Privacy-Preserving Federated Learning Framework Based on Chained Secure Multiparty Computing

Yong Li, *Member, IEEE*, Yipeng Zhou, *Member, IEEE*, Alireza Jolfaei, *Senior Member, IEEE*, Dongjin Yu, *Senior Member, IEEE*, Gaochao Xu, *Member, IEEE*, and Xi Zheng, *Member, IEEE*

***Abstract*—Federated learning (FL) is a promising new technology in the field of IoT intelligence. However, exchanging model-related data in FL may leak the sensitive information of participants. To address this problem, we propose a novel privacy-preserving FL framework based on an innovative chained secure multiparty computing technique, named chain-PPFL. Our scheme mainly leverages two mechanisms: 1) single-masking mechanism that protects information exchanged between participants and 2) chained-communication mechanism that enables masked information to be transferred between participants with a serial chain frame. We conduct extensive simulation-based experiments using two public data sets (MNIST and CIFAR-100) by comparing both training accuracy and leak defence with other state-of-the-art schemes. We set two data sample distributions (IID and NonIID) and three training models (CNN, MLP, and L-BFGS) in our experiments. The experimental results demonstrate that the chain-PPFL scheme can achieve practical privacy preservation (equivalent to differential privacy with ϵ approaching zero) for FL with some cost of communication and without impairing the accuracy and convergence speed of the training model.**

***Index Terms*—FedAVG algorithm, federated learning (FL), privacy preservation, secure multiparty computing (SMC).**

I. Introduction

THE CONCEPT of federated learning (FL) was proposed first by Google in 2016 [1]–[3]. The main principle of FL is to protect the privacy of data owners (i.e., participants) by training machine-learning models based on data sets distributed across multiple devices. During the entire model training process, participants only exchange gradients rather than raw data or encrypted/desensitized raw data with the server. In FL, it is necessary to set up a coordinating server that is responsible for aggregating gradients reported by participants and distributing the updated results back to participants.

Manuscript received July 16, 2020; revised August 16, 2020; accepted August 28, 2020. Date of publication September 8, 2020; date of current version April 7, 2021. This work was supported in part by the Australian Research Council Linkage Project under Grant LP190100676; in part by the Science Research and Development Program of Jilin Province in China under Grant 20200401076GX and Grant 20190303021SF; in part by the Natural Science Funds of Jilin Province in China under Grant 20190201180JC; in part by the State Scholarship Fund of CSC in China under Grant 201902335003; and in part by Australian Data61 under Grant CRP c020996. *(Corresponding author: Xi Zheng.)*

Yong Li is with the College of Computer Science and Technology, Jilin University, Changchun 130012, China, also with the College of Computer Science and Engineering, Changchun University of Technology, Changchun 130012, China, and also with the Department of Computing, Macquarie University, Sydney, NSW 2109, Australia (e-mail: yong.li@mq.edu.au).

Yipeng Zhou, Alireza Jolfaei, and Xi Zheng are with the Department of Computing, Macquarie University, Sydney, NSW 2109, Australia (e-mail: yipeng.zhou@mq.edu.au; alireza.jolfaei@mq.edu.au; james.zheng@mq.edu.au).

Dongjin Yu is with the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: yudj@hdu.edu.cn).

Gaochao Xu is with the College of Computer Science and Technology, Jilin University, Changchun 130012, China (email: xugc@jlu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2020.3022911

Due to its ability in privacy preservation, FL has attracted extensive attention from both academia and industry. However, recent studies indicate that exchanging gradients is not safe enough for participants. It is still possible to obtain private training data from the shared updates (i.e., gradients) [4]–[6].

To enhance the privacy preservation of FL in training processes, many schemes and strategies have been proposed, which are mainly based on differential privacy (DP) technique [7]–[10] and secure multiparty computing (SMC) technique [11]–[17]. Nevertheless, it is not easy to apply these schemes in practical systems. First, the DP technique relies on adding noises into original gradients to protect privacy, which gives rise to a challenging tradeoff between model accuracy and the level of privacy protection [5], [7], [8]. Second, the communication and computation overhead of SMC-based schemes are prohibitively expensive because the homomorphic encryption (HE) mechanism and the secret-sharing mechanism in SMC can considerably consume computation and communication resources, respectively. For example, some schemes are either computationally expensive or costly in communication, or limit the number of the participants, or require additional trust assumptions [14], [18], [19]. In many practical scenarios, especially mobile computing, the workload and complexity of the above methods far overstep the processing capacity of the devices, which makes them challenging to deploy.

