IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 19, 2021, accepted October 26, 2021, date of publication October 28, 2021, date of current version November 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3124020

Enhancing Differential Privacy for Federated Learning at Scale

CHUNGHUN BAEK[1], SUNGWOOK KIM[2], DONGKYUN NAM[1], AND JIHOON PARK[1]
[1]Samsung Research, Seoul 06765, South Korea
[2]Department of Information Security, Seoul Women's University, Seoul 01797, South Korea

Corresponding author: Sungwook Kim (kim.sungwook@swu.ac.kr)

The work of Sungwook Kim was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) Grant funded by the Korean Government (MSIT) under Grant 20210007270012002 (A Study on Cryptographic Primitives for SNARK, 90%); and in part by the Research Grant funded by Seoul Women's University under Grant 2020-0454 (Research on Privacy-Preserving Data Analysis, 10%).

ABSTRACT Federated learning (FL) is an emerging technique that trains machine learning models across multiple de-centralized systems. It enables local devices to collaboratively learn a model by aggregating locally computed updates via a server. Privacy is a core aspect of FL, and recent works in this area are advancing the privacy guarantee of an FL network. To ensure rigorous privacy guarantee for FL, prior works have focused on methods to securely aggregate local updates and provide differential privacy (DP). In this paper, we investigate a new privacy risk for FL. Specifically, FL may frequently encounter unexpected user dropouts because it is implemented over a large-scale network. We first observe that user dropouts of an FL network may lead to failure in achieving the desired level of privacy protection, i.e., over-consumption of the privacy budget. Subsequently, we develop a DP mechanism robust to user dropouts by dynamically calibrating noise with account of the dropout rate. We evaluate the proposed technique to train convolutional neural network models on MNIST and FEMNIST datasets over a simulated FL network. Our results show that our approach significantly improves privacy guarantee for user dropouts compared to existing DP algorithms on FL networks.

INDEX TERMS Differential privacy, federated learning, user dropout, noise calibration.

I. INTRODUCTION

Progress in machine learning (ML) is transforming the world we live in and the way we operate our businesses. We are experiencing vast adoption of ML at scale to drive change in the future. Data play an essential role in developing ML, and frequently, personal user data drive the development of an ML algorithm. Therefore, service providers attempt to collect substantial data from a user and utilize it for ML algorithms. This approach immediately puts the privacy of the users at risk because service providers can access raw data, which frequently contain private and sensitive information.

To address these problems, federated learning (FL) has been proposed to restrict the management of personal user data while training an ML algorithm for a particular service. FL networks use devices to collaboratively train a shared ML model without sharing sensitive data across multiple computing devices and services. FL is considered as a very promising privacy-preserving technology for ML. Several technology companies, such as Google and Apple, are commercially deploying FL in their products and for promoting their interest in improving privacy [1]–[5].

In principle, FL improves data privacy; however, there are still several challenges that need to be addressed to ensure privacy in an FL network. First, the update vectors of a user are sensitive because some information of the training data of the user can be revealed by the update vectors. Second, the training data of a user can also be inferred from the final output model of the server in FL. There have been studies on privacy concerns regarding the above threats, such as [6]–[8]. Prior studies to protect privacy in FL have outlined the approaches to 1) securely aggregate local update vectors against the server [9]–[12] and 2) generate a model with differential privacy (DP) in FL networks [13]–[17].

In this paper, we focus on developing an enhanced DP mechanism for FL (DP-FL). For the construction of DP-FL, two basic requirements have been considered in the literature. The first requirement is *distributed DP-noise*

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar.

generation [9]–[11], [18], [19]. In the standard centralized setting, the server generates DP noise and adds it to the trained model to make it differentially private. However, DP noise addition by the server is not preferred because the users must trust the server. Distributed DP-noise generation alleviates the need for each client to trust a central noise source. The second requirement is *user-level privacy* [14], [17]. Typical DP mechanisms guarantee the privacy of each individual example (example-level privacy). However, a user in FL contributes a large amount of data and has interest in the protection of the overall dataset. Therefore, privacy guarantee needs to be ensured at the user level (user-level privacy). A key outcome of ensuring user-level privacy guarantee is the ability to limit an attacker from distinguishing two models with the presence or absence of the entire dataset of the user.

In this paper, we investigate a new privacy risk for FL. An FL network typically consists of a large number of users and requires real time communication between the server and the users over it. Bonawitz *et al.* [1] have tested their FL system with a few hundred devices based on the work of McMahan *et al.* [20]. They reported dropout rate varies between 6% and 10% on average due to computation errors, network failures, or changes in eligibility. Therefore it necessarily happens that some devices in an FL network fail to complete the training within a given time or train the model, and frequently leave the network during the training process.

We first observe that user dropouts of an FL network can severely degrade the level of privacy protection in the notion of DP. Intuitively, to satisfy DP over a distributed setting, users add distributed noise sampled from a Gaussian distribution with a small variance to the update vectors. Thus, the aggregation of noise vectors is distributed over a Gaussian distribution with an appropriate variance, which is yielded by predetermined DP parameters ϵ and δ. However, if some users leave the network during aggregation, the aggregated noise vector follows a Gaussian distribution with a smaller variance than expected. This decreases the privacy protection level with respect to DP. Therefore, it is necessary to control the noise when user dropouts occur FL with a distributed DP noise mechanism. Our contributions are as follows:

- A new privacy requirement for a DP mechanism over an FL network is identified, which is referred as *DP-robustness toward user dropouts*. To rigorously analyze the privacy reduction due to user dropouts, we elaborate the concept of sensitivity under an FL setting. Specifically, we observe that the number of users participating in an FL network needs to be considered as one of the parameters. Therefore, it is considered when defining an adjacent dataset for the sensitivity of function. We find that the variance of an aggregated noise vector is reduced by a factor of $\frac{n'}{n}$ when the number of users decreases from n to n' due to dropouts, which leads to over-consumption of the privacy budget.
- We develop a DP-FL method robust to user dropouts over a distributed setting. We build it on a distributed version of the DP-FL solution with user-level privacy, that was proposed by McMahan *et al.* [17]. Our mechanism allows alive users to refresh the noise vectors so that an aggregate noise vector satisfies the expected variance against the dropout of a part of the initial users. Consequently, the proposed mechanism satisfies DP-robustness toward user dropouts as well as distributed noise generation and user-level privacy for DP-FL. We summarize the propose DP-FL method on the privacy requirements compared to notable prior works in Table 1.
- We perform intensive experiments to demonstrate the proposed DP-FL. We evaluate it for training convolutional neural network (CNN) models on MNIST [21] and FEMNIST [22] datasets over a simulated FL network. The evaluation shows that the proposed mechanism improves privacy protection by DP compared to the existing DP-FL methods without significant accuracy loss.

TABLE 1. Comparison of DP-FLs with respect to privacy requirements.

Algorithm	[9]	[14]	[17]	[10]	[11]	This work
Distributed Noise	✔	✗	✗	✔	✔	✔
User-level Privacy	✗	✔	✔	✗	✗	✔
Robustness towards User Dropouts	✗	✗	✗	✗	✗	✔

The remainder of the paper is organized as follows. We reviews some related work in Section II. In Section III, we provide the background on DP, FL, and a threat model on an FL network. In Section IV, we analyze the sensitivity of average function for FL and identify a new privacy risk due to user dropouts from the network. We then present our DP-FL solution robust to user dropouts in Section V. Section VI presents our experimental results with two datasets, MNIST and FEMNIST, to demonstrate the performance and privacy guarantees of our solution. Finally, this paper concludes with some remarks in Section VII.

II. RELATED WORKS

Recent studies show that deep neural networks (DNNs) can memorize the training dataset for CNNs [8] and recurrent neural networks (RNNs) [6]. Carlini *et al.* [6] presented the testing methodology for detecting memorization, and showed that DP is effective in preventing it. Shokri and Shmatikov [7] presented membership inference attacks to infer whether a record is in the model training dataset.

DP is a well-known tool for addressing privacy problems because it offers provable privacy guarantee [23]–[25]. It ensures that the output of the algorithm does not significantly depend on any particular data of the user. Differentially private learning has been studied extensively to ensure the privacy of output models. DP solutions for convex problems have been reported [26]–[28].

For DNNs in the centralized setting, Abadi *et al.* [29] first presented differentially private SGD with example-level

privacy. They proposed the moment accountant method to tightly track the overall use of privacy budget via the random sampling process. Yu *et al.* [30] proposed a differentially private DNNs based on the notion of concentrated differential privacy (CDP). They also suggested a dynamic privacy budget allocation to adjust the privacy budget for every training iteration and empirically demonstrated that the technique achieves better model accuracy while retaining the same privacy guarantee. In addition, Papernot *et al.* [31] and Wu *et al.* [32] proposed differential private machine learning with tighter DP guarantee. They suggested a new noisy aggregation mechanism with selective noise and provided a new analysis of sensitivity, to obtain better model accuracy under the same privacy guarantees.

