Gain Without Pain: Offsetting DP-Injected Noises Stealthily in Cross-Device Federated Learning

Wenzhuo Yang, Yipeng Zhou, *Member, IEEE*, Miao Hu, *Member, IEEE*, Di Wu, *Senior Member, IEEE*, Xi Zheng, Jessie Hui Wang, *Member, IEEE*, Song Guo, *Fellow, IEEE*, and Chao Li

***Abstract*—Federated learning (FL) is an emerging paradigm through which decentralized devices can collaboratively train a common model. However, a serious concern is the leakage of privacy from exchanged gradient information between clients and the parameter server (PS) in FL. To protect gradient information, clients can adopt differential privacy (DP) to add additional noises and distort original gradients before they are uploaded to the PS. Nevertheless, the model accuracy will be significantly impaired by DP noises, making DP impracticable in real systems. In this work, we propose a novel noise information secretly sharing (NISS) algorithm to alleviate the disturbance of DP noises by sharing negated noises among clients. We theoretically prove that: 1) if clients are trustworthy, DP noises can be perfectly offset on the PS and 2) clients can easily distort negated DP noises to protect themselves in case that other clients are not totally trustworthy, though the cost lowers model accuracy. NISS is particularly applicable for FL across multiple Internet of Things (IoT) systems, in which all IoT devices need to collaboratively train a model. To verify the effectiveness and the superiority of the NISS algorithm, we conduct experiments with the MNIST and CIFAR-10 data sets. The experimental results verify our analysis and demonstrate that NISS can improve model accuracy by 19% on average and obtain better privacy protection if clients are trustworthy.**

***Index Terms*—Differential privacy (DP), federated learning (FL), secretly offsetting.**

Manuscript received 15 January 2021; revised 14 April 2021; accepted 21 July 2021. Date of publication 3 August 2021; date of current version 7 November 2022. This work was supported in part by the National Natural Science Foundation of China under Grant U1911201, Grant U2001209, Grant 62072486, Grant 61802452, and Grant 61872310; in part by the Science and Technology Program of Guangzhou, China, under Grant 202002020045; in part by the Project PCL Future Greater-Bay Area Network Facilities for Large-Scale Experiments and Applications under Grant LZC0019; in part by the Natural Science Foundation of Guangdong under Grant 2021A1515011369; in part by the Australian Research Council Project under Grant ARC LP190100676 and Grant DE180100950; in part by the Hong Kong RGC Research Impact Fund (RIF) under Project R5060-19; in part by the General Research Fund (GRF) under Project 152221/19E and Project 15220320/20E; and in part by the Shenzhen Science and Technology Innovation Commission under Grant R2020A045. *(Corresponding author: Di Wu.)*

Wenzhuo Yang and Miao Hu are with the School of Computer Science and Engineering and the Guangdong Key Laboratory of Big Data Analysis and Processing, Sun Yat-sen University, Guangzhou 510006, China (e-mail: yangwzh8@mail2.sysu.edu.cn; humiao5@mail.sysu.edu.cn).

Yipeng Zhou is with the Department of Computing, FSE, Macquarie University, Sydney, NSW 2109, Australia, and also with the PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen 518000, Guangdong, China (e-mail: yipeng.zhou@mq.edu.au).

Di Wu is with the School of Computer Science and Engineering and the Guangdong Key Laboratory of Big Data Analysis and Processing, Sun Yat-sen University, Guangzhou 510006, China, and also with the PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen 518000, Guangdong, China (e-mail: wudi27@mail.sysu.edu.cn).

Xi Zheng is with the Department of Computing, FSE, Macquarie University, Sydney, NSW 2109, Australia (e-mail: james.zheng@mq.edu.au).

Jessie Hui Wang is with the Institute for Network Sciences and Cyberspace, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: jessiewang@tsinghua.edu.cn).

Song Guo is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: cssongguo@comp.polyu.edu.hk).

Chao Li is with the Security Product Department 2, Tencent Technology (Shenzhen) Company Ltd., Shenzhen 518000, China (e-mail: ethancli@tencent.com).

Digital Object Identifier 10.1109/JIOT.2021.3102030

I. Introduction

With the remarkable development of Internet of Things (IoT) systems, IoT devices, such as mobile phones, cameras, and Industrial IoT (IIoT) devices, have been widely deployed in our daily life [1], [2]. On the one hand, IoT devices with powerful computing and communication capacity are generating more and more data. On the other hand, to provide more intelligent services, decentralized IoT devices have the motivation to collaborate via federated learning (FL) so that distributed data can be fully exploited for model training [3], [4].

The training process via FL can be briefly described as follows. In a typical FL system, a parameter server (PS) is deployed to aggregate gradients uploaded by clients, and distribute aggregated results back to clients [5]–[8]. The model training process terminates after exchanging the gradient information between clients and the PS for a certain number of rounds. However, it has been studied in [9]–[12] that it can lead to the leakage of user privacy if the gradient information is disclosed. In addition, the PS is not always trustworthy [5], [13], which also possibly invades user privacy.

Recently, it has been extensively investigated by academia and industry to adopt differential privacy (DP) on each client [14]–[18] so as to protect the gradient information. DP can distort original gradients by adding additional noises, which however unavoidably distorts the aggregated gradients on the PS and hence impairs the model accuracy [14]. It has been reported in the work [11], [14] that DP noises can significantly lower model accuracy by 10%–30%. It implies that straightly implementing DP in real systems is impracticable when high model accuracy is required [16].

To alleviate the disturbance of DP noises on the aggregated gradients without compromising user privacy, we propose an algorithm to secretly offset DP noises. The idea of our work can be explained by the example shown in Fig. 1. There are three clients and the model to be trained by these clients is

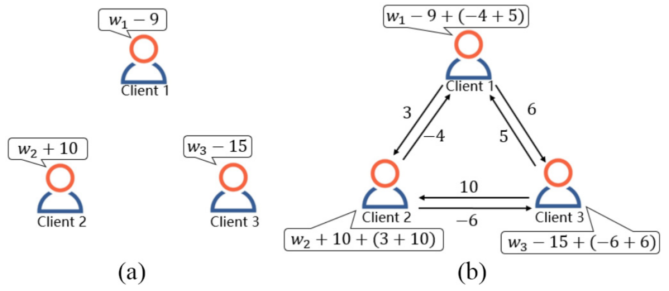


Fig. 1. Case with clients whose noises can be perfectly offset among themselves. Here, w_k represents the parameter for client k. (a) Clients add noises to their parameters. (b) Noises of clients are perfectly offset.

represented by the parameter w_k for client k. Each client distorts the original gradient information by adding a random number, as presented in Fig. 1(a). We suppose that the random number is generated according to the Gaussian distribution determined by the client's privacy budget. However, the noise can be offset if a client can negate and split its noise into multiple shares, and distribute these negated shares with other clients, as presented in Fig. 1(b). Each client uploads its gradients plus all noises (i.e., its own noises and negated noise shares from other clients) to the PS, and then these noises can be perfectly offset among themselves.

Inspired by this example, we propose the noise information secretly sharing (NISS) algorithm through which clients can secretly share their noise information with each other. We theoretically prove that: 1) if clients are trustworthy, DP noises can be perfectly offset on the PS without compromising privacy protection and 2) clients can easily distort negated noise shares received from other clients in case that other clients are not totally trustworthy. We also investigate the extreme case that the PS colludes with other clients to crack the gradient information of a particular client. In this extreme case, there is a tradeoff between model accuracy and privacy protection, and model accuracy cannot be improved without compromising privacy protection. However, we would like to emphasize that NISS is particularly applicable for FL across multiple IoT systems. IoT devices within the same system can trust each other to certain extent so that the model accuracy can be improved accordingly. Besides, devices within the same system can be connected with high-speed networks so that the communication overhead caused by NISS is acceptable.

Our main contributions are summarized as follows.

1) We propose the NISS algorithm that can secretly offset noises generated by DP adopted by each client so that the disturbance on the aggregated gradients can be removed.
2) We theoretically prove that the DP noises can be perfectly offset if clients are trustworthy. Even if clients are not totally trustworthy, clients can still protect themselves by distorting the negated noise shares transmitted between clients.
3) Finally, we conduct experiments with the MNIST and CIFAR-10 data sets, and the experimental results demonstrate that the NISS algorithm can obtain better privacy protection and higher accuracy.

