Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 31(7): 3741–3758.
DOI: 10.3934/era.2023190
Received: 04 February 2023
Revised: 13 April 2023
Accepted: 16 April 2023
Published: 28 April 2023

Research article

Hierarchical federated learning with global differential privacy

Youqun Long[1], **Jianhui Zhang**[2], **Gaoli Wang**[1,*] **and Jie Fu**[1]

[1] Software Engineering Institute, East China Normal University, Shanghai, China
[2] Shandong Luruan Digital Technology Co., Ltd., R & D Center, Jinan, China

* **Correspondence:** Email: glwang@sei.ecnu.edu.cn.

Abstract: Federated learning (FL) is a framework which is used in distributed machine learning to obtain an optimal model from clients' local updates. As an efficient design in model convergence and data communication, cloud-edge-client hierarchical federated learning (HFL) attracts more attention than the typical cloud-client architecture. However, the HFL still poses threats to clients' sensitive data by analyzing the upload and download parameters. In this paper, to address information leakage effectively, we propose a novel privacy-preserving scheme based on the concept of differential privacy (DP), adding Gaussian noises to the shared parameters when uploading them to edge and cloud servers and broadcasting them to clients. Our algorithm can obtain global differential privacy with adjustable noises in the architecture. We evaluate the performance on image classification tasks. In our experiment on the Modified National Institute of Standards and Technology (MNIST) dataset, we get 91% model accuracy. Compared to the previous two-layer HFL-DP, our design is more secure while as being accurate.

Keywords: differential privacy; federated learning; hierarchical architecture; privacy preservation; distributed network

1. Introduction

With the rapid development of mobile devices and Internet-of-Things (IoT), it is expected that everything will be connected closely [1, 2]. To apply artificial intelligence (AI) technology to different scenes, we prefer to choose distributed machine learning (ML) to process tasks in cooperation with each other [3]. Though practical and pragmatic as it is, it suffers many reliability and security risks in a real distributed environment [4, 5]. When training the models cooperatively, the privacy of clients' sensitive datasets becomes a great concern. Hence, federated learning, in which datasets are processed only on the client side, has been proposed [6, 7]. FL aims to fit a common model by an empirical risk minimization (ERM) objective. Nevertheless, FL still poses many worrying crises on information leakage, communication efficiency and device diversity [8–12].

As a commonly used architecture, client-server FL also suffers from stability problems, especially in poor network [13]. To address these problems, the more efficient and stable architecture client-edge-cloud hierarchical FL has been proposed [14]. The HFL allows multiple edge servers to perform partial model aggregation. In this way, the model can be trained faster, and better communication-computation trade-offs can be achieved. Edge servers are designed to aggregate the clients' model parameters. Hence, edge servers are capable of reducing the computing burden, message delay and communication burden of cloud servers [15].

FL usually adopts stochastic gradient descent (SGD) in the training process. Based on distributed SGD with a one-step local update, [16, 17] have analyzed the convergence performance of federated learning. Federated proximal (FedProx) [18], taking advantage of regularization on a local loss function, improves the convergence performance and system stability.

With FL becoming more and more attractive, data security and confidentiality also raise great concerns. FL seems to protect clients' individual data by exchanging parameters instead of sensitive information. However, there exist many malicious adversaries that are capable of stealing some secret information [19]. The most useful way to prevent information exposure is to apply differential privacy by adding noises in trained parameters [20]. As to the research about DP in FL, there are mainly three aspects: client-level, sample-level and LDP-FL [21–26].

Though there are various designs above, the global differential privacy on three-layer hierarchical federated learning is still unexplored. As to two-layer HFL-DP [27], the authors proposed a privacy-preserving scheme based on the theory of local differential privacy, with addition of noise to the shared model parameters before uploading them to edge and cloud servers. However, they did not control the amount of Gaussian noise in the architecture. So, if we want to expand our layer architecture, we may add so much noise to damage the utility, even leading to a tough convergence. Our design solves this problem.

To design a practical and efficient algorithm to reach the balance between privacy of the sensitive data and the quality of a trained model, in this paper, we propose our novel framework based on a DP mechanism. The key value of our design is to realize the global privacy budget in the whole architecture. At each time, we control the overall amount of noise we add instead of letting it increase blindly. Consequently, the model accuracy can be adjusted due to the trade-off between model accuracy and privacy cost. From our perspective, there are few papers about the research of global privacy in hierarchical federated learning. We conduct extensive tests based on real-world datasets to validate the performance of our algorithm. Evaluations show that theoretical results and test results are consistent. To summarize, we propose a novel scheme that satisfied the requirement of DP with global data under a certain noises perturbation level by adding proper Gaussian noise and exploring the trade-off between privacy parameters and model utility to comprehend the relation between privacy level and model quality. By doing so, sensitive user data can be protected even if there exist many adversaries from inner and outer cyberspace.

2. Related work

2.1. Client-level

In client-level, in order to protect the clients' information, we treat the clients' model as a piece of data. In [21] DP-FedAvg and DP-FedSGD were proposed. The sampling is to sample the client, and the noises are added at the center. Sensitivity is calculated based on the sampling rate, with average weights for each client. McMahan et al. suggest that with a huge number of users engaging in, realizing differential privacy costs more computation instead of decreasing utility. By their experiment, the LSTM language models can be similar to un-noised models from quantitative and qualitative aspects when using large datasets. They consider the effect of large step updates on the model. In [22] the model uploaded by the client is clipped at the center so that an adaptive clipping operation can be performed (such as taking the median value of the norm of each client model as the clipping norm). The key point is concealing the contribution of individual clients in the training process. It was shown that when the client's participation is hidden, the model obtained by federated learning still has a good performance. Then we can conclude that when enough clients are participating in the learning, the model can achieve high accuracy. In [23] the uplink privacy protection is carried out first, and then the downlink noises are added based on the uplink noises addition, and the convergence analysis is given. A theoretical convergence limit is specified for the loss function of the trained FL model. Based on this, we can know that the better the convergence performance is the lower the protection level. Vice, given a fixed privacy protection level, increasing the overall number N of clients participating in FL can improve the convergence performance of the final model; and under a given privacy protection level, there is an optimal maximum number of aggregations T in terms of convergence performance.