Several studies have conducted on DP-FL. Earlier works in the literature include [13], [14]. Shokri *et al.* [13] first proposed a DP mechanism for the distributed setting. In their architecture, participants selectively share the trained model parameters using a Laplace mechanism. Geyer *et al.* [14] and McMahan *et al.* [17] studied user-level privacy for FL and proposed DP-FL mechanisms supporting it based on [29]. Agarwal *et al.* [15] presented a communication efficient method for DP-FL. Their work considered distributed noise generation, however, assumed that the server is not curious about an individual update vector. Thakkar *et al.* [16] proposed the use of an adaptive clipping norm method when determining a clipping threshold with user-level privacy in DP-FL. Wei *et al.* [33] presented a DP-FL mechanism, which applies a Gaussian mechanism on the user side to provide example-level DP for each update vector. Since it does not consider distributed DP-noise generation, it necessarily require a huge privacy budget overall (for instance, from 50 to 100). Huang *et al.* [34] presented a DP mechanism with example-level privacy focusing on the unbalanced data scenario in FL. Their solution adaptively allocates the noise during each iteration of SGD to improve accuracy of a trained model. Zheng *et al.* [35] recently proposed a DP-FL mechanism satisfying example-level privacy. Their mechanism is based on GDP (Gaussian differential privacy) [36] that is motivated by the hypothesis testing interpretation of DP. We note that above prior works assume that the server is a trusted third party in the sense that individual update vectors are clearly aggregated by the server.

There have been several studies to apply local DP (LDP) to FL [37], [38]. Bhowmick *et al.* [37] introduced limited version of LDP to ensure stronger privacy. It limits the ability of potential adversaries and achieves better model accuracy than the original version of LDP. Li *et al.* [38] recently proposed a LDP algorithm for gradient-based parameter transfer in the context of meta-learning, which can be applied to FL with personalization, while retaining provable transfer learning guarantee in convex settings. However, despite these attempts, algorithms using LDP still have lower model accuracy compared to global DP algorithms. Therefore, instead of using LDP, we applied global DP using distributed noise generation and secure multiparty computation (SMC) to provide both model accuracy and high privacy.

Concurrently, cryptographic methods have been suggested for data confidentiality for machine learning. They focus on the methods for performing learning or inference on encrypted data by homomorphic encryption [39]–[41] or SMC [42]–[45].

There are several works in the literature to provide data confidentiality for users for FL. Bonawitz *et al.* [9] proposed a secure aggregation method for privacy preserving FL. Truex *et al.* [10] and Xu *et al.* [11] proposed solutions to provide both data confidentiality during computation and DP with example-level privacy. Both works generate DP-noise in a distributed way to reduce noise size while maintaining the same DP guarantee and securely aggregate update vectors using homomorphic encryption and functional encryption [12], respectively. Mugunthan *et al.* [18] designed an FL simulator. It supports distributed noise generation for DP and secure aggregation from SMC. However it only considers a Laplace mechanism, which is not applicable to DNNs. Wang *et al.* [19] proposed a DP-FL mechanism which supports distributed noise generation for DP and secure aggregation from SMC. Their work adds discrete Gaussian noise into client updates, which leads to communication efficiency in FL over the previous work by Agarwal *et al.* [15]. We remark that these prior works for DP-FL with secure aggregation do not consider privacy degradation of DP when users drop out. They only guarantee the correctness of aggregation in the case.

III. PRELIMINARIES

A. DIFFERENTIAL PRIVACY

DP is the concept about privacy guarantees of an individual data for algorithms on aggregate databases. Informally, an algorithm is said to be differentially private if the inclusion of a single individual record in the dataset does not give a statistically significant effect on the output of the algorithm. The formal definition of DP was introduced by Dwork *et al.* [24].

Definition 1 (Differential Privacy (DP)): A randomized mechanism $M : \mathcal{D} \mapsto \mathcal{R}$ is (ϵ, δ)-differentially private (DP) if for any adjacent $D, D' \in \mathcal{D}$ and $S \subset \mathcal{R}$ it holds that

$$\Pr[M(D) \in S] \leq e^{\epsilon} \Pr[M(D') \in S] + \delta.$$

When $\delta = 0$, we say that a mechanism is ϵ-differentially private. Here, ϵ is a privacy parameter called the privacy budget that determines the strength of privacy protection.

In the above definition, the meaning of ***adjacent*** depends on the algorithms or applications. In ML, it is usually used in two meanings depending on the model of privacy provided, i.e., example-level privacy and user-level privacy. In example-level privacy which is provided by most prior works [10], [29], [32], [46] on differentially private ML, two datasets D and D' are adjacent if D' is formed by adding or removing a single training example from D. User-level privacy [17] focuses on protection of whole user data on the

training set. In user-level privacy, two datasets D and D' are adjacent if D' is formed by adding or removing all of the examples associated with a single user from D.

The standard approach to achieve DP is adding noise whose size is proportional to the sensitivity of the output. The sensitivity measures the maximum of the output difference between two adjacent datasets. We denote the sensitivity of f by sen_f. We use the Gaussian mechanism to achieve DP for our construction for DP-FL in Section V.

Definition 2 (Gaussian Mechanism): Let $f : \mathcal{D} \to \mathbb{R}^d$ be a d-dimensional function of ℓ_2-sensitivity sen_f. Define the Gaussian mechanism with parameter σ for f as

$$M(D) = f(D) + \mathcal{N}(0, sen_f^2 \sigma^2 \mathbf{I}),$$

i.e., the mechanism adds noise scaled to $\mathcal{N}(0, sen_f^2\sigma^2)$ to each of the d components of the output.

The following theorem is the well-known result related to (ϵ, δ)-DP using a single application of the Gaussian mechanism.

Theorem 1 [25]: If $\epsilon \in (0, 1)$ and $\sigma^2 \geq 2 \log \frac{1.25}{\delta}/\epsilon^2$, the Gaussian mechanism with noise parameter $sen_f\sigma$ for a function f satisfies (ϵ, δ)-DP.

When a mechanism is performed multiple times, the evaluation of the privacy budget follows the composition theorem. In this paper, we use the moment accountant proposed in [29] to evaluate the privacy budget used in our algorithm.

B. DNN

A DNN is a type of artificial neural networks to define a function parameterized by a model parameters. A DNN has a multi-layered structure defined by the synthesis of fully-connected operations specified as affine transformations and nonlinear activation functions such as sigmoids and ReLUs. In the typical training process, a set of input and output examples is fed to the DNN to adjust a model so that the loss function $\mathcal{L}$ on a given model becomes less than a target loss over an input set of data points. For the train of a model, traditional DNNs employ the centralized setting, i.e., a server collects a large amount of data and trains a model from the collected data. For details of various techniques of DNNs, we refer to [47].

CNNs are one of major classes of DNNs to solve computer vision tasks, i.e., recognition and classification of the image. A CNN is configured to perform a classification operation on the filtered image after applying a filtering technique to the original image by adding a new layer called a convolutional layer and a pooling layer before the fully-connected layer. The convolution layer applies a filtering technique to the image and the pooling layer performs various functions such as reducing the size of the image by converting local parts of the image into one representative scalar value. We demonstrate our DP-FL mechanism over a CNN in Section VI.

C. FEDERATED LEARNING

FL is a machine learning technique that uses multiple local devices to train a shared model without sharing their own data. In centralized ML, service providers collect data from users and train a model upon it. Thus, the centralizes setting results in privacy risk for the data of the user. However, in FL, because each device (user) does not send raw data to the central server, FL can prevent the risk caused by the service providers' access to raw data. For this reason, many IT companies consider FL as a core technology for privacy-sensitive commercial services and make a lot of effort to study and develop it.

Algorithm 1: Local Update

Parameters:
  B, E, η, C
Clip(Δ):
  return $\min(1, \frac{C}{\|\Delta\|}) \cdot \Delta$
Local Update(user i, initial model θ):
  $\mathbf{w} \leftarrow \theta$
  for *each local epoch* $i = 1$ **to** E **do**
    $\mathcal{B} \leftarrow$ {split i's data into batches of size B}
    for *batch* $b \in \mathcal{B}$ **do**
      $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \mathcal{L}(\mathbf{w}; b)$
      $\mathbf{w} \leftarrow \theta + Clip(\mathbf{w} - \theta)$
  return $\mathbf{w} - \theta$

In this paper, we focus on federated averaging [20] as an algorithm to perform FL. Federated averaging is a method of training the global model by taking the average of the update vectors obtained from local update performed by each user. In local update, the user splits her own data into the size of batch and repeatedly performs gradient descent for each batch. Gradient descent is a method that computes the gradient of the loss function $\mathcal{L}$ and updates the parameters until the loss function converges to local minimum. We add clipping step after gradient descent for privacy guarantees, as usual ML algorithms that include privacy protection. Briefly, the local update is performed as described in Algorithm 1.