The remainder of this article is organized as follows. In Section II, we introduce the related work on FL, DP, and secure multiparty computing (SMC). In Section III, we introduce the preliminary knowledge. In Section IV, we elaborate the NISS algorithm. In Section V, we present the analysis of noise offsetting and security. In Section VI, we show the simulations, compare our scheme with other schemes, and discuss the experimental results. Finally, we conclude this article in Section VII.

II. Related Work

A. Federated Learning

FL, as a recent advance of distributed machine learning, empowers participants to collaboratively train a model under the orchestration of a central PS, while keeping the training data decentralized [5]. It was first proposed by Google in 2016 [6]. During the training process, each participant's raw data is stored locally and will not be exchanged or transferred for training. FL has the advantages of making full use of IoT computing power with preserved user privacy.

The work in [6] first proposed FedAVG, which is one of the most widely used model average algorithms in FL. The work in [19] analyzed the convergence rate of FedAVG with non-IID data simple distributions. The work [5] and [13] showed a comprehensive introduction to the history, technical methods, and unresolved problems in FL. The work in [18] proved that the bare FedAVG can protect the privacy of participants to some extent. However, only exchanging gradients information still has a high risk of privacy leakage [9]–[12]. Despite tremendous efforts contributed by prior works, there exist many issues in FL that have not been solved very well, such as inefficient communication and device variability [13], [14], [20].

B. Differential Privacy

DP is a very effective mechanism for privacy preservation that can be applied in FL [14], [16]–[18]. It uses a mechanism to generate random noises that are added to query results so as to distort original values.

The most commonly used mechanism for adding noises to FL is the Gaussian mechanism. The work in [16] investigated how to apply the Gaussian mechanism in machine learning systems. Then, the work in [18] studied how to use the Gaussian mechanism in FL. In [14], an FedSGD with the DP algorithm is proposed for FL systems and its convergence rate is analyzed. The work in [11] introduced a novel method named DLG to measure the level of privacy preservation in FL. In FL with DP, a higher ϵ implies a smaller variance of DP noises and, hence, a lower level of privacy preservation. Model accuracy can be largely affected by DP noises [11], [14].

In the field of IoT, FL with DP has also attracted a lot of attention recently. Briggs *et al.* [21] surveyed a wide variety of papers on privacy-preserving methods that are crucial for FL in IoT. The work in [22] designed an FL system with DP leveraging the reputation mechanism to assist home appliance manufacturers to train a machine learning model based on customers' data. The work in [23] proposed to integrate FL and DP to facilitate crowdsourcing applications to generate a machine learning model.

Basically, there is a tradeoff between the extent of privacy protection and model accuracy if DP is straightly incorporated into FL. Different from these works, we devise a novel algorithm through which clients can generate negatively correlated DP noises to get rid of the negative influence on model accuracy.

C. Secure Multiparty Computing

Other than DP, SMC is another effective way for privacy preservation in FL. In previous studies, SMC has been used in many machine learning models [24]–[27]. Presently, secret sharing (SS) and homomorphic encryption (HE) are two main ways in SMC to protect privacy in FL.

HE performs complicated computation operations on gradients. During the gradient aggregation and transmission, it is always calculated in an independent encryption space, instead of directly using the raw gradients value [28], [29]. SS is a method to generate several shares for a secret and send them to several participants. As long as most of the participants are present, the secret can be recovered. In FL, participants can add masks to their gradients and share their masks as a secret to others. If the PS can receive returns from a sufficient number of participants, the masks can be eliminated. Several works based on SS in FL have been proposed in [30]–[32].

However, SS and HE consume too much computing resources, which prohibit their deployment in the real world [33]. In fact, our work is a combination of SS and DP, but the computation overhead of our noise-sharing scheme is very low.

III. Preliminaries

To facilitate the understanding of our algorithm, the list of main notations used in our work is presented in Table I.

TABLE I
LIST OF SYMBOLS

Symbol	Meaning
K	The number of clients
k	The index of clients
t	The index of global training round
E	The number of local training round
η	The learning rate
h	The dimension of the parameters
ℓ	The loss function
$\nabla\ell$	The gradient of function ℓ
m	The number of clients in each global round
d_k	The cardinality of $\mathcal{D}_k$
p_k	The aggregation weight of client k
σ^2	The unit noise variance
σ_k^2	The Gaussian noise variance of client k
$\mathcal{N}$	Gaussian Distribution
$\mathcal{D}_k$	The dataset of client k
$\mathcal{M}_t$	The client set of m client in round t
$\mathbf{w}$	The global model parameters
$\mathbf{w}^k$	The local model parameters of client k
$\mathbf{n}$	The noise generated by DP mechanism
$\mathbf{r}$	The negated noise
s	A random variables to distort $\mathbf{r}$
τ_k^2	The variance of s
$\mathbb{I}_h$	The $h \times h$ identity matrix
ϵ, δ	DP parameters

A. Differential Privacy

It was assumed that user privacy will not be leaked if only gradient information is disclosed. However, it was investigated in [9]–[12] that privacy information can be reconstructed through gradient information. Therefore, it was proposed in [16] that clients can adopt DP to further disturb their gradient information by adding additional noises to their disclosed information. According to the prior work [15], an algorithm satisfying (ϵ, δ)-DP is defined as follows.

Definition 1: A randomized mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{R}$ with domain $\mathcal{X}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differentially privacy if for any two adjacent databases $\mathcal{D}_i, \mathcal{D}_i' \in \mathcal{X}$ and for any subset of outputs $S \subseteq \mathcal{R}$

$$\Pr[\mathcal{M}(\mathcal{D}_i) \in S] \leq e^{\varepsilon} \Pr\big[\mathcal{M}\big(\mathcal{D}_i'\big) \in S\big] + \delta. \tag{1}$$

Here, ϵ is the privacy budget which is the distinguishable bound of all outputs on adjacent databases $\mathcal{D}_i$ and $\mathcal{D}_i'$. δ represents the probabilities that two adjacent outputs of the databases $\mathcal{D}_i, \mathcal{D}_i'$ cannot be bounded by ϵ after using algorithm $\mathcal{M}$. ϵ is also called the privacy budget. Intuitively, a DP mechanism $\mathcal{M}$ with a smaller privacy budget ϵ has a stronger privacy protection and vice versa.

Theorem 1 (Gaussian Mechanism): Let $\epsilon \in (0, 1)$ be arbitrary and $\mathcal{D}_i$ denote the database. For $c^2 > 2\ln(1.25/\delta)$, the Gaussian mechanism $\mathcal{M} = f(\mathcal{D}_i) + \mathcal{N}(0, \sigma^2)$ with parameter $\sigma \geq [(c\Delta f)/\epsilon]$ is (ϵ, δ)-differentially private. Here, $f(\mathcal{D}_i)$ represents the original output and Δf is the sensitivity of f given by $\Delta f = \max_{\mathcal{D}_i, \mathcal{D}_i'} \|f(\mathcal{D}_i) - f(\mathcal{D}_i')\|_2$.

For detailed proof, refer to [15].

We assume that the Gaussian mechanism is adopted in our work because it is convenient to split DP noises obeying the Gaussian distribution into multiple shares [16].

B. DP-FedAVG

FedAVG is the most commonly used model average algorithm in FL, and thereby FedAVG is used for our study. Based on previous works [6], [14], [17], [18], we present the client-based DP-FedAVG here to ease our following discussion.

Without loss of generality, we assume that there are K clients. The client k owns a private data set $\mathcal{D}_k$ with cardinality d_k. These clients target to train a model with parameters represented by the vector $\mathbf{w} \in \mathbb{R}^h$. In FedAVG, clients need to exchange model parameters for multiple rounds with the PS. Each round is also called a global iteration. At the beginning of global round t, each participating client receives the global parameters $\mathbf{w}_t$ from the PS to conduct a number of local iterations. Then, clients return their locally updated model parameters plus DP noises to the PS. By receiving the computation results from a certain number of clients, the PS aggregates received parameters and embarks a new round of global iteration. The detail of the DP-FedAVG algorithm is presented in Algorithm 1.