To get rid of these weaknesses in FL, we propose a novel privacy-preserving FL framework based on a novel chained SMC technique, which is named as chain-PPFL. Our scheme can be illustrated in two aspects. First, participants are organized into the chained structure. For example, participants within the same area can form a chain because of their fast information exchange speed. For each formed chain, each participant adds the mask information to its gradients to obtain the output. The output of a parent participant is used by its descendent participant as the mask information to protect its gradients. Second, participants form a chain with a flexible distributed manner. For each round of iteration, there is a unique token with each chain that can be exclusively owned by each participant. The current participant owning the token will randomly select the next node from the rest neighbors without obtaining their mask information as the descendent node. Then, the current node passes its output to the descendent. The last participant sends its output as the final result, which contains the aggregation of gradients of all participants in the same chain, back to the server. The server generates a random number for each chain, which is transmitted to the first participant of each chain to kickoff the information exchange. Because the output of a single participant is masked with its precedent's output, the adversary (i.e., the curious node) cannot derive any privacy-sensitive details (i.e., gradients) from the participant's output. So chain-PPFL is equivalent to DP with $\epsilon \to 0$.

Our main contributions are as follows.

1) In the honest-but-curious setting, the proposed scheme can achieve privacy preservation without lowering the model accuracy, which is close to the baseline FL algorithm (i.e., FedAVG algorithm [1]) and superior to the DP-based FL algorithm [9].
2) The communication and computation complexities of our chain-PPFL scheme are much lower than a typical SMC-based scheme.
3) The architecture of the proposed chain-PPFL is relatively concise and easy to deploy on devices in some practical FL scenarios such as IoT.
4) We provide a prototype implementation of chain-PPFL. The Github link is https://github.com/ITSEG-MQ/Chain-PPFL.

The remainder of this article is organized as follows. In Section II, we first conclude with a discussion of related work. In Section III, we describe the architecture of chain-PPFL and implementation in detail. In Section IV, we provide a comparative analysis with other schemes, and in Section V, we give comparison experiments and discuss the experimental results. Finally, Section VI concludes the results in this article.

II. Related Work

Federal learning is increasingly applied to privacy-sensitive multiparty-collaborative environments, such as Google input tools, smart cognitive system, human emotion recognition, and more [20]–[24]. In order to coordinate various distributed computing nodes, model-related information needs to be transmitted [1], which requires privacy preservation for exchanged data (e.g., gradients) to prevent sensitive information in raw data from leaking through backward inference of publicly shared information [4]–[6], [25]. Currently, privacy preservation in FL is mainly implemented based on two techniques: 1) DP and 2) SMC.

A. Differential Privacy

For adversary to data/model analysis in FL, DP is a common privacy preservation method to quantify and limit leaking sensitive data [7]–[10]. It uses a randomized mechanism (such as introducing certain random subsampling or adding random noises) to distort the input or output of user processing so that the results of user processing can counter privacy analysis in a certain extent (i.e., privacy sensitivity). A common paradigm for approximating a deterministic real-valued function $f : D \to \mathbb{R}$ with a differentially private mechanism is via additive noise calibrated to f's sensitivity $\mathcal{S}_f$, which is defined as the maximum of the absolute distance $|f(d) - f(d')|$ where d and d' are adjacent inputs. For instance, the Laplace mechanism is defined by $f(d') \triangleq f(d) + \mathcal{L}(0, \mathcal{S}_f^2/\epsilon)$. $\mathcal{L}(0, \mathcal{S}_f^2/\epsilon)$ is the added noise with Laplace distribution to realise ϵ-DP. In addition to the Laplace mechanism, the Gaussian mechanism is often used. In DP-based FL, as the value of ϵ decreases, the added noise value becomes larger, the level of privacy protection increases, and at the same time, the training accuracy of the model decreases due to increased noise, and *vice versa*.