D. THREAT MODEL

We consider the following threat model. It includes the assumption for behavior of each participant in DP-FL and SMC.

- *Server*: We assume that the server is *honest-but-curious*, which is a common assumption [9]–[11]. That is, the server follows the protocol correctly, but may try to learn some information about the data of users. The server can also use any information to improve performance of learning.
- *User*: We assume that users can collude to learn private information of others. The users also may try to learn any private information from the final outputs. In addition, they want their private data to be kept at the level of privacy protection promised by the coordinator system.
- *Trusted third party (TTP)*: We do not need TTP for DP mechanism by utilizing SMC without TTP. For instance,

in [9], TTP is not required for honest-but-curious adversaries.

IV. PRIVACY RISK DUE TO USER DROPOUTS IN FL

A. ANALYSIS OF SENSITIVITY FOR FEDERATED AVERAGE

The sensitivity of a function is the maximum output difference between any two adjacent datasets. In this section, we revisit the concept of sensitivity and elaborate it under an FL setting. We start with the formal definition of sensitivity. In this paper, sensitivity is the ℓ_2-sensitivity.

Definition 3 (Sensitivity): Let $f : \mathcal{D} \to \mathbb{R}^d$ be a d-dimensional function. The sensitivity of f is defined as

$$sen_f = \max_{D,D' \in \mathcal{D}} \|f(D) - f(D')\|_2$$

where D and D' differ in only a single entry.

The definition of sensitivity is the maximum effect of one entry on the output of the function, for all the datasets in $\mathcal{D}$. In the definition, $\mathcal{D}$ depends on the application of interest. For instance, when f is a summation, the effect of an entry x of a dataset D is $|x|$ for any dataset D. However, in the case of average function f, the effect of an entry x of a dataset D depends on the number of entries in D. For example, if $|D| = 10$, the effect of x is $\frac{|x|}{10}$, and if $|D| = 10^4$, the effect of x is bounded by $\frac{|x|}{10^4}$. Therefore, in the case of average, it is required to specify the size of D used in the application.

We are interested in rigorously describing $\mathcal{D}$ in an FL setting. We focus on federated average of the elements in D, where the coordinator system specifies the size of D in advance and makes it public. More precisely, in federated averaging, each device sends a local update vector obtained by training its own data to the server. The server subsequently computes the (weighted) average of the update vectors received from the devices. In the process, DP noise is added to the output to provide privacy protection, and the server notifies the number of devices participating in the learning for the distributed DP-noise generation. We note here that the number of devices is determined and made public in advance. Therefore, the sensitivity should be computed only for the set of devices satisfying the condition for the number of devices.

We analyze the sensitivity of an average function in an FL setting, denoted as $f_{\text{avg}(n)}$. The function computes the average of n update vectors, where number n is a predetermined parameter. In this case, $\mathcal{D}$ consists of sets D of n vectors. As adjacent datasets for D, we may only consider the datasets in which one entry is replaced by another, and not the datasets that add or subtract one entry from D. The following proposition shows the sensitivity of $f_{\text{avg}(n)}$.

Proposition 1: Let $f_{avg(n)} : \mathcal{D} \to \mathbb{R}^d$ be a d-dimensional function such that $\mathcal{D}$ consists of sets D of n d-dimensional vectors and $|x| \le C$ holds for all $x \in D$ and some constant C. Thus, the sensitivity of $f_{avg(n)}$ is bounded by $\frac{2C}{n}$.

Proof: As adjacent datasets for a given dataset, we only consider the datasets in which one entry is replaced to another. This is due to constraint on the number of vectors. For any set $D = \{x_1, x_2, \cdots, x_n\}$ of n d-dimensional vectors, we consider an adjacent set $D' = (D \setminus \{x_n\}) \cup \{x'\}$ where $x' \notin D$, without loss of generality.

Then We have

$$\begin{aligned}\|f(D') - f(D)\|_2 &= \left\| \frac{1}{n}\left(\sum_{i=1}^{n-1} x_i + x'\right) - \frac{1}{n}\sum_{i=1}^{n} x_i \right\|_2 \\ &= \frac{\|x' - x_n\|_2}{n} \\ &\le \frac{2C}{n}\end{aligned}$$

Note that the last inequality is from the triangle inequality. This shows that the sensitivity of $f_{\text{avg}(n)}$ is bounded by $\frac{2C}{n}$. ∎

From the above result, we remark that calculating the sensitivity of the average function in consideration of the condition for the number of update vectors gives a significant difference to the bounds of the sensitivity. Suppose $\mathcal{D}$ has no condition for the number of vectors. One needs to compute that the sensitivity of the average function over $\mathcal{D}$ is $\frac{2C}{n_0}$ where n_0 is the smallest number of vectors in the dataset belonging to $\mathcal{D}$. Then the bound of the sensitivity is a constant regardless of the number of users actually participating in FL. This is because the sensitivity is computed over all the domain $\mathcal{D}$. However, by taking account for the number of update vectors, a tighter bound can be applied to the sensitivity, which leads to more precise computation of the privacy budget.

One can easily extend the above result to the weighted average function $f_{\text{wavg}(n)}$. $f_{\text{wavg}(n)}$ computes the weighted average of the vectors in D, where the weighted sum of the vectors is equal to n. $f_{\text{wavg}(n)}$ can be used for federated averaging using the weighted data, which is useful when the number or importance of the data that each user trains for learning is different. There is a similar bound on the sensitivity of $f_{\text{avg}(n)}$ in average case.

Proposition 2: For a d-dimensional vector x, $w(x)$ denotes the weight corresponding to x, where $|x| \le C$ and $w(x) \le W$ for some constants C and W. Suppose $D \in \mathcal{D}$ is a set that consists of d-dimensional vectors such that $\sum_{x_i \in D} w(x_i) = n$ for a given number n. Let $f_{wavg(n)} : \mathcal{D} \to \mathbb{R}^d$ be the weighted average function defined as

$$f(D) = \frac{\sum_{x_i \in D} w(x_i) x_i}{\sum_{x_i \in D} w(x_i)}.$$

Thus, the sensitivity of $f_{wavg(n)}$ is bounded by $\frac{2WC}{n}$.

Proof: Similar to Proposition 1, the dataset has constraints on the sum of weights. As adjacent datasets, we only consider the datasets in which one entry is replaced to another entry with the same weight. Let D be a set of d-dimensional vectors such that $\sum_{x_i \in D} w(x_i) = n$ for a given number n. Then, we consider an adjacent set $D' = (D \setminus \{x_k\}) \cup \{x'\}$ where $x_k \in D$, $x' \notin D$ and $w(x_k) = w(x')$, without loss of generality.

TABLE 2. Privacy for different dropout rate. For each row, 50 out of 5,000 users are sampled initially, and some users are dropped out with each dropout rate during the local update process. We set $\delta = 10^{-5}$ and compute ϵ for which (ϵ, δ)-DP holds after several rounds. We also compute the maximum number of rounds without exceeding several privacy budget.

Dropout rate	ϵ for (ϵ, δ)-DP after several rounds						max # of rounds in fixed privacy budget		
	1	10	10^2	10^3	10^4	10^5	$\epsilon < 2.0$	$\epsilon < 4.0$	$\epsilon < 8.0$
No dropout	1.317	1.414	1.612	2.538	7.429	28.552	429	2952	11479
10%-dropout	1.467	1.586	1.822	2.855	8.212	31.888	216	2348	9520
30%-dropout	1.894	2.089	2.463	3.867	10.703	42.623	3	1110	5690

Then We have

$$\begin{aligned}\|f(D') - f(D)\|_2 &= \left\| \frac{1}{n} \sum_{x_i \in D'} w(x_i)x_i - \frac{1}{n} \sum_{x_i \in D} w(x_i)x_i \right\|_2 \\ &= \frac{\|w(x')x' - w(x_k)x_k\|_2}{n} \\ &\leq \frac{2WC}{n}\end{aligned}$$

Note that the last inequality is from the triangle inequality. This shows that the sensitivity of $f_{\text{wavg}(n)}$ is bounded by $\frac{2WC}{n}$. ■

B. PRIVACY DEGRADATION CAUSED BY CHANGE IN SENSITIVITY IN FL

In FL, user dropouts can occur frequently owing to the unreliable network conditions or the device status. User dropouts change the number of users participating in the federated averaging. This can change the effect that the data of one user has on the final output of the federated averaging, thereby leading to overspending the privacy budget. Therefore, we apply the concept of $f_{\text{avg}(n)}$ or $f_{\text{wavg}(n)}$ with respect to the corresponding dropout conditions, for providing a tight privacy budget. For the case in which the data of the user are not weighted explicitly, we assume that all data has the same weight. Specifically, all weights are assumed to be 1 as in [17]. Therefore, we only deal with $f_{\text{wavg}(n)}$ in the remainder of the section.