2.2. Sample-level

In sample-level, we protect the sensitive information of the samples under the client. Each sample under the client is regarded as a piece of data, and the federation center is regarded as an opponent. In [24] integrates the DPAGD-CNN algorithm into federated machine learning. DPAGD-CNN is an adaptive allocation algorithm for differential privacy budget under centralized machine learning. Considering the difference in user data, the model gets accurate when adjusting differential privacy. In [25] sampling on the client instead of the data in the client side, but also applies the privacy measurement method of MA and then adds a method based on the privacy budget to adapt the optimal number of iterations, which is actually a novel form of self-adaptation privacy budget allocation. Wei et al. made an exact relation between communication round T and privacy level.

2.3. LDP-FL

In LDP-FL, a client has only one sample datum, and if there are multiple sample data, multiple gradients will be trained and uploaded to the federation center. In [26] the CLDP-SGD algorithm was proposed, which first samples the clients, then samples again under the clients and finally shuffles the gradients of different clients. There are multiple sample gradients, and the clients will pass multiple gradients to the center. Noises are added locally based on coded perturbation.

2.4. HFL-DP

In federated learning, each participant can build the model without disclosing the underlying data and only shares the weight update and gradient information of the model with the server. In order to ensure the security of federated learning, noise is added to the model update to obscure the contribution of the client. In [28], they suggest a novel DP aggregation scheme to the improve update strategy.

Meanwhile, they adopt the f-differential privacy to analyze the privacy loss so that the loss will be calculated more accurately at every round. In [27], they suggest a design on local differential privacy. They analyze the privacy loss by the moment accounting, so their design can get a tight privacy guarantee. By adding the noise to the shared model parameters before uploading them to edge and cloud servers, they realize the strict differential privacy guarantee for the layers of clients and edge servers.

Table 1. Main notations.

Main notations	
k_1	Number of local updates for one edge aggregation
k_2	Number of edge aggregations for one cloud aggregation
T	Number of iterations
w^0	Initial global parameter
σ_U	Variance of the Gaussian noises for client
σ_E	Variance of the Gaussian noises for edge server
t	Index of iteration
w_i^t	Local parameters of client i in iteration t
C	Clipping bound
n	Number of clients
N	Number of edge servers
μ	Presetting constant of the proximal term

Table 2. Acronyms.

Acronyms	
federated learning	FL
hierarchy federated learning	HFL
differential privacy	DP
Local Differential Privacy	LDP
Internet-of-Things	IoT
artificial intelligence	AI
machine learning	ML
empirical risk minimization	ERM
Federated Proximal	FedProx
stochastic gradient descent	SGD
convolutional neural network	CNN
Linear rectification function	ReLu
Independent Identically Distribution	IID

3. Preliminaries

In this section, we present the HFL framework and the thread model on which our paper based and the related knowledge about DP.

3.1. Hierarchical federated learning

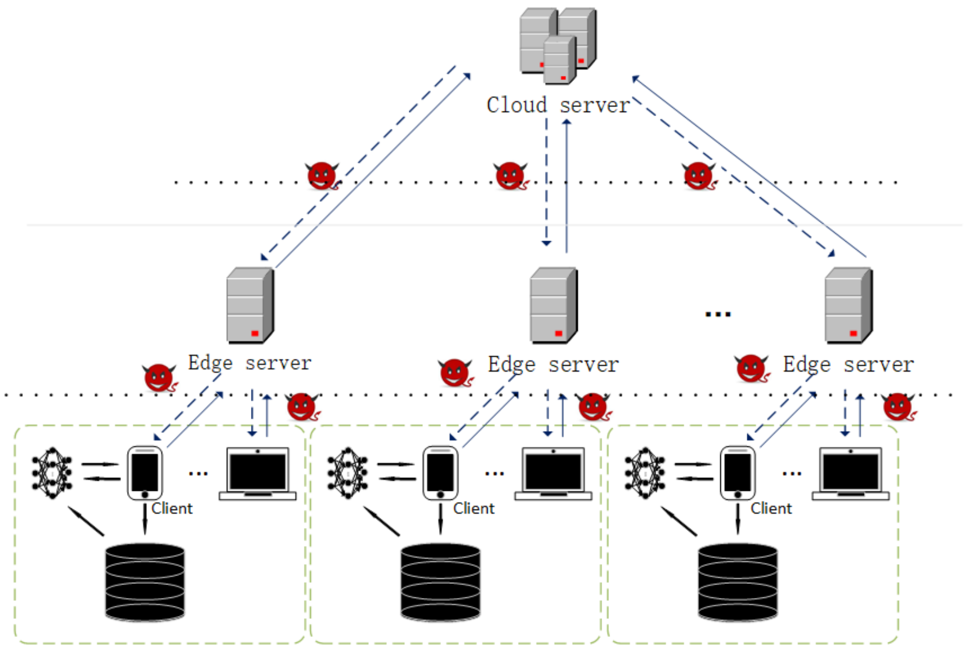

Figure 1. Hierarchical federated learning architecture.

Let us consider a general HFL system consisting of three different parts in three logistic layers, as shown in Figure 1. The clients connected to edge servers are various computers and mobile devices. They are to train a local model from their individual data, and then send local updates to edge server. Edge servers and cloud servers are responsible to aggregate and transmit back the aggregation to corresponding clients.

3.2. Threat model

To further improve security in our system, we consider clients, edge servers, and the cloud server to be all semi-honest, which means they will strictly follow the HFL framework but may infer privacy. Also, there may be external attackers trying to usurp clients' privacy. They are capable of inverting the broadcast models from the servers in order to seize the private individual data, as explained in [29].

Factually, download channels suffer more security risks than upload channels. That is because clients upload their model updates through direct and ephemeral upload channels while download channels are broadcasting and lasting. We consider at most t_1 exposures of uploaded updates from each client to their edge server. Due to the broadcast effect, we assume that aggregated parameters will be exposed in the downlink channel for every communication. Therefore, we assume t_2 exposures from each edge server to clients. Moreover, the value of t_2 satisfies the condition that $t|k_1 \wedge t \nmid k_1k_2$. Similarly, we assume there are t_4 exposures from each edge server to the cloud server and t_5 exposures from the cloud server to edge servers. The value of t_5 also satisfies the condition that $t \mid k_1k_2$. To be more specific, we assume that there are t_3 exposures in t_1 for cloud aggregation, and $t_1 - t_3$ exposures are counted as t_1 only for edge aggregation.