Algorithm 1: DP-FedAVG Algorithm

```
PS executes:
Initialize $\mathbf{w_0}$;
for each round $t = 1, 2, \ldots$ do
    $m \leftarrow \max(C \times K, 1)$
    $\mathcal{M}_t \leftarrow$(Random set of $m$ clients)
    for each client $k \in \mathcal{M}_t$ in parallel do
        $\widetilde{\mathbf{w}}_{t+1}^k \leftarrow$ ClientUpdate($k$, $\mathbf{w}_t$, $p_k$)
    $\mathbf{w}_{t+1} \leftarrow \sum_{k\in\mathcal{M}_t} \widetilde{\mathbf{w}}_{t+1}^k$

ClientUpdate($k, \mathbf{w}_t, p_k$)
$\mathcal{B} \leftarrow$(split $\mathcal{D}_k$ into batches of size $B$)
for each local round $i$ from 1 to $E$ do
    for batch $b \in \mathcal{B}$ do
        $\mathbf{w} \leftarrow \mathbf{w} - \eta\nabla\ell(w; b)$
$\sigma_k^2 \leftarrow$ (Gaussian Mechanism)
$\mathbf{n}_{t+1}^k \leftarrow \mathcal{N}(0, \sigma_k^2\mathbb{I}_h)$
return $p_k\mathbf{w} + \mathbf{n}_{t+1}^k$
```

In Algorithm 1, C is the fraction of clients that participate each global iteration, $\mathbf{n}_{t+1}^k$ is the Gaussian noise and p_k is the aggregation weight of client k, and $\mathcal{M}_t$ is the set of clients that participate in round t. Usually, $p_k = d_k/(\sum_{i\in\mathcal{M}_t} d_i)$. $\mathcal{B}$ is the set of local sample batches, E is the number of local iterations to be conducted, and η is the learning rate.

Let $f_k(\mathcal{D}_k, \mathbf{w}, p_k)$ represent the function returning the locally updated parameters with inputs $\mathbf{w}$ and p_k. The sensitivity of f_k is denoted by Δf_k. We assume that the privacy budget of client k is represented by ϵ_k and δ_k.

Corollary 1: Algorithm 1 satisfies (ϵ, δ)-differentially private, if $\mathbf{n}_{t+1}^k$ is sampled from $\mathcal{N}(0, \sigma_k^2\mathbb{I}_h)$ where $\sigma_k \geq [(c\Delta f_k)/\epsilon_k]$, $c^2 > 2\ln(1.25/\delta_k)$ and $\mathbb{I}_h$ is the $h \times h$ identity matrix.

Here, h is the model dimension. The proof is straightforward from Theorem 1.

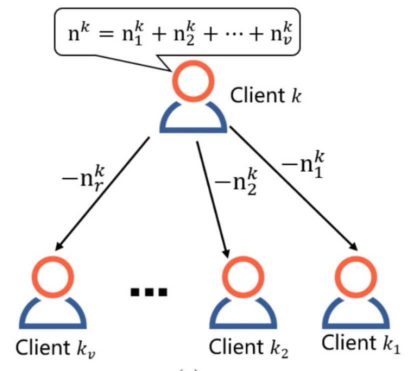
According to Algorithm 1, the disturbance of the DP noises on the aggregated parameters is

$$\mathbf{w}_{t+1} \leftarrow \sum_{k\in\mathcal{M}_t} p_k\mathbf{w}_{t+1}^k + \sum_{k\in\mathcal{M}_t} \mathbf{n}_{t+1}^k. \quad (2)$$

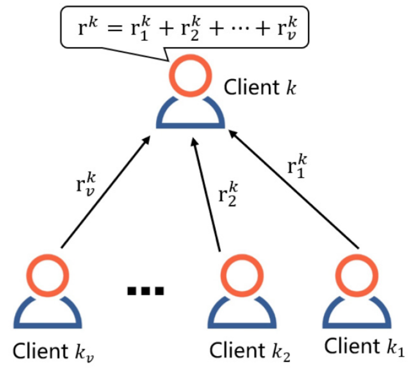
From the right-hand side of (2), we can see that the first term represents the aggregated parameters while the second term represents the disturbance of the DP noises. They are independently generated by all participating clients and, therefore, the variance of $\sum_{k\in\mathcal{M}_t} \mathbf{n}_{t+1}^k$ is $\sum_{k\in\mathcal{M}_t} \sigma_k^2\mathbb{I}_h$. Apparently, if the privacy budget ϵ_k is smaller, σ_k is higher, and the total variance on the server side is higher. Our approach is to make these noises negatively correlated so that the aggregated noise variance can be reduced.

IV. NISS Algorithm

In this section, we introduce the NISS algorithm in detail and leave the analysis of the reduced variance on the aggregation and the security analysis of NISS in the next section.



(a)



(b)

Fig. 2. Workflow of NISS for a particular client. (a) Client sends out negated noise shares. (b) Client receives noise shares from other clients.

A. Illustrative Example

Before diving into the detailed NISS algorithm, we present a concrete example to illustrate how NISS works. According to Algorithm 1, $\mathbf{n}_k$ is sampled from $\mathcal{N}(0, \sigma^2\mathbb{I}_h)$ since the dimension of $\mathbf{w}$ is h. It means that noises of h dimensions are generated independently and the noise offset is conducted for each dimension independently. Thus, to simplify our discussion, we only need to consider the noise n^k for a particular dimension of client k, and n^k is sampled from $\mathcal{N}(0, \sigma^2)$.

According to the property of the Gaussian distribution, n^k can be split into v shares and each share is sampled from $\mathcal{N}(0, [\sigma^2/v])$. The client can send out v negated share to v neighboring clients. If all clients conduct the same operation, the client is expected to receive v noise shares from other clients as well, which can be denoted as $r^k = (r_1^k, \ldots, r_v^k)$. To ease our understanding, the process is illustrated in Fig. 2.

Then, the client adds both its own noise n^k and the sum of negated noise shares received from other clients to its parameter w^k before it submits $p_k w^k + n^k + r^k$ to the PS. The $n^k + r^k$ can preserve the privacy of client k while r^k can be used to offset the noises generated by other clients by the PS. Since v negated shares are generated randomly, no other client can exactly obtain the noise information of client k. In addition, the parameter information is only disclosed to the PS. As long as all other clients are trustworthy, these DP noises can be offset perfectly by negated noise shares.

B. Algorithm Design

We proceed to design the NISS algorithm based on the FedAVG algorithm introduced in the last section.

First, a tracker server is needed so that clients can send and receive negated noise shares with each other. Each client needs to contact the tracker server to fetch a list of neighbor clients before it sends out negated noise shares. The tracker server is only responsible for recording live clients in the system and returning a random list of clients as neighbors for a particular client k. Obviously, the tracker server does not receive any noise information, and hence will not intrude the user privacy. It can be implemented with light communication cost, similar to the deployment of the tracker server in peer-to-peer file-sharing systems [34].

In NISS, the operation of the PS is the same as that in FedAVG. The only difference lies in the operation of each client. Based on its own privacy budget and function sensitivity, the client k needs to determine σ_k so that $w^k + \mathcal{N}(0, \sigma_k^2)$ satisfies (ϵ_k, δ_k)-differentially privacy. Then, the client can determine the number of noise shares according to $v = (\sigma_k^2/\sigma^2)$ so that the client can generate v noise shares and negated noise shares. Here, σ is a number much smaller than σ_k and σ can be a common value used by all clients. $\mathcal{N}(0, \sigma^2)$ is also called a unit noise.

Because clients disclose their noise information with other clients, the gradient information can be cracked to a certain extent if some clients are not trustworthy and collude with the PS to intrude the privacy of a particular client. To prevent the leakage of user privacy, we propose to multiply a noise component s_i to the received negated noise share r_i. s_i is also sampled from the Gaussian distribution $\mathcal{N}(1, \tau_k^2)$. Due to the disturbance of s_i, no other client and the PS can exactly crack the gradient information of the client. τ_k^2 can be set according to the probability that other clients will collude with the PS. How to exactly set τ_k^2 and the role of s_i will be further analyzed in the next section.

By wrapping up, the details of the NISS algorithm are presented in Algorithm 2.