Suppose that each user i in user set U has his/her own update vector Δ_i with associated weight w_i. We let function f be the weighted average of the update vectors of the user, as in Proposition 2. Suppose n is the weight sum of the users participating in the FL and that the size of the update vector of each user is bounded by some constant C from the clipping process. We may assume that the weight of each user does not exceed 1, without loss of generality. Here, the weight sum, n, is notified to all the users participating in the FL so that each user can generate a distributed noise. Therefore, it can be a fixed condition when estimating the bound of the sensitivity. Furthermore, by Proposition 2, the sensitivity with respect to the condition on weight sum $f_{\text{wavg}(n)}$ is bounded by $\frac{2C}{n}$.

In central DP setting, the server adds the Gaussian noise sampled from $\mathcal{N}(0, \sigma^2\mathbf{I})$, where $\sigma = \frac{2C}{n} \cdot z$, with some noise multiplier z. However, because FL is a distributed network, it is preferred that the users perform this without trusting the server. Specifically, each user samples his/her own noise independently so that their weighted average draws $\mathcal{N}(0, \sigma^2\mathbf{I})$. A noise vector can be simply sampled from $\mathcal{N}(0, \sigma^2\mathbf{I})$ in a distributed manner because the sum of the independent normal random variables is also distributed normally. More precisely, each user i samples e_i from $\mathcal{N}(0, nw_i\sigma^2\mathbf{I})$ and generates a noise update vector $w_i\Delta_i + e_i$. Since $n\sigma^2 \sum w_i = n^2\sigma^2$, the noise part of the weighted average draws $\mathcal{N}(0, \sigma^2\mathbf{I})$.

Now, suppose some users drop out, and $n'(< n)$ is the weight sum of the alive users. Here, the changed weight sum, n', is also notified to the alive users, and it can be considered that the condition on the weight sum is changed. Hence, we should compute the sensitivity of $f_{\text{wavg}(n')}$, and not of $f_{\text{wavg}(n)}$. Therefore, it is bounded by $\frac{2C}{n'}$.

In this case, to maintain the same level of privacy as before user dropouts occur, the noise distribution of the final output should be $\mathcal{N}(0, \sigma'^2\mathbf{I})$, where $\sigma' = \frac{2Cz}{n'}$. However, the weighted average of the e_is of the alive users draws $\mathcal{N}(0, \frac{n'}{n}\sigma'^2\mathbf{I})$, which is obtained from

$$\frac{1}{n'^2} \sum_{\text{alive users}} nw_i\sigma^2 = \frac{n}{n'}\sigma^2 = \frac{n'}{n}\sigma'^2.$$

The reduction in the noise variance by a factor of $\frac{n'}{n}$ causes over-consumption of the privacy budget at each round of the entire FL work. Thus, unless the number of rounds is cut, it fails to achieve the expected privacy protection by DP. Table 2 summarizes the change in ϵ and the maximum number of rounds that a user can participate in learning in a fixed privacy budget based on the user dropout rate.

It is worth noting that the variance of the noise of the final output is $\frac{n}{n'}\sigma^2$, which is larger than σ^2 of the case of no user dropout. Therefore, if we do not consider the change in the sensitivity, it can be misleading that the mechanism already provides sufficient privacy protection. However, by considering the change in the sensitivity due to user dropouts, it can be seen that the size of the noise is insufficient to achieve the expected privacy protection. Thus, it is essential to estimate the sensitivity considering the condition of the weight sum.

V. DP MECHANISM ROBUST TO USER DROPOUT

In this section, we present our DP-FL mechanism, which is DP-robust to user dropouts. We first give a brief description of the algorithm and then provide a privacy analysis of our algorithm.

A. ALGORITHM DESCRIPTION

Our approach is built on the differentially private federated averaging suggested by McMahan *et al.* [17]. Their method allows user-level privacy by applying the DP mechanism to the update vectors, and not directly on the training examples of the users. However, their solution assumes that the server generates a differentially private model following the collection of all the update vectors from the users. Thus, we aim to provide 1) a distributed DP-noise generation method from users and 2) robustness to user dropouts over their construction.

Our construction is presented in Algorithm 2. Similar to the construction of McMahan *et al.*, ours considers the weighted average of the update vectors obtained from a sampled user set. Each user i in user set U has his/her own dataset with associated weight w_i. The weight quantifies the degree of effect of the data of the user on the overall learning. We set it to be proportional to n_i, the number of data the user has, and set its maximum as 1. Specifically, we define the weight of user i as $w_i = \min\{\frac{n_i}{n_{max}}, 1\}$, where n_{max} is the limit number of the data used for the training per user. The server coordinates FL to train the model with the users, who are typically sampled by the server from all the candidate users with probability q. This sampling process is important for the use of the moment accountant [29], which provides a tight bound of the privacy loss. The server shares the common initial model and the hyper-parameters for training. All the sampled users are provided the common initial model, θ_0, and the public hyper-parameters: number of epochs E, batch size B, learning rate η, and clipping bound C. The distributed DP-noise generation is described from Line 13 to Line 16 in the algorithm. Let U^t be the set of sampled users for the round t. At Line 15, each user $i \in U^t$ independently samples the e_i from $\mathcal{N}(0, nw_i\sigma^2\mathbf{I})$. The user subsequently computes a noise local update vector $M_i := w_i\Delta_i + e_i$, where Δ_i is a plain update vector.

Even though each update vector $w_i\Delta_i$ is noised with e_i, this does not guarantee the confidentiality of $w_i\Delta_i$, which cannot be provided by DP. It should be noted that the server is only interested in the average of noised update vectors M_i. Hence, instead of sending a plain M_i to the server, our algorithm performs SMC at Line 16. SMC allows the computation of the weighted average of the update vectors of the users without revealing any information about individual update vectors. In particular, the functionality of our interest is the secure aggregation or averaging of the input vectors of the users supporting user dropouts. Any SMC with these properties can be employed in our construction, and we have adopted SMC in [9] proposed by Bonawitz *et al.*

Finally, we perform the noise calibration process to refresh an aggregated noise vector when the weight sum of the users decreases from n to n' due to the user dropouts during Round t. This process is given from Line 17 to Line 24 in the algorithm. Each user i knows his/her noise vector e_i generated at initially (Line 15). Hence, each user can sample new noise

Algorithm 2: DP-FL Robust to User Dropouts

1 **Parameters:**
2 noise multiplier z
3 user example limit n_{max}
4 clipping bound C
5 user sampling probability q

6 **Main Algorithm:**
7 Initial model θ^0, user set U
8 For each $i \in U$, i has n_i examples and $w_i = \min\{\frac{n_i}{n_{max}}, 1\}$
9 **for** *each round* $t = 0, 1, 2, \ldots$ **do**
10 **Server:**
11 $U^t \leftarrow$ sample user set with probability q
12 Broadcast to $U^t : n \leftarrow \sum_{i\in U^t} w_i$, $\sigma \leftarrow \frac{2C}{n}z$
13 **User:** *each user* $i \in U^t$ **do**
14 $\Delta_i^{t+1} \leftarrow$ **UserUpdate**(i, θ^t)
15 $e_i^{t+1} \leftarrow \mathcal{N}(0, nw_i\sigma^2\mathbf{I})$
16 Send $M_i \leftarrow w_i\Delta_i^{t+1} + e_i^{t+1}$ (via SMC)
17 **Server:**
18 $U_A^t \leftarrow \{$alive users $\in U^t\}$
19 Get Δ^{t+1} using SMC on $\{M_i \mid i \in U_A^t\}$
20 Broadcast to $U_A^t : n' \leftarrow \sum_{i\in U_A^t} w_i$, $\sigma' \leftarrow \frac{2C}{n'}z$
21 **User:** *each user* $i \in U_A^t$ **do**
22 Send $e'^{t+1}_i \leftarrow -e_i^{t+1} + \mathcal{N}(0, n'w_i\sigma'^2\mathbf{I})$
23 **Server:**
24 Share $\theta^{t+1} \leftarrow \theta^t + \Delta^{t+1} + \frac{1}{n'}\sum_{i\in U_A^t} e'^{t+1}_i$

vector e'_i from

$$\mathcal{N}(-e_i, \frac{n^2}{n'}w_i\sigma^2\mathbf{I}) = -e_i + \mathcal{N}(0, n'w_i\sigma'^2\mathbf{I}),$$

where $\sigma = \frac{2Cz}{n}$ and $\sigma' = \frac{2Cz}{n'}$, respectively. Subsequently, each user sends it to the server, and the server applies it on the final output (Line 24). The noise calibration process aims to change the noise distribution of the final output from $\mathcal{N}(0, \sigma^2\mathbf{I})$ to $\mathcal{N}(0, \sigma'^2\mathbf{I})$.