However, Zhu *et al.* [5] proposed the deep-leakage-from-gradients (DLGs) method, improved by Zhao *et al.* [6], and it experimentally showed that when the variance of noises added through the DP algorithm is less than 10^{-4}, the shared noisy gradients will leak private information by the DLG method. The DLG method will fail only when the variance is larger than 10^{-3}, but the added noises will significantly reduce the accuracy of the training model. In contrast, our proposed scheme can preserve privacy without impairing model accuracy.

B. Secure Multiparty Computing

Leveraging SMC to achieve secure aggregation of local model updates is a promising technique to protect privacy in FL. At present, there are various approaches, such as HE, secret sharing, and pairwise masking, which are briefly discussed as follows.

1) Homomorphic Encryption: HE schemes impose complicated mathematical operations on ciphertexts without decryption [26]–[31]. Because it does not directly use the plaintext during computing, HE is considered as an ideal method to realize the SMC protocol [13].

In the federated settings, Hardy and Aono *et al.* described several privacy-preserving solutions based on the additive HE (AHE) schemes [11], [12], [18]. Since there is no obfuscating/distorting operation in HE, the accuracy of the training model is almost not impaired. However, HE-based schemes confront the following challenges. First, HE algorithms, even if relatively simple AHE algorithms, demand extensive computational resources to scale or specific hardware conditions. Second, the massive parallelization is unavoidable to achieve real-time throughput on practical problems (some tasks may not be parallelizable) [32]. Third, some HE-based solutions may require a trusted third party to complete the encryption/decryption operation while the availability of a trusted noncolluding party is not standard in FL [13]. In comparison, our design is of light computation load and does not rely on a trusted third party.

2) Secret Sharing: Secret sharing refers to cryptographic methods for taking a secret, breaking it up into multiple shares, and distributing the shares among multiple parties so that only when the parties bring together their respective shares can the secret be reconstructed [33]. To secure aggregation of gradients updated by users, some people proposed several privacy-preserving and secure aggregation methods based on the secret-sharing mechanism [14]–[17].

TABLE I
LIST OF SYMBOLS

Symbols	Meaning
n	the total number of data-samples
$\mathcal{P}_d$	a partition over n data-samples
D	the number of data-partitions by $\mathcal{P}_d$
n_d	the number of data-samples in each data-partition
$K,\ k$	the number of users and index
C, m	a fraction rate over K and m users in each fraction
S_t	a random set of m users
B	the local minibatch size
E	the number of local epochs
η	the learning rate
t	training round
w_G^t	the global model parameters in round t
w_i^{t*}	the local optimal parameters of user i in round t
$\nabla\ell(w;b)$	gradient in each batch computed locally by user
$\mathbf{PRN}^t$	a random number generated by the aggregation node
U_0^t	the information sent by the aggregation node
$token_0^t$	the token constructed by the aggregation node
n_p	a counter which counts the number of participants
$Time\_limit$	a countdown timer for controlling duration of round
U_i^t	the information sent by user i
$token_i^t$	the token constructed by user i
$\mathcal{L}(0, \mathcal{S}_f^2/\epsilon)$	a noise with Laplace distribution
$\mathcal{S}_f^2$	the privacy sensitivity of DP
ϵ	the privacy budget of DP

The above schemes are similar to ours. In our scheme, it is not necessary to perform extra operations (e.g., obfuscation or encryption) before transmitting masked information because our scheme uses the peer-to-peer encrypted secure transmission channel. Therefore, our method is more efficient and requires fewer resources.

3) Pairwise Masking: Pairwise masking has been discussed by [34]–[36]. In the protocols proposed by [34] and [36], pairs of clients use the Diffie–Hellman key exchange to agree on pairwise masks. However, the aggregation result is usually insufficient to provide reasonable privacy for individual users when the number of participating users $N \gg 1$, therefore, the AHE technique is introduced to protect each uploaded local value after masked [14]. Their schemes are comparable to the DP-based scheme but the recovery phase in their protocol is brittle [36], and extra overhead will be incurred due to AHE. Compared with them, our chain-PPFL scheme is more efficient at the cost of communication and computation.

III. Architecture and Method

In this section, we introduce chain-PPFL in detail, which can implement the aggregation of model parameters in FL and simultaneously preserve the private information of the participants without compromising model accuracy. Table I presents the list of symbols used in this article and their meanings for easy readability.

