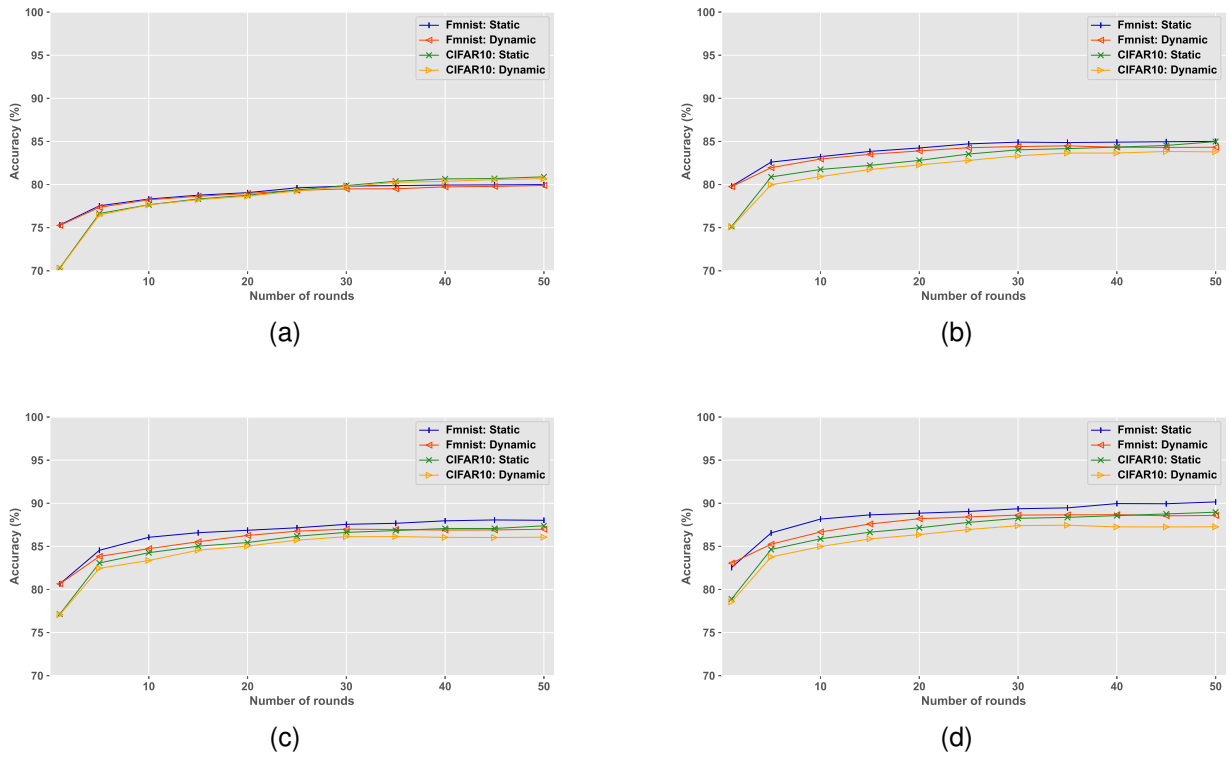


Fig. 7: Impact of dynamic on global model accuracy in our scheme. (a) 10 users on average. (b) 40 users on average. (c) 70 users on average. (d) 100 users on average.

local gradient and it achieves a low communication overhead. Further, in our scheme, each user can decide to join/quit an FL task at any time and no noise is added to the user's gradients. Regarding privacy, we have given a formal proof that our scheme preserves gradient privacy even under collusion attacks.

Acknowledgments

Partial support from the European Commission (projects H2020-871042 "SoBigData++" and H2020-101006879 "MobiDataLab") and the Government of Catalonia (ICREA Acadèmia Prize to J. Domingo-Ferrer) is gratefully acknowledged.

References

[1] M. Abadi, A. Chu, I. Goodfellow, HB. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

[2] N. Agarwal, AT. Suresh, F. Yu, S. Kumar, and HB. Mcmahan, "cpSGD: Communication-efficient and differentially-private distributed SGD," in *arXiv preprint arXiv:1805.10559*, 2018.

[3] M. Bellare, and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 62–73, 1993.

[4] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, HB. 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*, pp. 1175–1191, 2017.

[5] D. Butler, D. Aspinall, and A. Gascón, "How to simulate it in isabelle: Towards formal proof for secure multi-party computation," in *International Conference on Interactive Theorem Proving*, pp. 114–130, 2017.

[6] A. Castiglione, A. De Santis, and B. Masucci, "Key indistinguishability versus strong key indistinguishability for hierarchical key assignment schemes," in *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 4, pp. 451–460, 2015.

[7] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "When machine unlearning jeopardizes privacy," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 896–911, 2021.

[8] A. Dubey, and A. Pentland, "Differentially-private federated linear bandits," in *arXiv preprint arXiv:2010.11425*, 2020.

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

[10] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," in *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[11] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," in *arXiv preprint arXiv:1711.10677*, 2017.

[12] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong, "Personalized federated learning with differential privacy," in *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9530–9539, 2020.

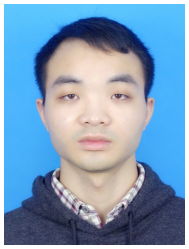
[13] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits," in *USENIX Security Symposium*, vol. 201, pp. 331–335, 2011.

[14] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *USENIX Security Symposium*, pp. 1895–1912, 2019.