3.3. Differential privacy

In our hierarchical architecture, we apply a DP mechanism with parameters ϵ and δ to provide strong privacy preservation. $\epsilon > 0$ represents the distinguishable bound of all outputs on neighboring datasets in a database, and δ is the probability of the event that the ratio of the probabilities for two adjacent datasets cannot be bounded by e^{ϵ} after the mechanism. Therefore, a larger ϵ entails a higher risk of privacy leakage for a clearer distinguishability of two neighboring datasets [30]. Now, we will formally define DP as follows.

Definition 1. *((ϵ,δ)-Differential privacy): A randomized mechanism $\mathcal{M}$ satisfies (ϵ,δ)-DP if for $\mathcal{X} \to \mathcal{R}$ with domain $\mathcal{X}$ and range $\mathcal{R}$, and all result set $\mathcal{S} \subseteq \mathcal{R}$, there can be any neighboring datasets $\mathcal{D}_i$, $\mathcal{D}'_i \in \mathcal{X}$ which only differ by one sample, and then we have*

$$Pr[\mathcal{M}(\mathcal{D}_i) \in \mathcal{S}] \leq e^{\epsilon} Pr[\mathcal{M}(\mathcal{D}'_i) \in \mathcal{S}] + \delta \tag{3.1}$$

In this paper, we use a Gaussian mechanism which adopts L_2 norm sensitivity. To protect the function output $s(x)$, it needs to add zero-mean Gaussian noises with variance $\sigma^2\mathbf{I}$ in each coordinate.

$$M(x) = s(x) + \mathcal{N}(0, \sigma^2\mathbf{I}), \tag{3.2}$$

where $\mathbf{I}$ is an identity matrix, and $s()$ is a real-valued function.

To achieve (ϵ,δ)-DP, the noise scale $\sigma \geq c\Delta s/\epsilon$, and the constant $c \geq \sqrt{2ln(1.25/\delta)}$ for $\epsilon \in (0, 1)$. Δs is the sensitivity of function $s(x)$, which can be computed easily by $\Delta s = max_{\mathcal{D}_i,\mathcal{D}'_i}\|s(\mathcal{D}_i) - s(\mathcal{D}'_i)\|$. Seeing the definitions above, we ought to choose a proper value of σ to satisfy (ϵ, δ)-DP.

4. Hierarchical federated learning with differential privacy

4.1. Global differential privacy

To protect the parameter in upload and download channels, we design a global (ϵ, δ)-DP. First, we elaborate our design from the clients' upload parameters to the edge server. In every iteration, we consider the batch size in a local training process to be equal to the number of training data, and then we can give the definition of the i-th client in a local training process

$$\begin{aligned} s_U^{\mathcal{D}_i} &= \arg\min_w F_i(w, \mathcal{D}_i) \\ &= \frac{1}{|\mathcal{D}_i|} \sum_{j=1}^{|\mathcal{D}_i|} \arg\min_w F_i(w, D_{i,j}), \end{aligned} \tag{4.1}$$

where $\mathcal{D}_i$ is the dataset of i-th clients, and $\mathcal{D}_{i,j}$ is the j-th sample in $\mathcal{D}_i$. Hence, we can define the sensitivity of $s_U^{\mathcal{D}_i}$

$$\begin{aligned}\Delta s_U^{\mathcal{D}_i} &= \max_{\mathcal{D}_i,\mathcal{D}'_i} \| s_U^{\mathcal{D}_i} - s_U^{\mathcal{D}'_i} \| \\ &= \max_{\mathcal{D}_i,\mathcal{D}'_i} \| \frac{1}{|\mathcal{D}_i|} \sum_{j=1}^{|\mathcal{D}_i|} \arg\min_w F_i(w, D_{i,j}) \\ &\quad - \frac{1}{|\mathcal{D}'_i|} \sum_{j=1}^{|\mathcal{D}'_i|} \arg\min_w F_i(w, D'_{i,j}) \| \\ &= \frac{2C}{|\mathcal{D}_i|},\end{aligned} \tag{4.2}$$

where C is a clipping bound for w_i, and $\mathcal{D}_i$ and $\mathcal{D}'_i$ are neighboring datasets. Generally, we define the global sensitivity in the upload channel as

$$\Delta s_U = max\{\Delta s_U^{\mathcal{D}_i}\} = \max\{\frac{2C}{|\mathcal{D}_i|}\}, \ \ \forall i. \tag{4.3}$$

When clients take advantage of all the training datasets, it comes out to be a small global sensitivity. Hence, we set the number of local datasets $\mathcal{D}_i$ to be minimum by m. Due to exposure in the upload channel, we define the Gaussian noises $\sigma_U = c\Delta s_U/\epsilon$. After t_1 and t_3 exposures, the noises increase to $\sigma_{U_1} = ct_1\Delta s_U/\epsilon$ and $\sigma_{U_2} = ct_3\Delta s_U/\epsilon$ due to the linear relation of ϵ and σ_U. Similarly, $\sigma_E = ct_4\Delta s_E/\epsilon$ in the process from edge server to cloud server.

Then, we consider edge servers that upload parameters to the cloud server. The aggregation weight can be expressed as

$$w = p_1 w_1 + p_2 w_2 + ... + p_N w_N, \tag{4.4}$$

where n means the number of edge servers connecting, and w represents the result of aggregation. Thus, the sensitivity of edge server's aggregation $\Delta s_E^{\mathcal{D}_i}$ can be expressed as

$$\Delta s_E^{\mathcal{D}_i} = \max_{\mathcal{D}_i,\mathcal{D}'_i} \| s_E^{\mathcal{D}_i} - s_E^{\mathcal{D}'_i} \|. \tag{4.5}$$

We have

$$s_E^{\mathcal{D}_i} = p_1 w_1(\mathcal{D}_1) + ... + p_i w_i(\mathcal{D}_i) + ... + p_n w_n(\mathcal{D}_n) \tag{4.6}$$

and

$$s_E^{\mathcal{D}'_i} = p_1 w_1(\mathcal{D}_1) + ... + p_i w_i(\mathcal{D}'_i) + ... + p_n w_n(\mathcal{D}_n). \tag{4.7}$$