A. Problem Preliminaries

FL typically aims to minimize the empirical risk over heterogeneous data distributed across multiple devices [37]. Before the start of training every iteration, each participating user gets the global model parameters from the aggregation node (or the server). Then, the user uses the global model parameters to train a local model with the local data set and sends new local model parameters as the local update to the aggregation node. The aggregation node computes new global model parameters by aggregating the local updates from the selected users. The process loops until the end condition is met.

Algorithm 1 FederatedAveraging Algorithm

```
Aggregation Node:
  initialize $w_G^1$    // Global Model Parameters
  FOR round $t = 1, 2, \ldots$ DO
    $m \leftarrow \max(C \times K, 1)$
    $S_t \leftarrow$ (random set of $m$ users)
    FOR each user $k \in S_t$ DO IN PARALLEL
      $w_k^{t*} \leftarrow \mathrm{UserUpdate}(k, w_G^t)$
        // Local Model Parameters Update
      $w_G^{t+1} \leftarrow \frac{1}{m}\sum_{k=1}^{m} w_k^{t*}$

UserUpdate$(k, w_G^t)$    // Run on user k
  $\mathcal{B} \leftarrow$ (select batches of size)
  FOR local epochs $i$ from 1 to $E$ DO
    FOR batch $b \in \mathcal{B}$ DO
      $w \leftarrow w - \eta\nabla\ell(w; b)$
  upload $w$ to Aggregation Node
```

In a federated setting, the optimization objective is

$$\min_{w\in\mathbb{R}^d} f(w), \quad \text{where } f(w) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(w). \tag{1}$$

If there are D data partitions selected with $n_d = |\mathcal{P}_d|$, where $\mathcal{P}_d$ is a partition over n data samples. If the partition $\mathcal{P}_d$ was uniformly set at random, thus

$$\text{let} \quad F_d(w) = \frac{1}{n_d}\sum_{i\in\mathcal{P}_d} f_i(w)$$
$$\text{then} \quad f(w) = \sum_{d=1}^{D}\frac{n_d}{n}F_d(w). \tag{2}$$

FedAVG [1] is widely used as a classic FL algorithm, and this article also uses it as the baseline algorithm. Here, we assume that there are one aggregation node and K users, and each user owns the fixed local data set by himself.

In the FedAVG algorithm [1], K users are indexed by k; B is the local minibatch size; E is the number of local epochs; and η is the learning rate. The details of the FedAVG algorithm are given in Algorithm 1.

In Algorithm 1, the work of the aggregation node includes distributing the global model parameters and aggregating the local updates from m users. Each user completes the training task based on the global model parameters w_G^t over the local data sets and uploads the local model update in the current round. The user uses the minibatch SGD (B batches) algorithm in local. The goal of the participating user i in round t is to find local optimal parameters w_i^{t*} that minimize the function $f(w_i^t)$, i.e.,

$$w_i^{t*} = \arg\min_{w_i^t} f(w_i^t). \tag{3}$$

Actually in FedAVG, the server selects K participants and computes the averaging aggregation as global weights [25]

$$w_G^{t+1} = \frac{1}{K}\sum_{i=1}^{K} w_i^{t*}. \tag{4}$$

B. Proposed Architecture

Our proposed chain-PPFL uses a novel chained-communication mechanism that enables masked information to be transferred between participants with a serial chain frame. In the baseline federated setting with a center server, shown in Fig. 1, the work of the server mainly includes: 1) broadcasting the list of participating users before training; 2) responsible for generating the random numbers as the masks; 3) constructing the token; 4) averaging the aggregation results extracted from the token received from the last user; and 5) distributing the global model parameters back to users.

The participating clients can create their lists of neighbor users based on the user list received from the server. In different network environments, we can leverage different methods to establish corresponding neighbor lists. For example, we can use the flood method to build the neighbor list in LAN environments or we can use the method proposed by Vela *et al.* to build the neighbor list in cellular networks [38]. In the P2P network, we can use the method proposed by Zhao *et al.* to build the neighbor list [39]. Since this is not the core issue discussed in this article, we assume that each participating client has built its neighbor list before training. For constructing a chained communication structure in each round, each participant (including the server) selects a neighbor randomly as the next node from its neighbor list. The establishment of the chained communication structure is introduced in detail in Section III-C.