B. PRIVACY ANALYSIS

Because we add M_i's using SMC, it is trivial that the server cannot access the value of each M_i. Thus, in this subsection, we focus on the DP analysis for the final output. We use the moment accountant as a privacy accountant method. Moment accountant additively accumulates the log values of the moments of the privacy loss at each round and uses this to compute the privacy budget of the entire algorithm. Therefore, in order for the algorithm to maintain the privacy level as desired, it is necessary to ensure that the privacy loss of each round does not become larger. However, if some users

drop out at a round and the weight sum of the alive users decreases from n to n', the weighted average of the e_is of the alive users draws $\mathcal{N}(0, \frac{n'}{n}\sigma'^2\mathbf{I})$ as in the previous section. For the desired privacy level, the weighted average of the DP noise of the alive users should draw $\mathcal{N}(0, \sigma'^2\mathbf{I})$, which means the privacy loss at this round becomes larger than expected.

The noise calibration process plays a role in reducing such increased privacy loss. At Line 24, we have

$$\Delta^{t+1} + \frac{1}{n'}\sum_{i\in U_A^t} e_i'^{t+1} = \bar{M}_i + \frac{1}{n'}\sum_{i\in U_A^t}(e_i^{t+1} + e_i'^{t+1})$$

where $\bar{M}_i$ is the weighted average of the update vectors of the alive users. Since $e_i'^{t+1}$ is sampled from $-e_i^{t+1} + \mathcal{N}(0, n'w_i\sigma'^2\mathbf{I})$, the final output $\Delta^{t+1} + \frac{1}{n'}\sum_{i\in U_A^t} e_i'^{t+1}$ draws $\bar{M}_i + \mathcal{N}(0, \sigma'^2\mathbf{I})$.

Because the server is assumed to be honest-but-curious, it can access not only final model parameters but also intermediate model parameters during an algorithm execution. Note that intermediate model parameters do not reflect the noise calibration process, hence, suffer from weaker privacy for user data. Therefore it is necessary that intermediate parameters are unavailable to the server. One approach to force the server to add noise is that each user first adds a noise vector with a large variance to the update vector and subsequently reduces it by a noise calibration process as follows. Let n_0 be the minimum weight sum of the users such that FL works, i.e., $n, n' \geq n_0$.[1] Given n_0, each user i samples e_i from $\mathcal{N}(0, n_0 w_i \sigma_0^2\mathbf{I})$ instead of $\mathcal{N}(0, n\, w_i\sigma^2\mathbf{I})$ at Line 15, where $\sigma_0 = \frac{2C}{n_0}z$. Subsequently, the user selects e_i' from $-e_i + \mathcal{N}(0, n'w_i\sigma'^2\mathbf{I})$ at Line 22. Since $n_0 w_i\sigma_0^2 \geq n'w_i\sigma'^2$, adding $e_i + e_i'$ is more advantageous than adding e_i to the server in terms of accuracy. We provide the full description of the algorithm applying this approach in the appendix.

In practice, user dropouts may occur even during the noise calibration process. We remark that our algorithm addresses the cases of no dropout in the noise calibration process or few dropouts with an acceptable level of privacy degradation. However, one can observe that the computation and communication costs required in the noise calibration process (from Line 21 to Line 22) are quite low than those in the local update process (from Line 10 to Line 20). Thus, because users who survived in the local update process are alive in the subsequent process with high probability, we expect that our algorithm properly works in practice.

VI. EXPERIMENTS

In this section, we access our algorithm to train CNN in a distributed environment. We implement our algorithm using TensorFlow and TensorFlow Privacy. The model built in TensorFlow Privacy [48] is referred as our CNN model. It comprises two convolution layers activated by ReLU, two max pooling layers, and two dense layers. We slightly modified the model to prevent overfitting and improve the training speed by inserting a dropout layer between the last two dense layers in [48]. In addition, we adopt the secure aggregation protocol of Bonawitz *et al.* [9] to perform secure averaging.

[1] It is reasonable to assume the minimum weight sum of the users because an FL work might require a certain number of participants so that the model learns various datasets.

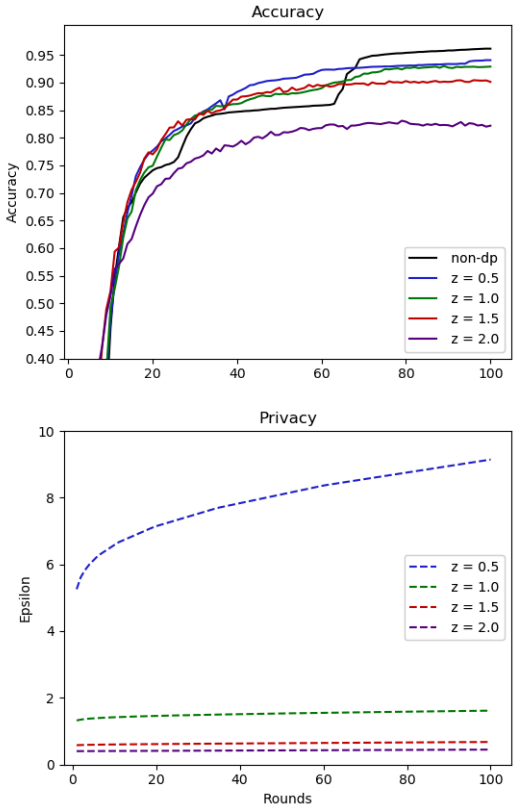


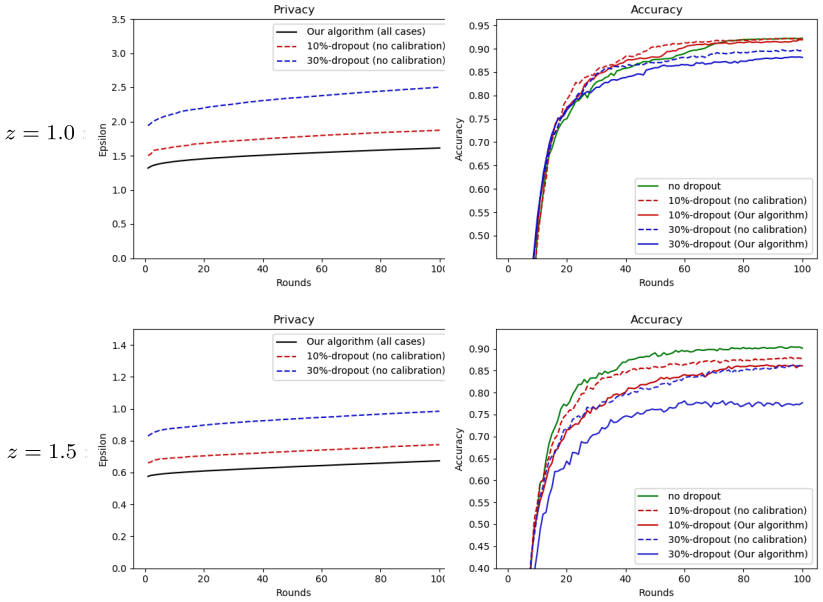
FIGURE 1. Effect of DP noise on MNIST. Figure shows accuracy and the privacy budgets, ϵ, for (ϵ, δ)-DP with $\delta = 10^{-5}$ for 1–100 rounds when noise multiplier z values are 0.5, 1.0, 1.5, and 2.0.

To track the overall use of the privacy budget, we use the moment accountant in [29], which yields a very tight privacy bound. The moment accountant is computed using the RDP accountant available with TensorFlow Privacy [49]. In addition, we selected 50 users from a total of 5,000 to participate in the learning at a sampling rate of 0.01. To present effects of different scenarios on the experimental result, we vary the noise multiplier, the dropout rate, and other hyper-parameters. In the experiments, the dropout rate is the proportion that drives a user to dropout from a local learning process, and it is used for dropout scenario analysis. We consider two dropout rates, 10% and 30%.

The experiments are conducted on two different datasets: MNIST [21] and Federated Extended MNIST (FEMNIST) [22]. The former is used for an independent and identically distributed (i.i.d.) case, whereas the latter is for a non-i.i.d. one.