Therefore, the sensitivity can be expressed as

$$\begin{aligned}\Delta s_E^{\mathcal{D}_i} &= \max_{\mathcal{D}_i,\mathcal{D}'_i} \| p_i w_i(\mathcal{D}_i) - p_i w_i(\mathcal{D}'_i) \| \\ &= p_i \max_{\mathcal{D}_i,\mathcal{D}'_i} \| w_i(\mathcal{D}_i) - w_i(\mathcal{D}'_i) \| \\ &= p_i \Delta s_U^{\mathcal{D}_i} \leq \frac{2Cp_i}{m}\end{aligned} \tag{4.8}$$

To meet the demand of small global sensitivity in the channel, the sensitivity Δs_E can be given as

$$\Delta s_E = max\{\Delta s_E^{\mathcal{D}_i}\} = \max\{\frac{2Cp_i}{m}\}, \ \ \forall i, \tag{4.9}$$

where m is the size of datasets in a client. To reach the proper sensitivity Δs_E, we set all $p_i = \frac{1}{n}$. Similarly, we can define the sensitivity in cloud server as $\Delta s_C = \max\{\frac{2Cp_iq_i}{m}\}$, where p_i is the ratio of selecting client and q_i is the ratio of selecting edge server. We ought to add enough Gaussian noises in upload channel and then consider whether or not to add noises in download channel to satisfy the global (ϵ, δ)-DP.

4.2. HFL-DP

In our hierarchical system, we realize a secure and efficient model by global (ϵ, δ)-DP in Algorithm 1. We define the initial global parameter $w^{(0)}$ and introduce the proximal term μ to improve the stability of the overall framework. The essence of this correction term is to increase the difference between the parameters in the local model and the parameters in the global model.

Theorem 1. (DP guarantee for downlink channels): To ensure (ϵ,δ)-DP in the downlink channels with aggregations, the standard deviation of Gaussian noises n_E and n_C can be defined as

$$n_E = \begin{cases} \frac{2c*C}{\epsilon_1 mn}\sqrt{t_2^2 - t_1^2 n} & ,t_2 \geq \sqrt{n}t_1, \\ 0 & ,t_2 < \sqrt{n}t_1. \end{cases} \tag{4.10}$$

$$n_C = \begin{cases} \frac{2c*C}{\epsilon_2 mnN}\sqrt{t_5^2 - t_4^2 N - t_3^2 Nn} & ,t_5 \geq \sqrt{t_4^2 N + t_3^2 Nn}, \\ 0 & ,t_5 < \sqrt{t_4^2 N + t_3^2 Nn}. \end{cases} \tag{4.11}$$

Proof. First, we start from proof of the value of n_E. According to the definition of global (ϵ, δ)-DP in the uplink channels, the additive noises of clients can be given as $\sigma_U = c * t_1 \Delta s_U/\epsilon_1$, for the linear relation between ϵ_1 and σ_U with Gaussian mechanism. Here, we assume the noises added in clients obey the same distribution $n \sim \phi(n)$ due to the property of global (ϵ, δ)-DP. Then, the aggregation in an edge server with additive noises can be given as

$$\tilde{w} = \sum_{i=1}^{n} p_i(w_i + n_i) = \sum_{i=1}^{n} p_i w_i + \sum_{i=1}^{n} p_i n_i \tag{4.12}$$

Hence, the distribution of $\sum\limits_{i=1}^{n} p_i n_i$ can be given as

$$\Phi(n) = \bigotimes_{i=1}^{n} \phi_i(n), \tag{4.13}$$

where $\bigotimes$ is a convolutional operation, and $p_i n_i \sim \phi_i(n)$. If we set n_i with clients' noise scale σ_U by Gaussian mechanism, then $\sum\limits_{i=1}^{n} p_i n_i$ is a Gaussian distribution. According to the definition of Δs_E, $p_i = \frac{1}{n}$ will lead to a small sensitivity. To make it a global (ϵ, δ)-DP in download channels, we assume the value of noise scale to be $\sigma_A = ct_2 \Delta s_E/\epsilon_1$. Therefore, the additive noises at edge server can be expressed as

$$n_E = \sqrt{\sigma_A^2 - \frac{\sigma_U^2}{n}} = \begin{cases} \frac{2c*C}{\epsilon_1 mn}\sqrt{t_2^2 - t_1^2 n} & ,t_2 \geq \sqrt{n}t_1, \\ 0 & ,t_2 < \sqrt{n}t_1. \end{cases} \tag{4.14}$$

Hence, the value of n_E has been proved.

Then, we consider the value of n_C. In the uplink channels, we set the standard deviation of additive noise in clients and edge servers to be $\sigma_U = ct_3\Delta s_U/\epsilon_2$ and $\sigma_E = ct_4\Delta s_E/\epsilon_2$. To ensure the global ($\epsilon$, δ)-DP, the noises vectors in each client obey the same distribution $n \sim \phi(n)$, and the noises in each edge server obey the same distribution $N \sim \phi(N)$. We can get the aggregation at cloud server with artificial noises from clients and edge server as

$$\begin{aligned} \tilde{w} &= \sum_{j=1}^{N}\sum_{i=1}^{n} q_j p_i (w_i + n_i) + \sum_{j=1}^{N} q_j N_j \\ &= \sum_{j=1}^{N}\sum_{i=1}^{n} q_j p_i w_i + \sum_{j=1}^{N}\sum_{i=1}^{n} q_j p_i n_i + \sum_{j=1}^{N} q_j N_j \end{aligned} \tag{4.15}$$

We know the distributions of $q_j p_i n_i$ and $q_j N_j$ are Gaussian distributions. Moreover, we set the standard deviation of noises $\sigma_A = ct_5\Delta s_C/\epsilon_2$.Therefore, the additive noises at cloud server can be expressed as

$$n_C = \sqrt{\sigma_A^2 - \frac{\sigma_U^2}{n*N} - \frac{\sigma_E^2}{N}} = \begin{cases} \frac{2c*C}{\epsilon_2 mnN}\sqrt{t_5^2 - t_4^2 N - t_3^2 Nn} & ,t_5 \geq \sqrt{t_4^2 N + t_3^2 Nn}, \\ 0 & ,t_5 < \sqrt{t_4^2 N + t_3^2 Nn}. \end{cases} \tag{4.16}$$

Hence, the value of n_C has been proved.