Each round, the server generates a unique token for each chain that can be exclusively owned by each user and transferred serially by the chained communication mechanism between users. Our scheme leverages the single-masking mechanism to preserve privacy. The current participant owning the token will randomly select the next node from the rest neighbors without obtaining their mask information as the descendent node. Then, the current participant sends the updated token to the descendent node. The last participant sends its updated token (involving the aggregation of model parameters of all participants in the same chain) back to the server. Because a single participant uses its precedent's output to mask its output in the chained aggregating process, the adversary (i.e., the curious node) cannot derive any privacy-sensitive information from the participant's output without collusion.

Fig. 2 shows that chain-PPFL can also be used in the edge-computing scenario discussed by Liu *et al.* [40]. Multiple edge servers perform the partial model aggregation, and the center server (or the cloud server) aggregates the global model updates from edge servers. Usually, in the edge cloud, the network latency between the participants is so small that they can exchange information quickly in a chain. Therefore,

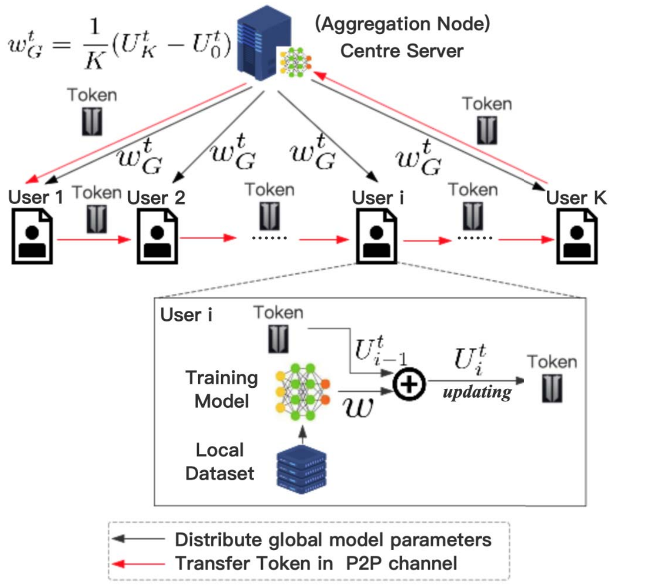


Fig. 1. Architecture of chain-PPFL with the center server.

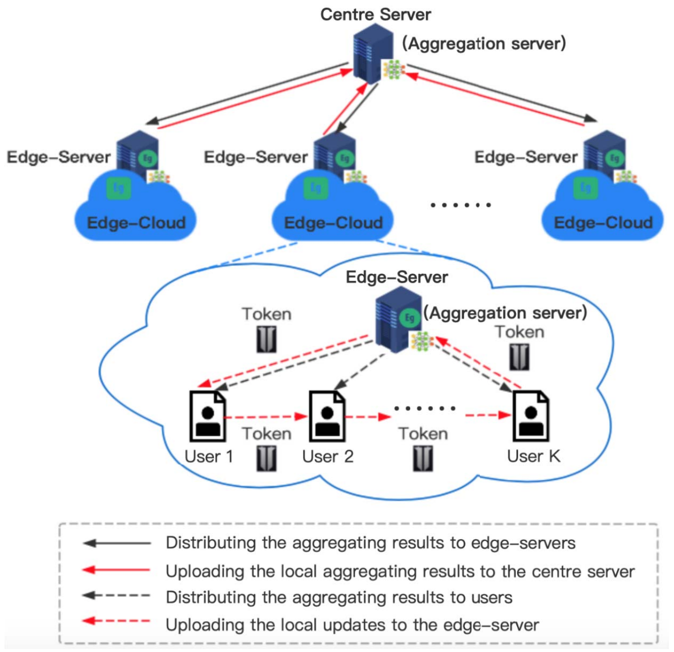

Centre Server
(Aggregation server)
Edge–Server
Edge–Cloud
Edge–Server
(Aggregation server)
Token
User 1
User 2
User K
Distributing the aggregating results to edge–servers
Uploading the local aggregating results to the centre server
Distributing the aggregating results to users
Uploading the local updates to the edge–server

Fig. 2. Architecture of chain-PPFL in edge-computing scenario.

Chain-PPFL can achieve a better tradeoff between communication and privacy preservation.