A. MNIST: i.i.d CASE

The MNIST database is a well-known dataset for image classification. It is a 28 × 28 image dataset of handwritten



Dropout rate		Acc.	ϵ
No dropout		0.9226	1.6118
10%-dropout	Previous	0.9212	1.8719
	Ours	0.9190	1.6118
30%-dropout	Previous	0.8964	2.5005
	Ours	0.8814	1.6118

Dropout rate		Acc.	ϵ
No dropout		0.9013	0.6731
10%-dropout	Previous	0.8776	0.7744
	Ours	0.8610	0.6731
30%-dropout	Previous	0.8601	0.9840
	Ours	0.7768	0.6731

FIGURE 2. Effects of user dropout and noise calibration on MNIST database when $z = 1$ and $z = 1.5$. Figure shows accuracy and accounted privacy budgets ϵ for 1–100 rounds. Table summarizes results after 100 rounds of training.

digits, containing 60,000 and 10,000 training and test examples, respectively. It is not classified based on writers, and hence, in this study, it is randomly distributed to the users and employed for the experiments for the i.i.d case. The maximum number of examples per user is set as $n_{max} = 1,200$. For simplicity, the number of training examples for a user is set as 1, 200. Specifically, $w_i = 1$ is set for each user i. In the experiments, the following hyper-parameter settings are used: $B = 100$, $E = 1$, and $\eta = 0.15$, where B is the batch size, E is the number of the epochs, and η is the learning rate.

1) EFFECT OF DP NOISE

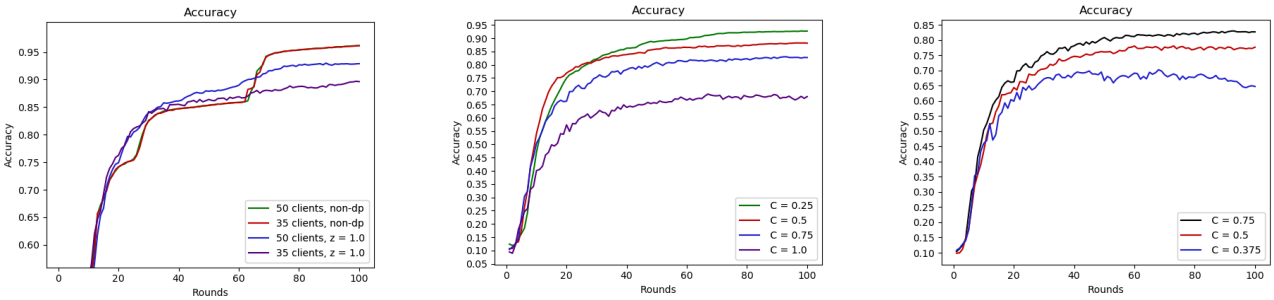
To examine the effect of DP noise on FL, we evaluate the privacy budget and the accuracy of the model for different noise multiplier z values. The weight sum, n, and the clipping bound for the limitation of the sensitivity are set as 50 and 0.5, respectively. To obtain only the effect of DP noise, a case in which there is no user dropout is considered.

Fig. 1 shows the accuracy and ϵ for (ϵ, δ)-DP with $\delta = 10^{-5}$. Experiments are conducted using various noise multiplier z values, i.e., 0.5, 1.0, 1.5, and 2.0. When $z = 0.5, 1.0, 1.5$, and 2.0, the image classification accuracies in 100 rounds are 0.9406, 0.9289, 0.9013, and 0.8218, respectively, and the privacy budgets (ϵ) are 9.1418, 1.6118, 0.6731, and 0.4437, respectively. Considering the accuracy in the non-DP case, i.e., 0.9615, as a baseline, z value of 1.5 or less yields a good result without significant accuracy loss. Specifically, there is an appropriate noise size that does not cause a large accuracy loss when applying DP. These results show that FL using our algorithms is effective and can achieve high accuracy without significant loss by appropriately setting the noise parameters.

2) EFFECT OF USER DROPOUTS

We evaluate the privacy budget and the accuracy of the model to examine the effects of both user dropout and noise calibration. To this end, in the experiments, we only changed the dropout rates and the noise calibration status while maintaining all other parameters fixed. In addition, we set z as 1.0 and 1.5, which are determined to be appropriate from the results of the previous experiments. To adjust the dropout rate, only the predetermined rates of the users, who participated in the learning in the aggregation phase, were considered to be dropped out in the order in which the data are sent, and the users who subsequently transmitted the data. Note that the decision of a user to drop out is not random but related to the communication environment of the user, device status, and other factors. Therefore, it is not reflected it in the sampling rate of the moment accountant. Fig. 2 shows the accuracy and ϵ for (ϵ, δ)-DP with $\delta = 10^{-5}$. Based on the analysis of the experimental results, we obtain the following results.

- The image classification accuracy decreases when user dropouts occur regardless of the noise calibration. In addition, the accuracy decreases as the dropout rate increases. Indeed, when user dropout occurs, hence, the weight sum decreases from n to n', the standard deviations of the noise of the final output with and without noise calibration are $\frac{n}{n'}\sigma$ and $\sqrt{\frac{n}{n'}}\sigma$, respectively. These two standard deviations are larger than σ when there is no user dropout.
- If the noise calibration step is introduced in the algorithm, in most cases, the target privacy level is achieved with only a small accuracy loss. For instance, when $z = 1.0$ and the dropout rates are 10% and 30%, the privacy



(1) the number of users $k\ (=n)$ (2) the clipping bound C (with fixed z) (3) the clipping bound C (with fixed σ)

FIGURE 3. Effect of parameters on MNIST. Figure shows accuracy when one parameter varies, whereas the others are fixed.

budgets, ϵ, reach 1.8719 and 2.5005, respectively. With noise calibration, these are decreased to 1.6118, which is the target ϵ when FL is established. However, the accuracy loss is less than 0.02. In our experiments, the only case in which a large accuracy loss occurs with noise calibration step is when the dropout rate is 30% and $z = 1.5$. The accuracy loss is mainly attributed to the noise applied to the final output and occurs significantly when the noise exceeds a certain level. As shown in Fig. 1, we observed that the accuracy loss is large when the noise applied to the final output is $\frac{z}{n} \geq \frac{2.0}{n}$, if there is no user dropout. In case of user dropout, the weight sum decreases from n to n', and the standard deviation of the noise applied to the final output is proportional to $\frac{z}{n'}$. When $z = 1.5$ and the dropout rate is 30%, $\frac{z}{n'} = \frac{1.5}{0.7n}$ is larger than $\frac{2.0}{n}$, corresponding to the above case in which a large accuracy loss occurs.

3) EFFECTS OF OTHER PARAMETERS

We also examined the effects of changes in other parameters related to privacy guarantee such as the number of users and clipping bounds. Fig. 3 shows the corresponding experimental results. Below we summarize our observations.

- The number of users: To evaluate the effect of number of users, the experimental results obtained with 50 and 35 users were compared. When DP is not applied, the accuracies of the models trained by 35 and 50 users are quite similar. This is because of using the average of the update vectors from the sampled users at each round, and the average of the vectors from 35 users is similar to that from 50 users. However, when some DP noise is added, there is a difference between the accuracies achieved with 35 and 50 users after some training rounds. For each round, although the sizes of the averaged update vectors from 35 and 50 users are very similar, the size of the averaged noise vectors from 35 users is larger than that from 50 users by $\frac{50}{35}$ times. Consequently, when DP is applied, the accuracy decreases if the number of users is reduced. Therefore, increasing the number of users participating during learning can be adopted as an approach to increase the accuracy while maintaining the same DP level.
- Clipping bounds: The clipping bound, C, is varied in two approaches. First, z and the dropout rate are fixed, and C is slightly changed. Because the number of users is fixed, the sensitivity, $\frac{2C}{n}$, is dependent only on C. Thus, the size of the noise vector is proportional to C. Consequently, the experimental results show that the image classification accuracy decreases as C increases.

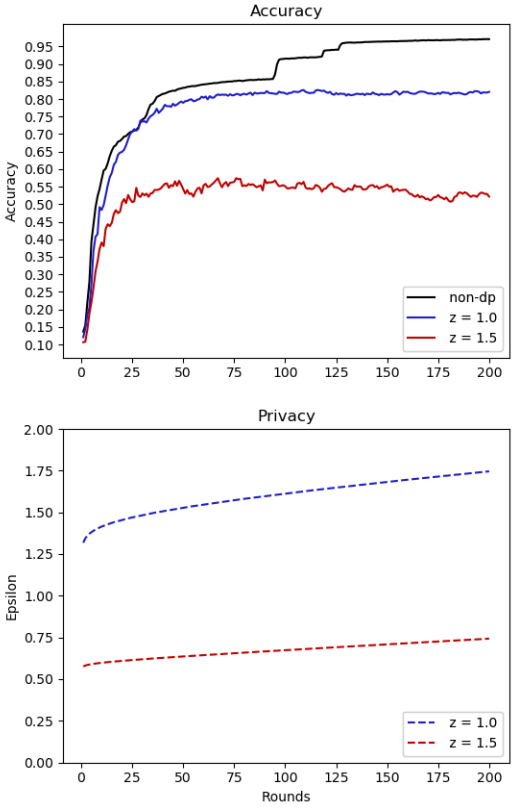
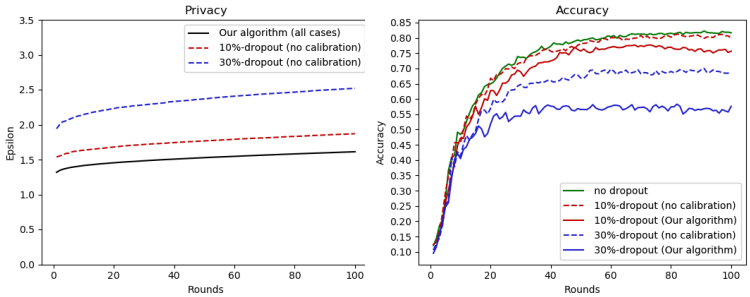


FIGURE 4. Effect of DP Noise on FEMNIST. Figure shows accuracy and privacy budgets ϵ for (ϵ, δ)-DP with $\delta = 10^{-5}$ for 1–200 rounds for noise multiplier z values of 1.0 and 1.5.