Algorithm 2: NISS Algorithm

```
PS executes:
Initialize $\mathbf{w_0}, \sigma^2$;
for each round $t = 1, 2, ...$ do
    $m \leftarrow \max(C \times K, 1)$
    $\mathcal{M}_t \leftarrow$(Random set of $m$ clients)
    for each client $k \in \mathcal{M}_t$ in parallel do
        $\widetilde{\mathbf{w}}_{t+1}^k \leftarrow$ClientUpdate$(k, \mathbf{w}_t, p_k, \sigma^2)$
    $\mathbf{w}_{t+1} \leftarrow \sum_{k \in \mathcal{M}_t} \widetilde{\mathbf{w}}_{t+1}^k$

ClientUpdate$(k, \mathbf{w}_t, p_k, \sigma^2)$:
$\mathcal{B} \leftarrow$(split $\mathcal{D}_k$ into batches of size $B$)
for each local round $i$ from 1 to $E$ do
    for batch $b \in \mathcal{B}$ do
        $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \ell(w; b)$
$\widetilde{\mathbf{n}}_{t+1}^k \leftarrow$ ClientShare$(k, \sigma^2)$
return $p_k \mathbf{w} + \widetilde{\mathbf{n}}_{t+1}^k$.

ClientShare$(k, \sigma^2)$:
$\sigma_k^2 \leftarrow$ (Gaussian mechanism)
$\tau_k^2 \leftarrow$(Client $k$'s setting)
$v \leftarrow \sigma_k^2/\sigma^2$
$\mathcal{U} \leftarrow$(Random set of $v$ clients in this round)
for $i = 1, 2, ...., v$ do
    $\mathbf{n}_i \leftarrow \mathcal{N}(0, \sigma_k^2 \mathbb{I}_h)$
    $u_i \leftarrow$(Connect with $i$-th client in $\mathcal{U}$)
    Send $(-\mathbf{n}_i)$ to $u_i$
    Receive $\mathbf{r}_i$ from $u_i$
    $s_i \leftarrow \mathcal{N}(1, \tau_k^2)$
$\widetilde{\mathbf{n}} = \sum_{i=1}^{v} (\mathbf{n}_i + s_i \mathbf{r}_i)$
return $\widetilde{\mathbf{n}}$
```

V. Theoretic Analysis

In this section, we conduct an analysis to show how much noise variance can be reduced by NISS on the PS side and how the NISS algorithm defends against attacks. Based on our analysis, we also discuss the application of NISS in real FL systems.

A. Analysis of Noise Offsetting

Similar to Section IV-A, to simplify our discussion, we only consider the noise offsetting for a particular dimension. Let n^k and r^k denote noise shares and negated noise shares received from other clients for client k, respectively. Let $l_{k,i}$ denote the client that receive the ith negated noise share from client k.

Based on Algorithm 2, the client k uploads $p_k w^k + \sum_{i=1}^{v} (n_i^k + r_i^k)$. The aggregation conducted on the PS becomes

$$\sum_{k \in \mathcal{M}} p_k w^k + \sum_{k \in \mathcal{M}} \sum_{i=1}^{v} \left(n_i^k + s_i^k r_i^k\right),$$

$$= \sum_{k \in \mathcal{M}} p_k w^k + \sum_{k \in \mathcal{M}} \sum_{i=1}^{v} \left(n_i^k - s^{l_{k,i}} n_i^k\right).$$

Here, n_i^k is sampled from $\mathcal{N}(0, \sigma^2)$ and $s^{l_{k,i}}$ is sampled from $\mathcal{N}(1, \tau_l^2)$. l is the abbreviation of $l_{k,i}$ if its meaning is clear from the context. Let $\mathbb{V} = \sum_{k \in \mathcal{M}} \sum_{i=1}^{v} (n_i^k - s^{l_{k,i}} n_i^k)$ denote the aggregated DP noises and our study focuses on the minimization of $\mathbb{V}$.

Let us first analyze the variance of a particular noise share after offsetting.

Lemma 1: The variance of a noise share plus its negated share is

$$\mathbf{Var}\left[n_i^k - s^{l_{k,i}} n_i^k\right] = \tau_l^2 \sigma^2. \tag{3}$$

Here, client l receives the negated share of n_i^k and $s^{l_{k,i}}$ is the noise imposed by client l.

Proof: According to the definition of the variance, we can obtain

$$\begin{aligned}\mathbf{Var}\Big[n_i^k - s^{l_{k,i}} n_i^k\Big] &= \mathbf{Var}\Big[\Big(1 - s^{l_{k,i}}\Big) n_i^k\Big] \\ &= \mathbf{E}\Big[\Big(1 - s^{l_{k,i}}\Big)^2 \Big(n_i^k\Big)^2\Big] \\ &\quad - \Big(\mathbf{E}\Big[\Big(1 - s^{l_{k,i}}\Big)\Big(n_i^k\Big)\Big]\Big)^2 \\ &= \mathbf{E}\Big[\Big(1 - s^{l_{k,i}}\Big)^2\Big]\mathbf{E}\Big[\Big(n_i^k\Big)^2\Big] \\ &= \mathbf{E}\Big[\Big(1 - s^{l_{k,i}}\Big)^2\Big]\sigma^2 \\ &= \mathbf{E}\Big[1 - 2s^{l_{k,i}} + \Big(s^{l_{k,i}}\Big)^2\Big]\sigma^2 \\ &= \Big(1 - 2\mathbf{E}\Big[s^{l_{k,i}}\Big] + \mathbf{E}\Big[\Big(s^{l_{k,i}}\Big)^2\Big]\Big)\sigma^2 \\ &= \Big(1 - 2 + \Big(1 + \tau_l^2\Big)\Big)\sigma^2 \\ &= \tau_l^2 \sigma^2. \end{aligned} \tag{4}$$

The above formula holds because n_i^k is sampled from $\mathcal{N}(0, \sigma^2)$ and $s^{l_{k,i}}$ is sampled from $\mathcal{N}(1, \tau_l^2)$. So we can obtain $\mathbf{E}[(n_i^k)^2] = \mathbf{Var}[n_i^k] + \mathbf{E}[n_i^k] = \sigma^2$ and $\mathbf{E}[(s^{l_{k,i}})^2] = 1 + \tau_l^2$ similarly. Apparently, n_i^k and $s^{l_{k,i}}$ are dependent according to our algorithm. ■

Theorem 2: After noise offsetting, the variance of the aggregated noise on the PS side is

$$\mathbb{V} = \sum_{k \in \mathcal{M}} \sigma_k^2 \tau_k^2. \tag{5}$$

Proof: According to Lemma 1, because each n_i and s_i are dependent, we can obtain

$$\mathbf{Var}\Bigg[\sum_{k \in \mathcal{M}} \widetilde{n}^k\Bigg] = \mathbf{Var}\Bigg[\sum_{k \in \mathcal{M}} \sum_{l \in \mathcal{U}^k} \Big(n_i^k - s^{l_{k,i}} n_i^k\Big)\Bigg]. \tag{6}$$

Since client k will send and receive v_k noise shares and negated noise shares, thus each τ_l^2 will be added v_l times according to Algorithm 2. By substituting l by k, we can obtain

$$\begin{aligned}\mathbf{Var}\Bigg[\sum_{k \in \mathcal{M}} \sum_{l \in \mathcal{U}^k} \Big(n_i^k - s^{l_{k,i}} n_i^k\Big)\Bigg] &= \sum_{k \in \mathcal{M}} \sum_{l \in \mathcal{U}^k} \tau_l^2 \sigma^2 \\ &= \sigma^2 \sum_{k \in \mathcal{M}} v_k \tau_k^2 = \sum_{k \in \mathcal{M}} \sigma_k^2 \tau_k^2. \end{aligned} \tag{7}$$

The third equality holds because $v_k = \sigma_k^2 / \sigma^2$. ■

Remark: From Theorem 2, we can observe that $\mathbb{V} = 0$ if $\tau = 0$ implying that DP noises are perfectly offset on the PS side. However, if $\tau = 1$, the value of $\mathbb{V}$ is the same as that without any noise offsetting. The value of τ depends on the trustworthiness between clients. We will further discuss how to set τ after the security analysis in the next section.

B. Security Analysis

We conduct the security analysis through analyzing the privacy preservation for a particular client. We suppose that the target of a particular client is to satisfy the (ϵ_k, δ_k)-differentially private.

It is easy to understand that the NISS algorithm satisfies the (ϵ_k, δ_k)-differentially private by setting $\tau_k = 0$ or $s^l = 1$, if the PS and clients do not collude. What client k submits to the PS is $w^k + \sum_{i=1}^{v}(n_i^k + r_i^k)$. The noise $\sum_{i=1}^{v}(n_i^k + r_i^k)$ is also a Gaussian random variable with variance $2\sigma_k^2$ and, hence, the NISS algorithm on client k satisfies (ϵ_k, δ_k)-differentially private. Meanwhile, no other client can crack the parameter information since the parameter information is only disclosed to the PS.

However, it is not guaranteed that the PS never colludes with clients. To conduct more general analysis, we assume that there is $\rho \in [0, 1]$ fraction of other clients will collude with the PS. The problem is how to set τ_k so that the NISS algorithm on client k can still satisfy (ϵ_k, δ_k)-differentially private.