C. Chain-PPFL Algorithm

In Section II, we have discussed the challenges of privacy protection for exchanging information. In this article, we only consider the problem of privacy preserving among all participants and assume that all nodes are trust-but-curious, and the transmission channel is reliable and secure. The chained procedure of chain-PPFL in round i is as follows.

Step 1:

Aggregation Node: The aggregation node broadcasts the global w_G^t to all users and uses the pseudorandom generator to generate a random number PRN^t ($\text{PRN}^t \in \mathbb{R}^d$, d is the dimension of w) as a mask, and let $U_0^t = \text{PRN}^t$. Next, the aggregation node constructs $\text{token}_0^t(t, U_0^t, n_p, Time\_limit)$, where n_p is a counter that counts the number of participants and $Time\_limit$ is a countdown timer that controls duration of each round. Due to the assumption of this article, nodes need not check the token. The server sends the token to the first participant (named user 1) selected randomly from the neighbor users in the current round.

Step 2:

Participant 1: User 1 is the first participant of round t and the local optimal parameters w_1^{t*} is its local update. The training process of local model parameters is the same as FedAVG. Next, user 1 computes

$$U_1^t = w_1^{t*} + U_0^t \tag{5}$$

and transports the masked result U_1^t by updating token_1^t to the next participant, which is selected randomly from the neighbor users by user 1.

Participant i: User i computes the local update w_i^{t*} and uses the output of user $i-1$ as the mask to compute

$$U_i^t = w_i^{t*} + U_{i-1}^t = \sum_{j=1}^{i} w_j^{t*} + U_0^t \tag{6}$$

and transports the result U_i^t by updating token_i^t to the next participant (named user $i+1$), which is selected randomly from its neighbor users.

Participant K: User K computes the local update w_K^{t*} and uses the output of user $(K-1)$ as the mask to compute

$$U_K^t = w_K^{t*} + U_{K-1}^t = \sum_{j=1}^{K} w_j^{t*} + U_0^t. \tag{7}$$

Assuming that at this time $Time\_limit \le 0$, then user K just is the final node, so user K should transport the result U_K^t by updating token_K^t to the aggregation node.

Step 3:

Aggregation Node: The aggregation node receives Token_K^t from the Kth participant and computes the global weights of the next round

$$w_G^{t+1} = \frac{1}{K}\left(U_K^t - U_0^t\right) = \frac{1}{K}\sum_{i=1}^{K} w_i^{t*}. \tag{8}$$

Go to step 1.

The details of the proposed chain-PPFL algorithm are given in Algorithm 2.

IV. Analysis and Comparison

In this section, we will analyze chain-PPFL in the following three aspects: 1) training accuracy; 2) the level of privacy preservation; and 3) latency. Based on the analysis, we compare FedAVG with chain-PPFL, DP-based FL, and AHE-based FL and list the detailed analysis in Table II.

Algorithm 2 Chain-PPFL Algorithm

```
Aggregation Node:
  initialize and broadcast $w_G^1$   // Global Model Parameters
  FOR round $t = 1, 2, \ldots$ DO
    $U_0^t \leftarrow \mathbf{PRN}^t$   // by the pseudo-random generator
    $n_p \leftarrow 0$   // Count the number of participating users
    create $token_0^t(t, U_0^t, n_p, Time\_limit)$
    User 1 ← (randomly select from the set of users)
    send $Token_0^t$ to User 1
    wait UNTIL receive $Token_K^t$ from the final User $K$
    $w_G^{t+1} \leftarrow \frac{1}{n_p}(U_K^t - U_0^t)$

UserUpdate($Token_{i-1}^t$)   // Run on User $i$
  $\mathcal{B} \leftarrow$ (select batches of size)
  FOR local epochs $i$ from 1 to $E$ DO
    FOR batch $b \in \mathcal{B}$ DO
      $w \leftarrow w - \eta\nabla\ell(w; b)$
  $U_i^t \leftarrow w + U_{i-1}^t$
  $n_p \leftarrow n_p + 1$
  refresh $token_i^t(t, U_i^t, n_p, Time\_limit)$
  IF $time\_limit$ == true THEN
    send $Token_i^t$ to the Aggregation Node
  ELSE
    User $i+1$ ← (randomly select from Neighbor List)
    send $Token_i^t$ to User $i+1$
```