Dropout rate		Acc.	ϵ
No dropout		0.8164	1.6118
10%-dropout	Previous	0.7991	1.8696
	Ours	0.7567	1.6118
30%-dropout	Previous	0.6863	2.5199
	Ours	0.5758	1.6118

FIGURE 5. Effects of user dropout and noise calibration on FEMNIST database when $z = 1$. Figure shows accuracy and accounted privacy budgets ϵ for 1–100 rounds. Table lists results after 100 rounds of training.

We also conducted experiments with various values of C while fixing σ instead of z. When σ is fixed, the accuracy does not change with the size of the noise. However, a larger update vector Δ_i is obtained by increasing its clipping bound C, which implies that the size ratio of the update vector, Δ_i, and its noise vector, e_i, increases as C increases. Thus, by setting a high C, high image classification accuracy can be achieved. However, to increase C to enhance the accuracy, z should be decreased because $\sigma = \frac{2C}{n}z$, which increases the privacy budget. Therefore, if the server desires to use a fixed value of σ, it is essential to find the optimal combination of these two parameters.

B. FEMNIST: NON-i.i.d CASE

In this section, we present the evaluation of our mechanism for the non i.i.d case using FEMNIST database [22]. FEMNIST is a dataset for image classification and is built by classifying the data in the Extended MNIST database based on writers. We use the modified version provided by Google [50]. It contains 341,873 and 40,832 training and test examples, respectively. Examples in FEMNIST are already grouped by writers. The number of examples each user has varies, and hence, the weight of each user is also set differently. Thus, for the experiments, we only change some parameters along with the structure of FEMNIST as follows: $n_{max} = 120$, and the weight of each user i by $w_i \leq 1$ accordingly. We also set $B = 16$, $E = 10$, and $\eta = 0.15$.

First, the privacy budget and the accuracy of the model are evaluated by varying the noise multiplier z to examine the effect of DP noise on FL. The number of users that participate in learning is set as 50, whereas the weight sum n is not fixed because the weight w_i differs with the user. In the experiments for evaluating the effect of only DP noise, as for MNIST, a case in which there is no user dropout is considered.

Fig. 4 shows the accuracy and ϵ for (ϵ, δ)-DP with $\delta = 10^{-5}$. When $z = 1.0$ and $z = 1.5$, the accuracies of image classification in 200 rounds are 0.8205 and 0.5217, and the privacy budgets ϵ are 1.7456 and 0.7421, respectively. The accuracy for the non-DP case considered as a baseline is 0.9707. Therefore, relatively the accuracy loss due to DP noise is much larger than that in the case of MNIST. This is attributed to the difference in the data distributions of i.i.d and non-i.i.d datasets. In general, non-i.i.d data are biased, and therefore, the update vector obtained from training on them has lower quality than that from training on i.i.d data [51]. Thus, even for the same size of the noise vector, the resultant accuracy degradation with non-i.i.d data is more than that with i.i.d data. Therefore, to achieve high accuracy, the noise vector size should be reduced. Reducing the noise vector size while maintaining the privacy protection level can be achieved by increasing n. This is in the same context of increasing the accuracy when the model is trained using non-i.i.d data without applying DP.

We also evaluate the privacy budget and the accuracy of the model to examine the effects of the user dropout and the noise calibration. To this end, only the dropout rates and the noise calibration size are varied while keeping the other parameters fixed. Fig. 5 presents the accuracy and ϵ for (ϵ, δ)-DP with $\delta = 10^{-5}$. The experimental results show that changes in the privacy and accuracy caused by noise calibration are similar to those in case of MNIST database.

In conclusion, we believe that our algorithm is also effective for FL on non-i.i.d data. However, owing to the bias of the data distribution, we observe that it is difficult to achieve high accuracy with the same parameter setting as in the i.i.d case. To achieve high accuracy, it is necessary to change the parameter settings, such as the privacy budget and the number of users.

VII. CONCLUSION

In this paper, we first observed a new privacy risk for FL, i.e., user dropouts of an FL network causing failure in achieving the desired level of privacy protection. We then propose a DP mechanism for application to FL to address such privacy degradation and enhance DP for FL. Our algorithm satisfies user-level DP guarantee and distributed DP-noise generation, to prevent the privacy leak from the local update vectors of the user or their aggregation. The idea behind the construction is to develop a noise calibration step to ensure DP-robustness towards user dropouts in terms of privacy. In addition, from the experiments, we showed that the proposed mechanism increases the level of privacy protection by 15% and 50% for 10% and 30% dropout cases, respectively, over the existing FL mechanisms with DP. Our algorithm provides stability for

Algorithm 3: DP-FL Robust to User Dropouts (Extended)

Parameters:
- noise multiplier z
- user example limit n_{max}
- clipping bound C
- user sampling probability q
- threshold for secret sharing k
- lower bound of the weight sum that algorithm works n_0

Main Algorithm:

Initial model θ^0, user set U

For each $i \in U$, i has n_i examples and $w_i = \min\{\frac{n_i}{n_{max}}, 1\}$

for *each round* $t = 0, 1, 2, \ldots$ **do**

User: *each user* $i \in U$ **do**

Generate DH keypairs (c_i^{SK}, c_i^{PK}) and (s_i^{SK}, s_i^{PK})

Send public keys (c_i^{PK}, s_i^{PK}) and weight w_i

Server:

$U^t \leftarrow$ sample user set with probability q

Check $n_0 \leq \sum_{i \in U^t} w_i$ and broadcast to $U^t : \{(c_i^{PK}, s_i^{PK}) \mid i \in U^t\}$ and $\sigma_0 \leftarrow \frac{2C}{n_0} z$

User: *each user* $i \in U^t$ **do**

Generate b_i and compute keys $c_{i,j} \leftarrow (c_j^{PK})^{c_i^{SK}}$ and $s_{i,j} \leftarrow (s_j^{PK})^{s_i^{SK}}$ for $j(\neq i) \in U^t$

Compute Shamir's $(k, |U^t|)$ secret shares for b_i and s_i^{SK}: $\{(j, b_{i,j}, s_{i,j}^{SK}) \mid j \in U^t\}$

Send encrypted shares $e_{i,j}$ of $(i, j, b_{i,j}, s_{i,j}^{SK})$ using shared key $c_{i,j}$ for $j(\neq i) \in U^t$

Server:

Send $\{e_{i,j} \mid i(\neq j) \in U^t\}$ to each user $j \in U^t$

User: *each user* $i \in U^t$ **do**

$\Delta_i^{t+1} \leftarrow$ **UserUpdate**(i, θ^t)

$e_i^{t+1} \leftarrow \mathcal{N}(0, n_0 w_i \sigma_0 0^2 \mathbf{I})$

Send $M_i \leftarrow w_i \Delta_i^{t+1} + e_i^{t+1} + H(b_i) + \sum_{j \in U^t, i<j} H(s_{i,j}) - \sum_{j \in U^t, i>j} H(s_{i,j})$

Server:

$U_A^t \leftarrow \{\text{alive users} \in U^t\}$

Broadcast to $U_A^t : n' \leftarrow \sum_{i \in U_A^t} w_i$, $\sigma' \leftarrow \frac{2C}{n'} z$

User: *each user* $i \in U_A^t$ **do**

Decrypt $e_{j,i}$'s for $j(\neq i) \in U^t$

Send $\{b_{j,i} \mid j \in U_A^t\}$ and $\{s_{j,i}^{SK} \mid j \in U^t \setminus U_A^t\}$

Send $e'^{t+1}_i \leftarrow -e_i^{t+1} + \mathcal{N}(0, n' w_i \sigma'^2 \mathbf{I})$

Server:

Reconstruct $\{b_j \mid j \in U_A^t\}$ and $\{s_j^{SK} \mid j \in U^t \setminus U_A^t\}$

Compute $\{s_{i,j} \mid i \in U_A^t, j \in U^t \setminus U_A^t\}$

Share $\theta^{t+1} \leftarrow \theta^t + \frac{1}{n'} \sum_{i \in U_A^t} (M_i + e'^{t+1}_i - H(b_i) - \sum_{j \in U^t \setminus U_A^t, i<j} H(s_{i,j}) + \sum_{j \in U^t \setminus U_A^t, i>j} H(s_{i,j}))$

privacy levels without large accuracy loss even in network where user dropouts occur frequently.