[15] M. Keller, V. Pastro, and D. Rotaru, "Overdrive: Making SPDZ great again," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 158–189, 2018.

[16] A. Krizhevsky, and G. Hinton, "Learning multiple layers of features from tiny images," in *Citeseer*, 2009.

[17] Y. Lindell, "How to simulate it–a tutorial on the simulation proof technique," in *Tutorials on the Foundations of Cryptography*, pp. 277–346, 2017.
[18] R. Liu, Y. Cao, H. Chen, R. Guo, and M. Yoshikawa, "FLAME: Differentially Private Federated Learning in the Shuffle Model," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, pp. 8688–8696, 2021.
[19] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," in *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4574–4588, 2021.
[20] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera-Arcas, "Communication-efficient learning of deep networks from decentralized data," in *20th Intl. Conf. on Artificial Intelligence and Statistics (AISTATS 2017)*, pp. 1273–1282, 2017.
[21] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *IEEE Symposium on Security and Privacy*, pp. 691–706, 2019.
[22] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *IEEE Symposium on Security and Privacy*, pp. 19–38, 2017.
[23] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *IEEE Symposium on Security and Privacy*, pp. 334–348, 2013.
[24] M. S. Ozdayi, M. Kantarcioglu, and Y. R. Gel, "Defending against Backdoors in Federated Learning with Robust Learning Rate," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.
[25] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," in *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.
[26] MS. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 707–721, 2018.
[27] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "Deepsecure: Scalable provably-secure deep learning," in *Proceedings of the 55th Annual Design Automation Conference*, pp. 1–6, 2018.
[28] V. Smith, C. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *arXiv preprint arXiv:1705.10467*, 2017.
[29] Z. Sun, R. Sun, and L. Lu, "Mind your weight (s): A large-scale study on insufficient machine learning model protection in mobile apps," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1955–1972, 2021.
[30] T. Veugen, R. de Haan, R. Cramer, and F. Muller, "A framework for secure computations with two non-colluding servers and multiple clients, applied to recommendations," in *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 445–457, 2014.
[31] Q. Wang, M. Du, X. Chen, Y. Chen, P. Zhou, X. Chen, and X. Huang, "Privacy-preserving collaborative model learning: The case of word vector training," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2381–2393, 2018.
[32] S. Warnat-Herresthal, H. Schultze, K. L. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, and N. A. Aziz, "Swarm Learning for decentralized and confidential clinical machine learning," in *Nature*, vol. 594, no. 7862, pp. 265–270, 2021.
[33] K. Wei, M. Ding, F. Farokhi, and T. Q. S. Quek, "Federated learning with differential privacy: Algorithms and performance analysis," in *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.
[34] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image data set for benchmarking machine learning algorithms," in *arXiv preprint arXiv:1708.07747*, 2017.
[35] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," in *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.
[36] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *IEEE Symposium on Security and Privacy*, pp. 332–349, 2019.
[37] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *USENIX Annual Technical Conference*, pp. 493–506, 2020.
[38] W. Zheng, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Helen: Maliciously secure coopetitive learning for linear models," in *IEEE Symposium on Security and Privacy*, pp. 724–738, 2019.
[39] L. Zhu, Z. Liu and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.



Liangyu Zhong received the B.S. degree in electrical engineering and automation from University of Shanghai for Science and Technology, Shanghai, China. He is currently pursuing the master's degree with the School of Software Engineering, East China Normal University, Shanghai, China. His current research interests include information security, multi-party computation and privacy-preserving machine learning.

Lulu Wang received the B.S. degree (awarded outstanding graduates) in software engineering from Jiangsu University of Science and Technology, Suzhou, China. He is currently pursuing his Ph.D. degree with the School of Software Engineering, East China Normal University, Shanghai, China. His current research interests include information security, multi-party computation and privacy-preserving machine learning.

Lei Zhang (Member, IEEE) received the Ph.D. degree in computer engineering from Universitat Rovira i Virgili, Tarragona, Spain. Since then, he has been with Universitat Rovira i Virgili, as a Post-doctoral Researcher. He is currently a Full Professor with the School of Software Engineering, East China Normal University, Shanghai, China. He has been a holder/coholder of more than ten China/Spain-funded (key) projects. His fields of activity are information security, VANET security, cloud security, data privacy, and network security. He has authored over 90 publications. He has served in the program committee of more than 80 international conferences in information security and privacy.

Josep Domingo-Ferrer (Fellow, IEEE) is a Distinguished Professor of Computer Science and an ICREA-Academia Researcher at Universitat Rovira i Virgili, Tarragona, Catalonia, where he holds the UNESCO Chair in Data Privacy and leads CYBERCAT (Center for Cybersecurity Research of Catalonia). He received the MSc and PhD degrees in Computer Science from the Autonomous University of Barcelona in 1988 and 1991, respectively. He also holds an MSc degree in Mathematics. His research interests are in data privacy, data security and cryptographic protocols. More information on him can be found at http://crises-deim.urv.cat/jdomingo

Burong Kang received the B.S. degree in computer science from Northwest Normal University, Lanzhou, China. She is currently working toward the PhD degree with the School of Software Engineering, East China Normal University, Shanghai, China. Her research interests include information security, public key cryptography, and network security.

Rui Zhang received the B.S. degree in software engineering from East China Normal University, Shanghai, China. She is currently pursuing her Ph.D. degree with the School of Software Engineering, East China Normal University, Shanghai, China. Her current research interests include information security, cloud security, and VANET security.