TABLE II
Comparison With Other Methods

	Train accuracy VS FedAVG	Level of privacy preservation	Latency VS FedAVG
DP-based FL	decreasing	weak	same
AHE-based FL[14]	same	strong	heavily large
Chain-PPFL	same	strong in group	relatively large

A. Analysis of Training Accuracy

Each round in FedAVG, the server selects a fragment of users in random and averages the aggregation of local updates into the global model parameters w_G^t as shown in (5). While in chain-PPFL, the aggregation node gets the aggregating model parameters from the final user and averages it, as shown in (9). Due to the randomness of selecting the next participant from the neighbors in chain-PPFL, if using the appropriate $Time\_limit$, the accuracy of the training model using our method is the same as the value given by FedAVG. Since $Time\_limit$ is related to computation and communication delays, we will discuss the setting of $Time\_limit$ in Section IV-C. Most methods based on SMC do not modify the core algorithm of FL and only provide a secure computation environment, so the accuracy of the training model is close to the value given by FedAVG on the whole. However, in the algorithm based on DP and pairwise masking techniques, training accuracy has decreased to varying degrees, and the reducing degree of accuracy depends on the value of ϵ-DP. Actually, the higher the added noise, the greater the loss of accuracy, which has been proven experimentally by [5] and [6].

B. Analysis of the Level of Privacy Preservation

Privacy preservation of raw data in FL is acceptable but shared gradients among the participants still possibly leak the sensitive information or even the raw training samples. The state-of-the-art research work focuses on the privacy preservation of exchanged data related to the training model between collaborative participants. Next, we will analyze the privacy preservation and security of some main schemes.

In baseline FL, the data of the local updates are the weights of the training model $w_i^{t^*}$ as formula (4). In FL with DP (leveraging Laplace mechanism), the uploaded noisy data are $w_i^{t^*} + \mathcal{L}(0, \mathcal{S}_f^2/\epsilon)$ [9], where $\mathcal{L}(0, \mathcal{S}_f^2/\epsilon)$ is a noise with Laplace distribution, $\mathcal{S}_f^2$ is the privacy sensitivity, and ϵ is the privacy budget. In [5], it is shown that the public-shared information in baseline FL and FL with DP has been proven experimentally to have high the risks of leaking privacy. In chain-PPFL, the exchanged data between the adjacent nodes are U_i^t, i.e., $U_i^t = \sum_{j=1}^{i} w_j^{t^*} + \text{PRN}^t$, where PRN^t is a random constant given by the aggregation node. Our method is equivalent to the DP-based FL with $\epsilon \to 0$ without collusion and makes the adversary (i.e., the curious node) almost impossible to derive any privacy-sensitive details from the information received. Where the aggregation node gets nothing but the sum of the local updates, it is required that there are at least two nodes, which participated in the collaborative training each round. That can be achieved by choosing a suitable *Time_limit*. Leveraging the AHE-based scheme, the participants will upload the crypttext of the local weights $w_i^{t^*}$, and in some more complicated methods based on HE, the ciphertext of the masked local weights will be uploaded. The degree of privacy preservation depends on the security of the encryption algorithm itself.

C. Analysis of Latency

In federated setting, the latency in round i includes three parts: 1) T_{local}^i, the latency for computing local model parameters in distributing participants; 2) T_{global}^i, the latency for computing the global model parameters in the center node; 3) $T_{\text{comm,up}}^i$, the communication latency of uploading the local updates. Here, we do not consider the time it takes to broadcast the global model to local nodes because it is the same to all methods compared in this article. To facilitate inference, we assume that the baselines T_{local}^i and $T_{\text{comm,up}}^i$ are the same for all participating users. Users participating in baseline FL upload in parallel the local updates to the server, so the total latency each round in baseline FL $T_{\text{total}}^i = T_{\text{local}}^i + T_{\text{global}}^i + T_{\text{comm,up}}^i$.

In the FL with DP, the latency for computing local model parameters $T_{\text{local,DP}}^i$ is slightly larger than its in baseline FL since mainly implementing privacy preserving by adding noise. So the total latency $T_{\text{total,DP}}^i = T_{\text{local,DP}}^i + T_{\text{global}}^i + T_{\text{comm,up}}^i$.