At the beginning of this algorithm, the edge servers and cloud server broadcast their own privacy guarantees, and the cloud server sends the initial global parameter to all clients. In every local update, the client trains the model parameters by using its own local datasets. After the local training of each client, the local parameter will be clipped by the threshold C, and then Gaussian noise are added to the parameter w_i of each client i. The amount of noises depends on the parameter σ_U, which is calculated above. Upon receiving all its clients' local parameters, the edge server calculates the aggregated model parameter by weighted average.

After aggregating the model parameters, the edge server chooses to broadcast the aggregation to the selected client or to upload it towards the cloud server. In the case of the aggregation, the aggregated parameters need to be broadcast to clients, and it should add the noises n_E to aggregation according to Theorem 1. In another case, the aggregation parameter which will be sent to the cloud server in edge also needs protection by adding noises according to the value of σ_E.

The cloud server aggregates the model received from all edge servers if iteration t meets the pre-set condition. After the aggregation in the cloud server, the model parameters will have noises added according to the value of n_C. Here, we need to pay attention to the privacy protection performance of our algorithm. First, any malicious adversary cannot get reach to individual training datasets. Due to the local perturbations, it is very tough to infer any valid information from the uploaded parameters. Similarly, an adversary may reveal sensitive information from parameters in a download channel. Hence, when broadcasting the aggregated model parameters to clients, additive noises may be added to parameters based on the theorem above.

Algorithm 1 HFL-DP

```
1:  for $t = 1, 2, 3, ..., T$ do
2:      // Client:
3:      for $Client_i, i = 1, 2, ..., n$ do
4:          if $t|k1 = 0$ then
5:              Update the local parameters $w_i^t$ as
6:              $w_i^t = \arg\min_{w_i}(F_i(w_i) + \frac{\mu}{2}\|w_i - w^{t-1}\|^2)$
7:              Clip the local parameters
8:              $w_i^t = w_i^t / \max(1, \frac{\|w_i^t\|}{C})$
9:              Add noises and upload parameters
10:             $\tilde{w}_i^t = w_i^t + \sigma_U$
11:         else
12:             Update the local parameters $w_i^t$ as
13:             $w_i^t = \arg\min_{w_i}(F_i(w_i) + \frac{\mu}{2}\|w_i - w^{t-1}\|^2)$
14:         end if
15:     end for
16:     // Edge Server:
17:     if $t|k_1 = 0$ then
18:         for $Edge\ server_l, l = 1, 2, ..., N$ do
19:             Conduct Edge Aggregation
20:             $w^l \leftarrow \frac{\sum_{i \in n} |D_i| \tilde{w}_i^t}{|D^l|}$;
21:             if $t|k_1 k_2 = 0$ then
22:                 Add noises and upload parameters
23:                 $\tilde{w}^l = w^l + \sigma_E$
24:             else
25:                 for $Client_i, i \in n$ do
26:                     $w_i^t \leftarrow w^l + n_D$
27:                 end for
28:             end if
29:         end for
30:     end if
31:     // Cloud Server:
32:     if $t|k_1 k_2 = 0$ then
33:         Conduct Cloud Aggregation
34:         $w \leftarrow \frac{\sum_{i \in N} |D_i^l| \tilde{w}^l}{|D|}$;
35:         for $Client_i, i \in n$ do
36:             $w_i^t \leftarrow w + n_D$
37:         end for
38:     end if
39: end for
```

5. Evaluation

In this part, we evaluate the proposed HFL-DP in Python 3.7 and PyTorch 1.12.1 with real-world datasets. To analyze the performance of our algorithm, we conduct the experiment by varying the protection levels ϵ, the number of clients n, the number of Edge servers N, the value of clipping bound C.

We train our convolutional neural network (CNN) on the standard MNIST dataset for handwritten digit classification datasets and the CIFAR-10 dataset for ten class images with Python. MNIST dataset is composed of 60,000 training examples and 10,000 testing examples. Each example is a 28×28 grey-level image of handwritten digits. The CIFAR-10 dataset is composed of 32×32 color images in ten different classes. There are 60,000 examples for each class, with 40,000 examples for training, 10,000 for testing and 10,000 for validation.

In our neural network, we use two convolutional layers with ReLu units and softmax. We also use the cross-entropy function as a criterion. We assume the data we use are IID. In our experiment, the relevant parameters of the same layer are consistent, and we assume the communication channels among different layers are unimpeded.

5.1. *Performance loss with protection*

To show the influence of privacy preservation, we compare the proposed algorithm with non-private HierFAVG [14] on MNIST. We set $n = 50$, $N = 5$, $C = 15$, $\epsilon_1 = 20$ and $\epsilon_2 = 25$. Figures 2 and 3 indicate that privacy preservation costs a bit of accuracy. We can get that the non privacy-preserving trained model is better than the privacy-preserving trained model due to the unfavorable influence of Gaussian noise. Also, HFL converges faster than HFL-DP. However, the loss of accuracy is decreasing with the iteration round proceeding on. Though there may be 5 and 11% losses in the final model, it costs not much.

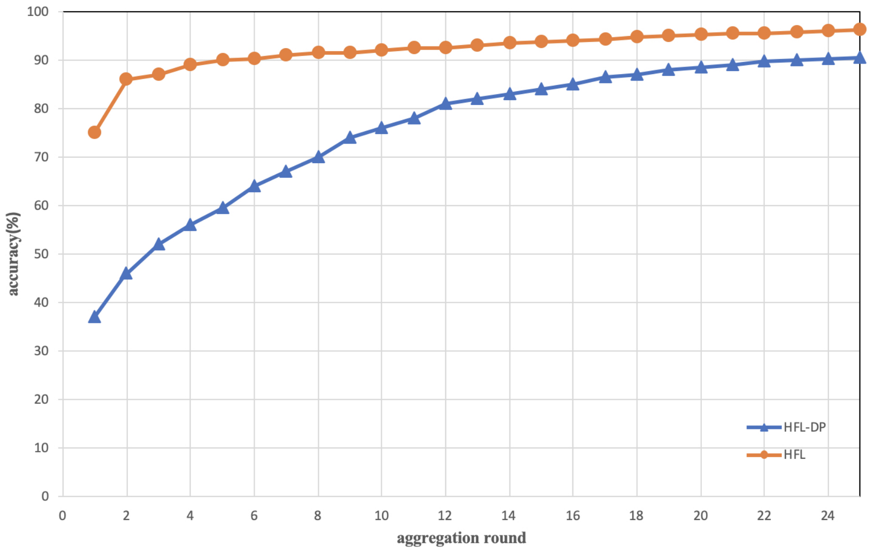

Figure 2. Accuracy comparison on MNIST.