Let $\mathcal{U}^k$ represent the set of clients that client k will contact. There is no prior knowledge about which client will collude with the PS. The tracker server randomly select clients for $\mathcal{U}^k$. It implies that ρ fraction of $\mathcal{U}^k$ will disclose the noise share information with the PS. We use $\mathcal{U}_\rho^k$ to denote the clients who collude with the PS and $\mathcal{U}_{1-\rho}^k$ to denote the clients who do not collude. Apparently, the size of $\mathcal{U}_\rho^k$ and $\mathcal{U}_{1-\rho}^k$ is ρv and $(1 - \rho)v$. Thus, the effective noise uploaded by client k becomes $\sum_{i \in \mathcal{U}_{1-\rho}^k}(n_i^k + s^{l_{k,i}} r_i^k) + \sum_{i \in \mathcal{U}_\rho^k} s^{l_{k,i}} r_i^k$. To ensure that (ϵ_k, δ_k)-differentially private can be satisfied, it requires $\mathbf{Var}[\sum_{i \in \mathcal{U}_{1-\rho}^k}(n_i^k + s^{l_{k,i}} r_i^k) + \sum_{i \in \mathcal{U}_\rho^k} s^{l_{k,i}} r_i^k] \geq \sigma_k^2$.

Theorem 3: If n^k sampled from $\mathcal{N}(0, \sigma_k^2)$ can make $w^k + n^k$ satisfy (ϵ_k, δ_k)-differentially private, the NISS algorithm satisfies (ϵ_k, δ_k)-differentially private as long as $\tau_k^2 \geq \max\{2\rho - 1, 0\}$. Here, $\rho \in [0, 1)$ represents the percentage of other clients that collude with the PS.

The detailed proof is presented in the Appendix.

Remark: It is worth to mention the special case with $\rho = 1$. According to Theorem 3, $\tau = 1$ and $s^{l_{k,i}}$ is sampled from $\mathcal{N}(1, 1)$ if $\rho = 1$. In this case, $\mathbf{Var}[\sum_{i \in \mathcal{U}_{1-\rho}^k}(n_i^k + s^{l_{k,i}} r_i^k) + \sum_{i \in \mathcal{U}_\rho^k} s^{l_{k,i}} r_i^k] = \mathbf{E}[(\sum_{i=1}^{v} r_i^k)^2]$. According to the central limit theorem, as long as $v \gg 1$, we have $\mathbf{E}[(\sum_{i=1}^{v} r_i^k)^2] \approx \sigma_k^2$. Thus, if $\rho = 1$ and $\tau = 1$, it implies that $\mathbf{Var}[\sum_{i \in \mathcal{U}_{1-\rho}^k}(n_i^k + s^{l_{k,i}} r_i^k) + \sum_{i \in \mathcal{U}_\rho^k} s^{l_{k,i}} r_i^k] = \sigma_k^2$ and $\mathbb{V} = \sum_k \sigma_k^2$. The variance of the aggregated noise on the PS is the same as that without any offsetting operation. In this extreme case, there exists a tradeoff between model accuracy and privacy protection. One cannot improve the model accuracy without compromising privacy protection.

C. Application of NISS in Practice

As we have discussed in the last section, ρ is a vital parameter. Our analysis uses ρ to cover all cases with different fractions of malicious clients colluding with the PS. If ρ is close to 1, it will significantly impair the performance of the NISS algorithm. In practice, ρ can be set as a small value, which can be illustrated from two perspectives.

First, most FL systems are of a large scale with tens of thousands of clients. If there are more normal clients, the fraction of malicious clients that will collude with the PS will be a smaller value. Second, our analysis is based on the assumption that $\mathcal{U}^k$ is randomly selected by the tracker server. In fact, clients can play the coalitional game with other clients they trust. For instance, the IoT devices of the same system can trust each other substantially. They can share negated noise information with each other by setting a small τ_k^2 since the probability that neighboring clients collude with the PS is very low. From this example, we can also conclude that the NISS algorithm is particularly applicable for FL across multiple IoT systems. IoT devices in the sample system can form a coalition so that the variance of the aggregated noise is minimized. Besides, devices within the same system can be connected with high-speed networks so that the communication overhead to transmit noise shares is insignificant.

VI. Experiment

In this section, we conduct experiments with MNIST and CIFAR-10 to evaluate the performance of NISS.

A. Experimental Settings

1) Data Sets: We use public data sets MNIST and CIFAR-10 for our experiments. The MNIST data set contains 60 000 28×28 grayscale images of handwritten digits from 1 to 10. The CIFAR-10 data set consists of 60 000 32×32 color images in ten classes, with 6000 images per class. Both these data sets are divided into two sets: 1) training set and 2) test set. The training set includes 50 000 randomly selected images and the rest 10 000 images are used for testing.

2) Models: We employ simplified convolution neural network (CNN) and multilayer perceptron (MLP) models in [6] and [16] to classify MNIST and CIFAR-10, respectively.

1) *Convolution Neural Network:* For MNIST, we use a CNN with two 5×5 convolution layers (each followed by a max-pooling layer and ReLU activation), then followed by two fully connected layers with 50 and 10 units, respectively. There is a final softmax output layer. The structure of the CNN model used for CIFAR-10 is similar with that for MNIST. The difference is that three fully connected layers with 384, 192, and 10 units, respectively, are used.
2) *Multilayer Perceptron:* There are 2-hidden layers with 64 and 10 units each using ReLU activations. The MLP structure for MNIST and CIFAR-10 is identical.

3) Sample Distribution: We allocate the data set samples among clients with both IID and non-IID manners. For the IID setting, these samples are purely randomly distributed on 100 clients, and each client is allocated 500 images (50 000 in total). For the non-IID setting, the data samples are classified into 200 groups, and the label of each group is identical. Then, each client can only get two randomly selected groups.

4) Simulation Settings: Based on [16], the Gaussian mechanism is used for our experiments to add noises to local parameters. We use experimental settings similar to those in [6], [14], and [16]. The other key parameters used for training are set as follows.

1) Number of clients: $K = 100$.
2) Client participation rate: $C = 0.1$.
3) Local minibatch size: $B = 10$.
4) Number of local rounds: $E = 5$.
5) Unit noise variance: $\sigma^2 = 0.01$.

We set a decreasing learning rate function to run our experiments. For the CNN model with the MNIST data set, we set the initial learning rate as $\eta = 0.01$ which is decayed with a factor 0.995 after each local round. For the CNN model with CIFAR-10 data sets, we set the initial learning rate as $\eta = 0.02$ which is decayed with a factor 0.992 after each local round. For the MLP model with MNIST and CIFAR-10 data sets, the initial learning rate is set as $\eta = 0.01$ with a decay rate 0.995 after each local round. Note that we set the same τ_k and ϵ_k abbreviated as τ and ϵ for all clients. In addition, to implement DP-FedAVG, we use the norm clipping technique with a clipping threshold ζ to restrict the range of the client's gradients. If a client's any gradient exceeds ζ, it will be clipped to ζ. The details of the clipping technique can be found in [14]. For our experiments, we set $\zeta = 3$.

Our experimental environment is Pytorch [35] running on the computer with a processor with six 2.6-GHz Intel i7 cores. The computer is equipped with 16-GB RAM and a GPU of AMD Radeon Pro 5300M.

5) Metrics and Baselines: We use the model accuracy on the test data set to evaluate the accuracy performance of the NISS algorithm. Meanwhile, to evaluate the leak-defense performance, the DLG loss introduced in [11] is also used as the metric. The method uses randomly initialized weights and L-BFGS [36] to match gradients from all trainable parameters. The DLG loss is the gradient match loss for L-BFGS. The DLG loss reflects the degree of discrimination for the raw samples. A smaller DLG loss implies that the smaller the matching loss obtained with the leaked gradients is, the more information is leaked. In addition, we implement FedAVG and DP-FedAVG algorithms as baselines in our experiments.

B. Experimental Results

1) Model Accuracy: In this experiment, the privacy budget of each client is set as $\epsilon = 10$ and $\delta = 10^{-4}$. FedAVG and DP-FedAVG algorithms are implemented as baselines (equivalent to setting $\tau = 0$ and 1, respectively, in NISS).