APPENDIX

DETAILED DESCRIPTION OF ALGORITHM

We provide the extended description of Algorithm 2. We apply the approach to force the server to add noise mentioned in the last paragraph of Section V. Furthermore, the SMC protocol proposed by Bonawitz *et al.* [9] is applied to the algorithm to explain how the algorithm actually works. Algorithm 3 outlines the extended version of DP-FL robust to user dropouts.

We provide a brief description of several terms about SMC to be used in the algorithm below.

- DH keypair: DH keypair is a key pair that consists of a secret key and a public key generated by Diffie-Hellman Key Agreement [52]. For a given generator g of a multiplicative group with a prime order p, a DH keypair is generated in the form of (x, g^x) where x is a secret key randomly sampled from $\mathbb{Z}_p$ and g^x is a public key. Then two users with DH key pairs (x, g^x) and (y, g^y) respectively, can agree the shared key $g^{xy} = (g^x)^y = (g^y)^x$.
- Shamir's secret share: SMC protocol proposed by Bonawitz *et al.* exploits Shamir's Secret Sharing [53] to remove the masking of the dropped user. Shamir's (k, n) Secret Sharing allows a user to split her secret into n shares, such that any k shares can reconstruct the secret, but $k - 1$ or less shares reveal no information about the secret. In the SMC protocol, k is a parameter of the protocol which is determined depending on the application. If more than k users are alive, the masking of the dropped users can be recovered and removed using Shamir's Secret Sharing.
- Pseudorandom generator H: SMC protocol proposed by Bonawitz *et al.* requires a secure pseudorandom generator H. It takes a seed of a fixed length as input and produces an output of the length of the update vector. The use of H can reduce the communication complexity in the protocol by sharing only a random seed instead of a random vector of the update vector length.

REFERENCES

[1] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *Proc. MLSys*, 2019.

[2] *Federated Learning: Collaborative Machine Learning Without Centralized Training Data*. Accessed: Oct. 29, 2021. [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[3] *Privacy That Works for Everyone*. Accessed: Oct. 29, 2021. [Online]. Available: https://blog.google/technology/safety-security/privacy-everyone-io/

[4] R. Prakash and B. V. Ryswyk, "Build trust through better privacy," in *Proc. WWDC 2020*, Jun. 2020.

[5] M. Xue and J. Freudiger, "Designing for privacy," in *Proc. WWDC 2019*, Jun. 2019.

[6] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *Proc. USENIX Secur. Symp.*, 2019, pp. 267–284.

[7] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.

[8] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. ICLR*, 2017.

[9] 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 *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1175–1191.

[10] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur. (AISec)*, 2019, pp. 1–11.

[11] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "HybridAlpha: An efficient approach for privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur. (AISec)*, 2019, pp. 13–23.

[12] M. Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu, "Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings," in *Proc. CRYPTO*, 2018, pp. 597–627.

[13] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 1310–1321.

[14] C. R. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv: 1712.07557*.

[15] N. Agarwal, A. T. Suresh, X. F. Yu, S. Kumar, and B. McMahan, "cpSGD: Communication-efficient and differentially-private distributed SGD," in *Proc. NIPS*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 7575–7586.

[16] O. Thakkar, G. Andrew, and H. B. McMahan, "Differentially private learning with adaptive clipping," 2019, *arXiv:1905.03871*.

[17] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. ICLR*, 2018.

[18] V. Mugunthan, A. Peraire-Bueno, and L. Kagal, "PrivacyFL: A simulator for privacy-preserving and secure federated learning," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 3085–3092.

[19] L. Wang, R. Jia, and D. Song, "D2P-Fed: Differentially private federated learning with efficient communication," 2020, *arXiv:2006.13039*.

[20] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.

[21] *MNIST Handwritten Digit Database*. [Online]. Available: https://yann.lecun.com/exdb/mnist/

[22] *FEMNIST Dataset*. [Online]. Available: https://github.com/TalwalkarLab/leaf/tree/master/data/femnist

[23] C. Dwork, "Differential privacy," in *Proc. ICALP*, 2006, pp. 1–12.

[24] C. Dwork, F. McSherry, K. Nissim, and D. Adam Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. TCC*, 2006, pp. 265–284.

[25] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.

[26] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *Proc. IEEE 55th Annu. Symp. Found. Comput. Sci.*, Oct. 2014, pp. 464–473.

[27] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *J. Mach. Learn. Res.*, vol. 12, pp. 1069–1109, Mar. 2011.

[28] R. Iyengar, J. P. Near, D. Song, O. Thakkar, A. Thakurta, and L. Wang, "Towards practical differentially private convex optimization," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 299–316.

[29] M. Abadi, A. Chu, I. Goodfellow, H. B. Mcmahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 308–318.

[30] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 332–349.

[31] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, "Scalable private learning with PATE," in *Proc. ICLR*, 2018.

[32] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, and J. Naughton, "Bolt-on differential privacy for scalable stochastic gradient descent-based analytics," in *Proc. ACM Int. Conf. Manage. Data*, May 2017, pp. 1307–1322.

[33] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, Q. S. T. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.

[34] X. Huang, Y. Ding, Z. L. Jiang, S. Qi, X. Wang, and Q. Liao, "DP-FL: A novel differentially private federated learning framework for the unbalanced data," *World Wide Web*, vol. 23, no. 4, pp. 2529–2545, Jul. 2020.

[35] Q. Zheng, S. Chen, Q. Long, and J. W. Su, "Federated f-differential privacy," in *Proc. AISTATS*, A. Banerjee and K. Fukumizu, Eds., vol. 130, 2021, pp. 2251–2259.

[36] J. Dong, A. Roth, and J. W. Su, "Gaussian differential privacy," 2019, *arXiv:1905.02383*.

[37] A. Bhowmick, C. J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," 2018, *arXiv:1812.00984*.

[38] J. Li, M. Khodak, S. Caldas, and A. Talwalkar, "Differentially private meta-learning," in *Proc. ICLR*, 2020, pp. 1–18.

[39] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 334–348.

[40] J. Yuan and S. Yu, "Privacy preserving back-propagation neural network learning made practical with cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 212–221, Jan. 2014.
[41] A. Brutzkus, R. Gilad-Bachrach, and O. Elisha, "Low latency privacy preserving inference," in *Proc. ICML*, vol. 97, 2019, pp. 812–821.
[42] V. Chen, V. Pastro, and M. Raykova, "Secure computation for machine learning with SPDZ," 2019, *arXiv:1901.00329*.
[43] P. Mohassel and P. Rindal, "ABY 3: A mixed protocol framework for machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 35–52.
[44] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. Asia Conf. Comput. Commun. Secur.*, May 2018, pp. 707–721.
[45] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "FALCON: Honest-majority maliciously secure framework for private deep learning," in *Proc. Priv. Enhancing Technol. (PoPETs)*, 2021, pp. 188–208.
[46] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *Proc. ICLR*, 2017, pp. 1–16.
[47] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org
[48] *Tutorial for MNIST in Tensorflow*. [Online]. Available: https://github.com/tensorflow/privacy/blob/master/tutorials/mnist_dpsgd_tutorial_keras.py
[49] *RDP Accountant in Tensorflow-Privacy*. [Online]. Available: https://github.com/tensorflow/privacy/tree/master/tensorflow_privacy/privacy
[50] *Libraries for FEMNIST*. Accessed: Oct. 29, 2021. [Online]. Available: https://github.com/tensorflow/federated/blob/main/tensorflow_federated/python/simulation/datasets/emnist.py
[51] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018, *arXiv:1806.00582*.
[52] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.
[53] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

CHUNGHUN BAEK received the Ph.D. degree in mathematics from Seoul National University (SNU), in 2016. He is currently a Senior Engineer with Samsung Research. His research interests include computational number theory, cryptography, information security, and data privacy.

SUNGWOOK KIM received the B.S. and joint M.S./Ph.D. degrees in mathematics from Seoul National University (SNU), South Korea, in 2005 and 2012, respectively. He is currently an Assistant Professor with the Department of Information Security, Seoul Women's University (SWU). Before joining SWU, in 2020, he was the Principal Engineer of Samsung Research, Samsung Electronics, Seoul. His research interests include cryptography, information security, and privacy-preserving technology.

DONGKYUN NAM is working as an Engineer with Samsung Research. His research interests include data privacy and distributed systems.

JIHOON PARK is currently working with the Security Team, Samsung Research, as a Staff Engineer. His research interests include privacy, blockchain, zero knowledge proof, and multiparty computation.