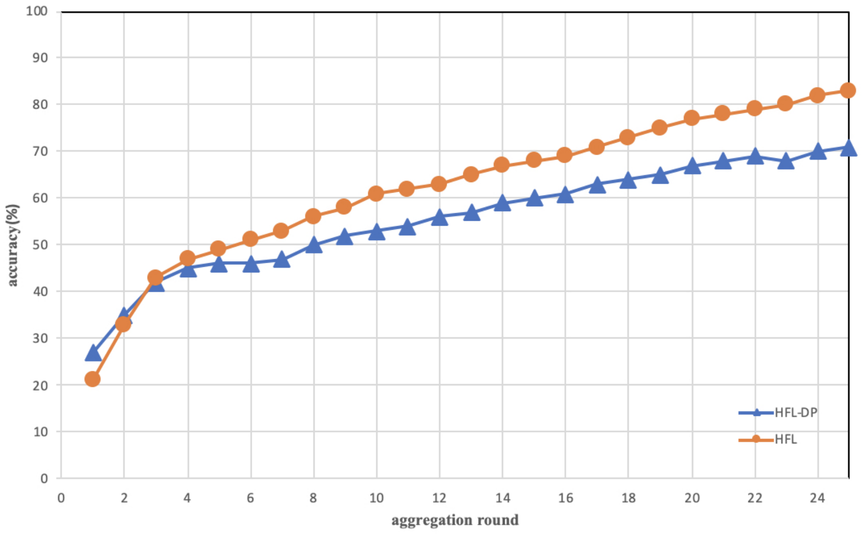

Figure 3. Accuracy comparison on CIFAR.

5.2. Performance evaluation on protection levels

In Figure 4, we set different values of protection levels $\epsilon = 10$, $\epsilon = 20$ and $\epsilon = 50$, both for ϵ_1 and ϵ_2. In our experiment, we choose $n = 50$, $N = 5$, $T = 50$, $k_1 = 2$ and $k_2 = 2$ and then we can get model accuracy of the aggregation rounds. As shown in Figure 4, the accuracy of the model increases if we relax the privacy guarantees. That is because ϵ represents the distinguishable bound of all outputs on neighboring datasets in a database. A larger ϵ entails a higher risk of privacy leakage for a clearer distinguishability of two neighboring datasets. However, as we continue to increase the value of ϵ, the accuracy does not increase much. So, a better choice of ϵ will balance the safety and utility. For example, when the value of ϵ gets bigger than 20, it will not entail a better model accuracy. We can set 20 as a preferable value in this case.

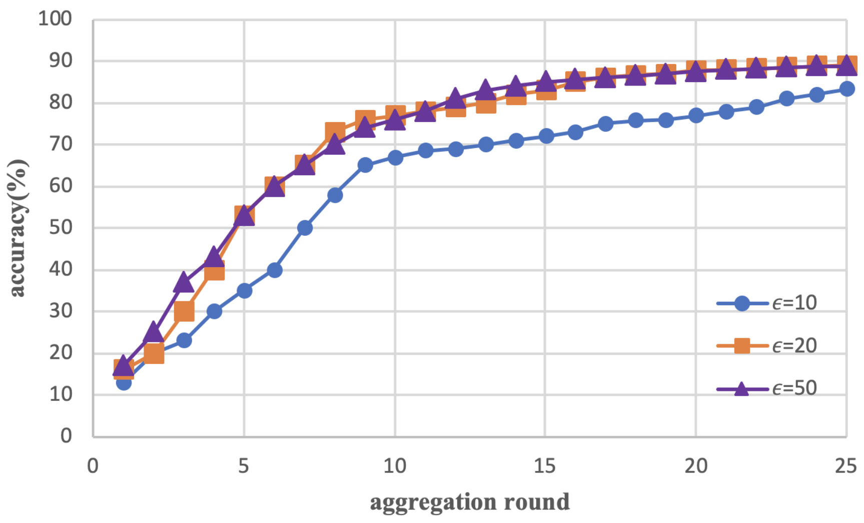

Figure 4. Different privacy preservation levels.

5.3. Performance evaluation on number of clients and edge servers

In this experiment, we set the protection parameter $\epsilon = 20$ and four groups of different numbers of clients and servers. We choose n = 50, N = 5, T = 50, $k_1 = 2$ and $k_2 = 2$. Figure 5 shows that when the scale of communication system becomes larger, it needs more noise to realize security. So, a larger communication system may entail more performance loss. In our algorithm, we try to control the amount of noise by Eqs (4.10) and (4.11) instead of adding Gaussian noise blindly. So, it may not decrease the accuracy greatly. For example, when $n = 100$ and $N = 20$, it gets the result of 82%. Under the same condition, when $n = 10$ and $N = 5$, it gets the result of 91%. Moreover, we can also know that a more practical model can be trained by fewer participants while preserving sensitive information. A safe system that contains many users remains to be researched.

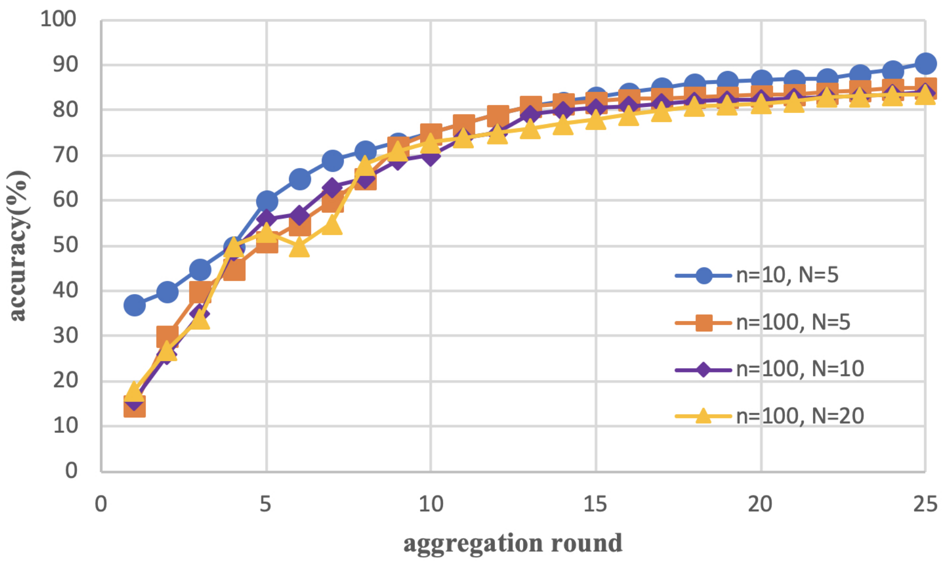

Figure 5. Number of clients and edge servers.