Fig. 3 shows the results on the test accuracy of training models. Since we have two data sets, IID or non-IID sample distributions and two models, we have eight combinations of experiments. τ denotes the standard deviation of s_i. Then, $\tau = 0$ means the offsetting of NISS is perfect which implies that the accuracy performance of NISS should be very close to that of FedAVG. $\tau = 1$ means the variance of the aggregated noise on the PS side is $\sum_{k \in \mathbf{M}} \sigma_k^2$ which is as same as that of DP-FedAVG. Thus, we use $\tau = 0$ and $\tau = 1$ to denote FedAVG and DP-FedAVG. As shown in Fig. 3, the test accuracy of training models is lower if τ is higher. In other words, the higher τ is, the larger the variance of added noises is, and the more significant the accuracy deteriorates. This is consistent with our analysis. When the sample distribution is IID, our NISS algorithm can increase the test accuracy by about $5 \sim 10\%$ on MNIST and 7%–25% on CIFAR-10 if all clients will not collude with the PS, namely, perfectly offsetting. This is because CIFAR-10 are all three-channel color pictures, and the amount of noise has a higher impact on the

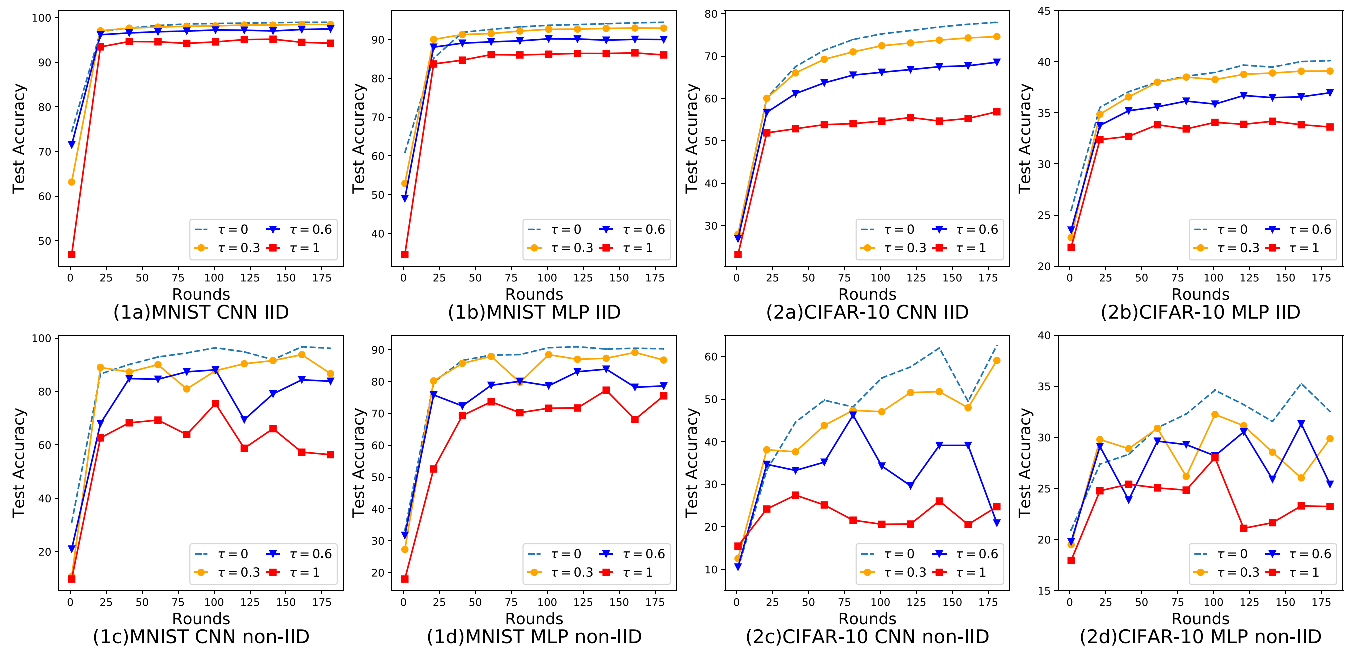


Fig. 3. If $\tau = 0$, FedAVG. If $\tau = 1$, DP-FedAVG. For the rest, NISS with $\tau = 0.3$ and $\tau = 0.6$.

accuracy. When the sample distribution is non-IID, the model accuracy can be improved by 20%–30% for MNIST and 12%–40% for CIFAR-10. In addition, the test accuracy of MLP with CIFAR-10 is low because the structure of the MLP model is too simple for the CIFAR-10 data set. Non-IID data distribution also results in lower test accuracy [37]. Fig. 3 also shows the tradeoff between model accuracy and privacy protection. If we increase τ, the accuracy will decrease and if we decrease τ, the accuracy will increase.

In addition, we execute NISS on MNIST and CIFAR-10 by setting $\epsilon = 5$, 10, and 15, respectively, with $\delta = 10^{-4}$. Results with $\tau = 0.5$ and 1 are compared to evaluate how NISS can improve the final model accuracy of FL.

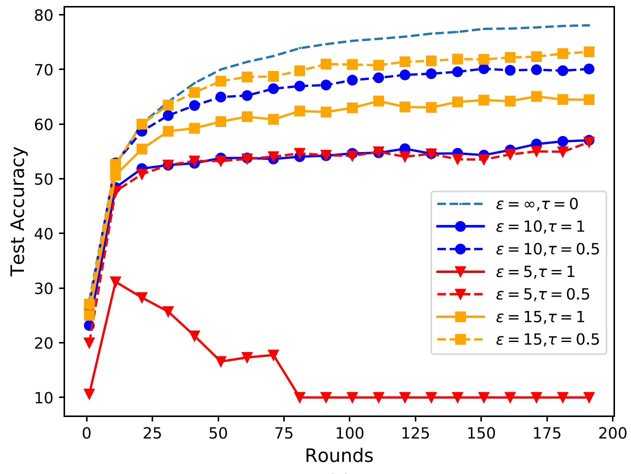
Fig. 4 shows the results on the test accuracy by tuning ϵ. Recall that τ is the fraction of clients that collude with the PS to crack clients' privacy. $\tau = 1$ corresponds the case without any noise offset. Our results show that when the fraction of colluders is less than 0.5, NISS can significantly improve the model accuracy with various privacy budgets. In particular, the improvement is higher when ϵ is smaller because a smaller privacy budget will generate noises with bigger variances resulting in lower model accuracy. When $\epsilon = 5$, NISS with $\tau = 0.5$ can approximately improve the model accuracy by 55% on average. When $\epsilon = 10$ or 15, NISS can also improve the model accuracy by 6.2% on average. This experiment validates that NISS is applicable under various privacy budgets and is particularly effective when the privacy budget is small.

In summary, when $\tau = 0$, since the noise can be offset perfectly, the model accuracy given by NISS is very close to that of FedAVG on the whole and better than that of the DP-FedAVG algorithm if all the clients will not collude with the PS. Even if some clients collude with the PS, by tuning τ, each client can protect its privacy but the model accuracy will decrease.

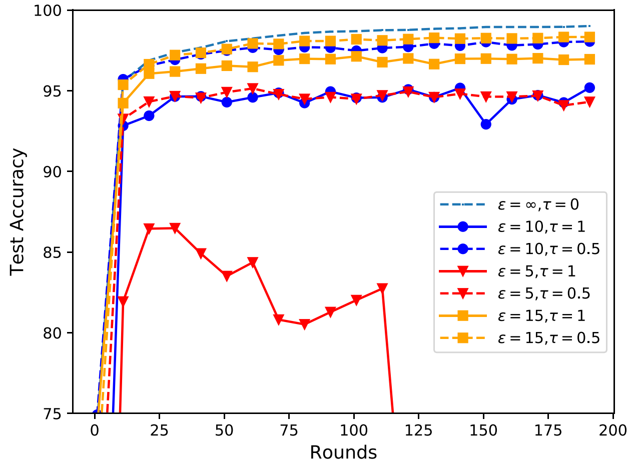
2) Privacy Protection: For this experiment, we use the CIFAR-10 data set with the CNN model. For NISS, we suppose that there is no client who will collude with the attacker to crack clients' information. It means that $\tau = 0$ in NISS. For both DP-FedAVG and NISS, we set the variance of the Gaussian noises is $\sigma_k^2 = 1.6\times10^{-5}$ on each client. We evaluate the leak-defense ability of FedAVG, DP-FedAVG, and NISS by determining whether effective information can be recovered from gradients. We assume that all transmitted gradients via the Internet can be leaked to malicious attackers. An attacker is deployed which tries to reconstruct raw image samples with leaked gradients.