5.4. Performance evaluation on clipping bound

We choose various clipping bounds $C = 5$, $C = 10$ and $C = 15$. In our experiment, we set $\epsilon = 20$, $n = 50$ and $N = 5$. Figure 6 indicates that once the value of the clipping bound is large enough, the model accuracy will not increase anymore. First, the gradient is clipped, so that the gradient of all samples is less than clipping bound C. Then, a sufficiently large Gaussian noise is added to the clipped gradient to achieve differential privacy. The clipping bound C is equivalent to the concept of sensitivity mentioned before, i.e., the maximum impact that each sample can cause will not exceed C. When a large enough noise positively correlated with C is added to the gradient, the sample can be hidden to meet the requirements of differential privacy.

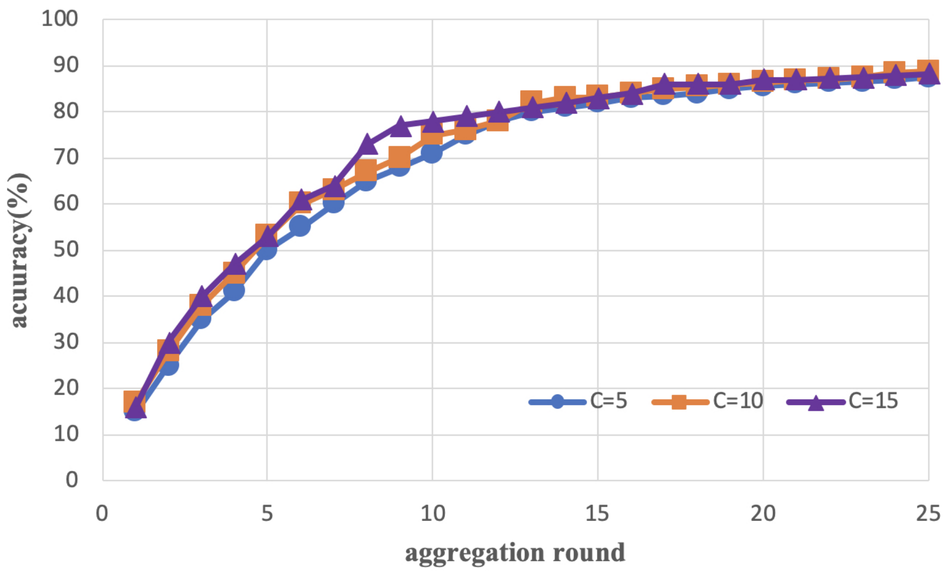

Figure 6. Clipping bound.

6. Result comparison

To further summarize the results of the existing methodological analysis of HFL-DP, the results of the analysis in this paper were compared with the existing literature. The results of the analysis of the different HFL-DP schemes are shown in the Table 3. In [28], the authors researched differential privacy on Server-Client architecture. They used f-DP for training tracking in order to get a tighter definition of privacy. This can make it more communication rounds than using a moment accountant. Meanwhile, they implemented a trick on the clipping method to let the model curve better. In their experiment, the final model accuracy of 25 rounds was 91%. In [27], the authors calculated the noises to realize (ϵ_i^l,δ_i^l)-DP for client and (ϵ^l,δ^l)-DP for edge. The final model accuracy was 90%. This idea effectively realized privacy leakage from the upload aspect.

In [28], the research is designed for Server-Client architecture from client level. The authors do not consider the three-layer HFL yet. In [27], though it gets a better result seemingly, it ignores the privacy leakage in the download channel. If we add directly adequate noises in download channel as the author mentions in [27], it will entail a negative effect of noise accumulation in the hierarchical system on the model utility. Because of the hierarchical structure, we may add noises on the edge side and cloud side in the download channel. Although the amount of noise on each side is different, it still makes a bad influence after several transmissions. Therefore, our design aims to control the amount of noise from the whole system as articulated in Algorithm 1. The additive noises in every download channel can be calculated by (4.13) and (4.15).

In summary, a better model utility can be obtained in a three-layer HFL-DP by studying the overall privacy loss and then controlling the amount of noise added each time.

Table 3. Comparison among different HFL-DP.

Client	Edge	ϵ	ACC	Reference
100	-	-	82%	paper [28]
100	10	10	90%	paper [27]
100	10	10	82%	this paper

7. Conclusions

In this paper, we proposed a novel privacy-preserving method with the concept of global differential privacy. Our scheme provides confidentiality for clients' sensitive data to avoid security threats from not only participants in the communication system but also adversaries from outer cyberspace. We realize the communication protection by adding noises to shared model parameters in each channel. So, user data cannot be directly inferred by any adversary while still keeping high utility for the training model. In our experiment on the MNIST dataset, we get 91% model accuracy. Compared to previous two-layer HFL-DP [27], our design is more secure while being as accurate. Then, we evaluated the performance based on the image classification. The experiment results show that our method is practical and efficient with proper parameters. Therefore, we can balance the trade-off between efficiency and safety by adjusting relevant parameters. In the future, we will do further research about adaptive adjustment to realize the preferable privacy budget and the non-IId data's effect on the utility of our algorithm. Meanwhile, we will tune our algorithm to adapt to the changes in user data distribution.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (2020YFA0712300), National Natural Science Foundation of China (62132005), Shanghai Trusted Industry Internet Software Collaborative Innovation Center.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. C. Wang, X. Wu, G. Liu, T. Deng, K. Peng, S. Wan, Safeguarding cross-silo federated learning with local differential privacy, *Digital Commun. Networks*, **8** (2022), 446–454. https://doi.org/10.1016/j.dcan.2021.11.006
2. J. Shi, P. Cong, L. Zhao, X. Wang, S. Wan, M. Guizani, A two-stage strategy for UAV-enabled wireless power transfer in unknown environments, *IEEE Trans. Mob. Comput.*, **2023** (2023), 1–15. https://doi.org/10.1109/TMC.2023.3240763

3. Q. Liu, Z. Zeng, Y. Jin, Distributed machine learning, optimization and applications, *Neurocomputing*, **489** (2022), 486–487. https://doi.org/10.1016/j.neucom.2021.12.058

4. M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, S. Camtepe, Privacy preserving distributed machine learning with federated learning, *Comput. Commun.*, **171** (2021), 112–125. https://doi.org/10.1016/j.comcom.2021.02.014

5. M. Sun, R. Yang, L. Hu, A secure distributed machine learning protocol against static semi-honest adversaries, *Appl. Soft Comput.*, **102** (2021), 107095. https://doi.org/10.1016/j.asoc.2021.107095

6. J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, et al., From distributed machine learning to federated learning: a survey, *Knowl. Inf. Syst.*, **64** (2022), 885–917. https://doi.org/10.1007/s10115-022-01664-x

7. C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, *Knowledge-Based Syst.*, **216** (2021), 106775. https://doi.org/10.1016/j.knosys.2021.106775

8. X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, M. Chen, In-Edge AI: intelligentizing mobile edge computing, caching and communication by federated learning, *IEEE Network*, **33** (2019), 156–165. https://doi.org/10.1109/MNET.2019.1800286

9. T. Li, A. K. Sahu, A. Talwalkar, V. Smith, Federated learning: challenges, methods, and future directions, preprint, arXiv:1908.07873.

10. H. Yang, Z. Liu, T. Q. S. Quek, H. V. Poor, Scheduling policies for federated learning in wireless networks, *IEEE Trans. Commun.*, **68** (2019), 317–333. https://doi.org/10.1109/TCOMM.2019.2944169

11. M. Hao, H. Li, G. Xu, S. Liu, H. Yang, Towards efficient and privacy-preserving federated deep learning, in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, Paris, France, 2019. https://doi.org/10.1109/ICC.2019.8761267

12. J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, M. Guizani, Reliable federated learning for mobile networks, *IEEE Wireless Commun.*, **27** (2020), 72–80. https://doi.org/10.1109/MWC.001.1900119

13. S. Liu, J. Yu, X. Deng, S. Wan, FedCPF: an efficient-communication federated learning approach for vehicular edge computing in 6G communication networks, *IEEE Trans. Intell. Transp. Syst.*, **23** (2022), 1616–1629. https://doi.org/10.1109/TITS.2021.3099368

14. L. Liu, J. Zhang, S. H. Song, K. B. Letaief, Client-Edge-Cloud hierarchical federated learning, in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, (2020), 1–6. https://doi.org/10.1109/ICC40277.2020.9148862

15. S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, et al., Adaptive federated learning in resource constrained edge computing systems, *IEEE J. Sel. Areas Commun.*, **37** (2019), 1205–1221. https://doi.org/10.1109/JSAC.2019.2904348

16. A. Agarwal, J. C. Duchi, Distributed delayed stochastic optimization, in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 2012. https://doi.org/10.1109/CDC.2012.6426626

17. X. Lian, Y. Huang, Y. Li, J. Liu, Asynchronous parallel stochastic gradient for nonconvex optimization, *ACM NIPS*, (2015), 2737–2745. Available from: https://proceedings.neurips.cc/paper/2015/hash/452bf208bf901322968557227b8f6efe-Abstract.html.

18. T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, preprint, arXiv:1812.06127.

19. A. Wainakh, A. S. Guinea, T. Grube, M. Mühlhäuser, Enhancing privacy via hierarchical federated learning, in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, (2020), 344–347. https://doi.org/10.1109/EuroSPW51379.2020.00053

20. M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, et al., Deep learning with differential privacy, in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, (2016), 308–318. https://doi.org/10.1145/2976749.2978318

21. H. B. McMahan, D. Ramage, K. Talwar, L. Zhang, Learning differentially private recurrent language models, in *ICLR 2018*, (2018), 1–14. Available from: https://openreview.net/forum?id=BJ0hF1Z0b.

22. R. C. Geyer, T. Klein, M. Nabi, Differentially private federated learning: a client level perspective, preprint, arXiv:1712.07557.

23. K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, et al., Federated learning with differential privacy: algorithms and performance analysis, *IEEE Trans. Inf. Forensics Secur.*, **15** (2020), 3454–3469. https://doi.org/10.1109/TIFS.2020.2988575

24. X. Huang, Y. Ding, Z. L. Jiang, S. Qi, X. Wang, Q. Liao, DP-FL: a novel differentially private federated learning framework for the unbalanced data, *World Wide Web*, **23** (2020), 2529–2545. https://doi.org/10.1007/s11280-020-00780-4

25. K. Wei, J. Li, M. Ding, C. Ma, H. Su, B. Zhang, et al., User-level privacy-preserving federated learning: analysis and performance optimization, *IEEE Trans. Mob. Comput.*, **21** (2022), 3388–3401. https://doi.org/10.1109/TMC.2021.3056991

26. A. Girgis, D. Data, S. Diggavi, P. Kairouz, A. T. Suresh, Shuffled model of differential privacy in federated learning, in *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, **130** (2021), 2521–2529. Available from: http://proceedings.mlr.press/v130/girgis21a.html.

27. L. Shi, J. Shu, W. Zhang, Y. Liu, HFL-DP: hierarchical federated learning with differential privacy, in *2021 IEEE Global Communications Conference (GLOBECOM)*, (2021), 7–11. https://doi.org/10.1109/GLOBECOM46510.2021.9685644

28. T. Zhou, Hierarchical federated learning with gaussian differential privacy, in *AISS '22: Proceedings of the 4th International Conference on Advanced Information Science and System*, (2022), 61. https://doi.org/10.1145/3573834.3574544

29. M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in *CCS '15: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, (2015), 1322–1333. https://doi.org/10.1145/2810103.2813677

30. H. Zhu, F. Yin, S. Peng, X. Tang, Differentially private hierarchical tree with high efficiency, *Comput. Secur.*, **118** (2022), 102727. https://doi.org/10.1016/j.cose.2022.102727

AIMS Press

©2023 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)