We use the gradients from FedAVG, DP-FedAVG, and NISS to run DLG. Fig. 5 shows the comparison of DLG loss for different algorithms. Images in this figure are the finally reconstructed images by DLG in [11] and the original image is an "automobile." The lower the DLG loss is and the more information is leaked, the clearer final reconstructed image will be. As shown in Fig. 5, FedAVG suffers from severe privacy leakage because the attacker can almost completely reconstruct the raw image. This experiment also indicates that DP-FedAVG cannot perfectly prevent privacy leakage, though the reconstructed image is very vague. In comparison, the attacker cannot reconstruct any useful information if the NISS algorithm is adopted. Recall that DP-FedAVG and NISS pose Gaussian noises of the same variance on individual clients, the model accuracy obtained with NISS should be much higher than that with DP-FedAVG. Thus, we can conclude that NISS can well protect data privacy without compromising model accuracy performance.

This experiment studies the case that $\tau = 0$ implying that there is no client colluding with the PS to crack data privacy. Each client sets $s_i = 1$. According to our analysis in Section V-B, if DP-FedAVG and NISS use the same ϵ_k and δ_k, namely, adding Gaussian noises with the same variance σ_k^2,



(a)



(b)

Fig. 4. Impact of ϵ on NISS. $\epsilon = \infty, \tau = 0$ represents the noise-free FedAvg and $\tau = 1$ represents DP-FedAVG. (a) MNIST CNN IID. (b) CIFAR-10 CNN IID.

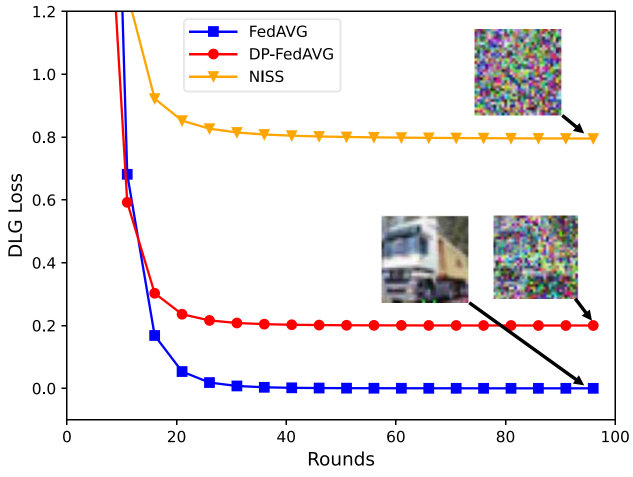


Fig. 5. DLG loss for FedAVG, DP-FedAVG, and NISS. The above images show the effect of finally reconstructed image by DLG in [11] and the original image is an automobile.

then the variance of noises on each individual client should be $2\sigma_k^2$ in NISS. If DP-FedAVG can preserve (ϵ_k, δ_k)-DP, then NISS can preserve $(\epsilon_k/2, \delta_k)$-DP because of the larger variance of noises. Thus, NISS provides a higher level of privacy protection than DP-FedAVG. Meanwhile, when $\tau = 0$, noises injected by NISS can perfectly offset on the PS when noises from different clients are aggregated. Thus, the FL can deduce a model with higher accuracy if NISS is employed.

In summary, the above experiments demonstrate that our NISS algorithm can achieve extraordinary performance. When clients do not collude with the PS, our NISS can achieve the same accuracy as that of FedAVG which is better than DP-FedAVG, and better privacy protection due to its large scale of noise for a single client. If some clients collude with the PS, the client can set its τ_k to protect its privacy. Our experiments also show the tradeoff between model accuracy and privacy protection. If clients set a higher τ_k, the accuracy will be lower and vice versa.

VII. Conclusion

In this work, we propose a novel algorithm called NISS to offset the DP noises independently generated by clients in FL systems. NISS is a method for clients to generate negatively correlated noises. Intuitively, each client splits its noise into multiple shares. Each share is negated and sent out to a neighboring client. Each client uploads its parameter plus its own noise and all negated noise shares received from other neighbors. A noise share of a particular client can be potentially offset by its negated value uploaded by another client. We theoretically prove that the NISS algorithm can effectively reduce the variance of the aggregated noise on the PS so as to improve the model accuracy in FL. Experiments with MNIST and CIFAR-10 data sets are carried out to verify our analysis and demonstrate the extraordinary performance achieved by NISS.

Appendix
Proof of Theorem 3

Proof: We will calculate $\mathbf{Var}[\sum_{i\in\mathcal{U}_{1-\rho}^k}(n_i^k + s^{l_{k,i}} r_i^k) + \sum_{i\in\mathcal{U}_\rho^k} s^{l_{k,i}} r_i^k]$ first. For $i \in \mathcal{U}_{1-\rho}^k$ and $i \in \mathcal{U}_\rho^k$, we will discuss separately. First, for $i \in \mathcal{U}_{1-\rho}^k$, we have

$$\begin{aligned}\mathbf{Var}\left[\sum_{i\in\mathcal{U}_{1-\rho}^k}\left(n_i^k + s^{l_{k,i}} r_i^k\right)\right] &= \sum_{i\in\mathcal{U}_{1-\rho}^k}\mathbf{Var}\left[\left(n_i^k + s^{l_{k,i}} r_i^k\right)\right]\\ &= \sum_{i\in\mathcal{U}_{1-\rho}^k}\mathbf{Var}\left[\left(n_i^k\right] + \mathbf{Var}\left[s^{l_{k,i}} r_i^k\right)\right]\\ &= \sum_{i\in\mathcal{U}_{1-\rho}^k}\sigma^2 + \mathbf{E}\left[\left(s^{l_{k,i}} r_i^k\right)\right)^2\right]\\ &= \sum_{i\in\mathcal{U}_{1-\rho}^k}\sigma^2 + \left(\tau_k^2 + 1\right)\sigma^2\\ &= (1-\rho)\left(\tau_k^2 + 2\right)\sigma_k^2. \qquad (8)\end{aligned}$$

Second, for $i \in \mathcal{U}_\rho^k$, note that here r_i^k is no longer a random variable and it is a certain number which we can approximate using the central limit theorem, then we can calculate it as

$$\begin{aligned}\mathbf{Var}\left[\sum_{i\in\mathcal{U}_\rho^k} s^{l_{k,i}} r_i^k\right] &= \sum_{i\in\mathcal{U}_\rho^k}\mathbf{Var}\left[s^{l_{k,i}} r_i^k\right]\\ &= \sum_{i\in\mathcal{U}_\rho^k}\left(r_i^k\right)^2\mathbf{Var}\left[s^{l_{k,i}}\right]\\ &= \tau_k^2\sum_{i\in\mathcal{U}_\rho^k}\left(r_i^k\right)^2. \qquad (9)\end{aligned}$$

According to the central limit theorem, as long as $\rho v \gg 1$, $\sum_{i\in\mathcal{U}_\rho^k}(r_i^k)^2 \approx \mathbf{E}[\sum_{i\in\mathcal{U}_\rho^k}(r_i^k)^2] = \rho v\sigma^2 = \rho\sigma_k^2$. Thus, we have

$$\mathbf{Var}\left[\sum_{i\in\mathcal{U}_\rho^k} s^{l_{k,i}} r_i^k\right] = \tau_k^2\rho\sigma_k^2. \qquad (10)$$

Then, we can obtain

$$\begin{aligned}&\mathbf{Var}\left[\sum_{i\in\mathcal{U}_{1-\rho}^k}\left(n_i^k + s^{l_{k,i}} r_i^k\right) + \sum_{i\in\mathcal{U}_\rho^k} s^{l_{k,i}} r_i^k\right]\\ &= \mathbf{Var}\left[\sum_{i\in\mathcal{U}_{1-\rho}^k}\left(n_i^k + s^{l_{k,i}} r_i^k\right)\right] + \mathbf{Var}\left[\sum_{i\in\mathcal{U}_\rho^k} s^{l_{k,i}} r_i^k\right]\\ &= (1-\rho)\left(\tau_k^2+2\right)\sigma_k^2 + \tau_k^2\rho\sigma_k^2 \qquad (11)\end{aligned}$$

To ensure that (ϵ_k, δ_k)-differentially private, it requires that

$$(1-\rho)\left(\tau_k^2+2\right)\sigma_k^2 + \tau_k^2\rho\sigma_k^2 \geq \sigma_k^2 \qquad (12)$$

Then, we have

$$\tau_k^2 \geq 2\rho - 1. \qquad (13)$$

Hence, we can obtain $\tau_k^2 \geq \max\{2\rho - 1, 0\}$. ■

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.

[4] J. Li, S. Chu, F. Shu, J. Wu, and D. N. K. Jayakody, "Contract-based small-cell caching for data disseminations in ultra-dense cellular networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 5, pp. 1042–1053, May 2019.

[5] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019. [Online]. Available: arXiv:1912.04977.

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

[7] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016. [Online]. Available: arXiv:1610.02527.

[8] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016. [Online]. Available: arXiv:1610.05492.

[9] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Security Privacy (SP)*, 2019, pp. 691–706.

[10] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," In *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 603–618.

[11] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 14774–14784.

[12] B. Zhao, K. R. Mopuri, and H. Bilen, "IDLG: Improved deep leakage from gradients," 2020. [Online]. Available: arXiv:2001.02610.

[13] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[14] K. Wei *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.

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

[16] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, 2016, pp. 308–318.

[17] N. Wu, F. Farokhi, D. Smith, and M. A. Kaafar, "The value of collaboration in convex machine learning with differential privacy," in *Proc. IEEE Symp. Security Privacy (SP)*, 2020, pp. 304–317.

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

[19] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-iid data," 2019. [Online]. Available: arXiv:1907.02189.

[20] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[21] C. Briggs, Z. Fan, and P. Andras, *A Review of Privacy-Preserving Federated Learning for the Internet-of-Things*. Cham, Switzerland: Springer, 2020.

[22] Y. Zhao *et al.*, "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, Feb. 2021.

[23] Y. Zhao *et al.*, "Local differential privacy based federated learning for Internet of Things," 2020. [Online]. Available: arXiv:2004.08856.

[24] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2000, pp. 439–450.

[25] W. Du, Y. S. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *Proc. SIAM Int. Conf. Data Min.*, 2004, pp. 222–233.

[26] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2002, pp. 639–644.

[27] J. Vaidya, M. Kantarcıoğlu, and C. Clifton, "Privacy-preserving Naïve Bayes classification," *VLDB J.*, vol. 17, no. 4, pp. 879–898, 2008.

[28] S. Hardy *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017. [Online]. Available: arXiv:1711.10677.

[29] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 1333–1345, 2017.

[30] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 1175–1191.

[31] K. Mandal, G. Gong, and C. Liu, "NIKE-based fast privacy-preserving highdimensional data aggregation for mobile devices," Commun. Security Lab, Univ. Waterloo, Waterloo, ON, Canada, Rep. CACR 2018-10, 2018.

[32] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 911–926, 2019.

[33] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, and A. Dubey, "No peek: A survey of private distributed deep learning," 2018. [Online]. Available: arXiv:1812.03288.

[34] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking 2002*, vol. 4673. Bellingham, WA, USA: Int. Soc. Opt. Photon., 2001, pp. 156–170.

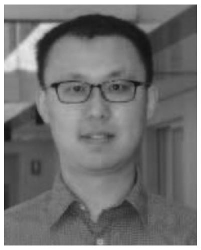
[35] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017, pp. 1–4.

[36] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, nos. 1–3, pp. 503–528, 1989.

[37] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018. [Online]. Available: arXiv:1806.00582.

Wenzhuo Yang received the B.E. degree in computer science from Central South University, Changsha, China, in 2020. He is currently pursuing the M.S. degree with the School of Computer Science, Sun Yat-sen University, Guangzhou, China.

His research interests include federated learning and privacy preserving.



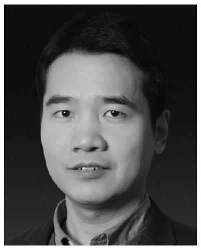
Yipeng Zhou (Member, IEEE) received the bachelor's degree in computer science from the University of Science and Technology of China, Hefei, China, in 2006, and the M.Phil. degree supervised by Prof. D. M. Chiu and Prof. J. C. S. Lui and the Ph.D. degree supervised by Prof. D. M. Chiu from the Information Engineering Department, The Chinese University of Hong Kong (CUHK), Hong Kong, in 2008 and 2012, respectively.

He is a Lecturer of Computer Science with the Department of Computing, Macquarie University, Sydney, NSW, Australia, and also with PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China. From August 2016 to February 2018, he was a Research Fellow with the Institute for Telecommunications Research, University of South Australia, Adelaide, SA, Australia. From September 2013 to September 2016, he was a Lecturer with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. He was a Postdoctoral Fellow with the Institute of Network Coding, CUHK, from August 2012 to August 2013.

Dr. Zhou was the recipient of the ARC Discovery Early Career Research Award in 2018.

Miao Hu (Member, IEEE) received the B.S. and Ph.D. degrees in communication engineering from Beijing Jiaotong University, Beijing, China, in 2011 and 2017, respectively.

He is currently an Associate Professor with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. From September 2014 to September 2015, he was a Visiting Scholar with Pennsylvania State University, State College, PA, USA. His research interests include edge computing, multimedia system, and software-defined network.



Di Wu (Senior Member, IEEE) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 2000, the M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2003, and the Ph.D. degree in computer science and engineering from the Chinese University of Hong Kong, Hong Kong, in 2007.

He was a Postdoctoral Researcher with the Department of Computer Science and Engineering, Polytechnic Institute of New York University, Brooklyn, NY, USA, from 2007 to 2009, advised by Prof. K. W. Ross. He is currently a Professor and the Associate Dean of the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. He is also with the Guangdong Key Laboratory of Big Data Analysis and Processing, Sun Yat-sen University, and PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China. His research interests include edge/cloud computing, multimedia communication, Internet measurement, and network security.

Dr. Wu was the recipient of the IEEE INFOCOM 2009 Best Paper Award and the IEEE Jack Neubauer Memorial Award.

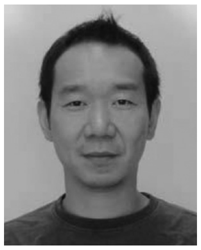
Xi Zheng received the Ph.D. degree in software engineering from the University of Texas at Austin, Austin, TX, USA, in 2015.

From 2005 to 2012, he was the Chief Solution Architect with Menulog, Sydney, NSW, Australia. He is currently the Director of the Intelligent Systems Research Group and the Senior Lecturer (also known as Associate Professor U.S.) and the Deputy Program Leader of Software Engineering with Macquarie University, Sydney. His research interests include CPS verification, machine learning security, human vehicle interaction, edge intelligence, and intelligent software engineering.

Dr. Zheng has a number of highly cited papers and serves as the PC Member for FSE (CORE A*), PerCom (CORE A*), and TrustCom (CORE A).

Jessie Hui Wang (Member, IEEE) received the B.S. and M.S. degrees in computer science from Tsinghua University, Beijing, China, in 1999 and 2001, respectively, and the Ph.D. degree in information engineering from The Chinese University of Hong Kong, Hong Kong, in 2007.

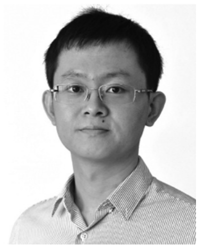
She is currently an Associate Professor with Tsinghua University. Her research interests include Internet routing, cloud computing, data analysis, network measurement, and Internet economics.



Song Guo (Fellow, IEEE) received the B.E. degree from Huazhong University of Science and Technology, Wuhan, China, in 1995, the M.E. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 1998, and the Ph.D. degree from University of Ottawa, ON, Canada, in 2005.

He is a Full Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His research interests are mainly in big data, edge AI, mobile computing, and distributed systems. He published many papers in top venues with wide impact in these areas and was recognized as a Highly Cited Researcher (Clarivate Web of Science).

Prof. Guo is the recipient of over a dozen best paper awards from IEEE/ACM conferences, journals, and technical committees. He also holds a Changjiang Chair Professorship awarded by the Ministry of Education of China. He is the Editor-in-Chief of IEEE OPEN JOURNAL OF THE COMPUTER SOCIETY and the Chair of IEEE Communications Society (ComSoc) Space and Satellite Communications Technical Committee. He was an IEEE ComSoc Distinguished Lecturer and a member of the IEEE ComSoc Board of Governors. He has served for the IEEE Computer Society on Fellow Evaluation Committee, and been named on editorial board of a number of prestigious international journals, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING. He has also served as the chair of organizing and technical committees of many international conferences. He is a Fellow of the Canadian Academy of Engineering.



Chao Li received the bachelor's degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China, in 2006, and the Ph.D. degree from the Electrical and Computer Engineering Department, National University of Singapore, Singapore, in 2012.

He is a Principal Researcher with Tencent Security, Shenzhen, China, and leading the research and development of the risk management product line. From 2012 to 2017, he was the Head of Data Science with Mozat Pte. Ltd., Singapore, and the Property Guru Group, Singapore, respectively.