As for the method based on HE, the latency in the local node, $T_{\text{local,HE}}^i$, mainly includes two parts: 1) T_{local}^i, the latency for computing local model parameters and 2) T_{encrypt}^i, the latency for implementing HE of local model parameters, i.e., $T_{\text{local,HE}}^i = T_{\text{local}}^i + T_{\text{encrypt}}^i$. Here, we assume that the center node completes the decryption of the aggregation sum, so $T_{\text{global,HE}}^i = T_{\text{global}}^i + T_{\text{decrypt}}^i$. Therefore, the total latency $T_{\text{total,HE}}^i = T_{\text{local,HE}}^i + T_{\text{global,HE}}^i + T_{\text{comm,up}}^i$, regardless of the time for key generation and key distribution.

In chain-PPFL, shown as Fig. 1, the communication latency $T_{\text{comm},o}^i$ includes two parts: 1) $T_{\text{comm},sa}^i$, the communication latency between local nodes and 2)$T_{\text{comm,up}}^i$, the communication latency of uploading the local model parameters that is the same as other methods. Here, to facilitate inference, we assume that the communication latency between a local participant and its neighbor is the same, named as $T_{\text{comm},nb}^i$, so

$$\begin{aligned} T_{\text{comm},o}^i &= T_{\text{comm},sa}^i + T_{\text{comm,up}}^i \\ &= (K-1) \times T_{\text{comm},nb}^i + T_{\text{comm,up}}^i \end{aligned} \tag{9}$$

where K is the number of users participating in the model training each round. In addition, the aggregation node needs not to compute the sum, which is aggregated serially by participants, but also to average the serial aggregating result, so the latency for computing local model parameters $T_{\text{local},o}^i$ is slightly larger than its in baseline FL and the latency for computing the global model parameters $T_{\text{global},o}^i$ is smaller than T_{global}^i. Therefore, the total latency of our scheme $T_{\text{total},o}^i = T_{\text{local}}^i + T_{\text{global},o}^i + T_{\text{comm},o}^i$. In the aspect of computing time, because of assuming the baseline time of the training model is the same, there is

$$T_{\text{local}}^i < T_{\text{local},o}^i < T_{\text{local,DP}}^i < T_{\text{local,HE}}^i \tag{10}$$

and

$$T_{\text{global},o}^i < T_{\text{global}}^i < T_{\text{global,HE}}^i. \tag{11}$$

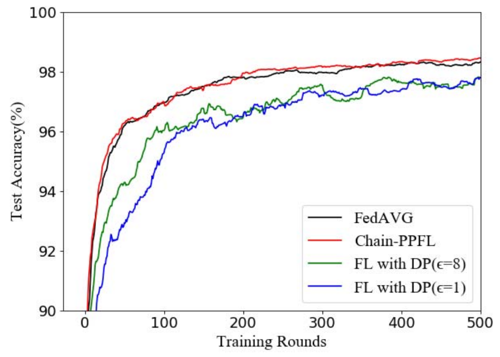
In the aspect of the communication latency, when $T_{\text{comm},sa}^i \ll T_{\text{comm,up}}^i$, that is to say, the network delay between the local nodes and the aggregation node is larger, then $T_{\text{total},o}^i \approx T_{\text{total}}^i$. Otherwise, when $T_{\text{comm},sa}^i \ll T_{\text{local}}^i$ or $T_{\text{comm},o}^i \ll T_{\text{local}}^i$, then $T_{\text{total},o}^i \approx T_{\text{total}}^i$. So we may see that the key factor that affect the efficiency of our algorithm is the communication latency between any local participant and its neighbor $T_{\text{comm},nb}^i$. Moreover, due to the security of our method, *Time_limit* in Token_i^t should be longer than $[T_{\text{local}}^i + T_{\text{comm},nb}^i + T_{\text{comm,up}}^i]$. There are n users to participate in training when *Time_limit* is greater than $[T_{\text{local}}^i + (n-1) \times T_{\text{comm},nb}^i + T_{\text{comm,up}}^i]$.

The proposed chain-PPFL is more suitable for scenarios with large network bandwidth and small network delays, such as LANs or high-speed P2P networks. Also, the chain-PPFL framework is suitable for entire decentralized scenarios without the center node, in which the aggregation process of local training results can be completed serially in a chain between nodes. This direction is one of our future works.

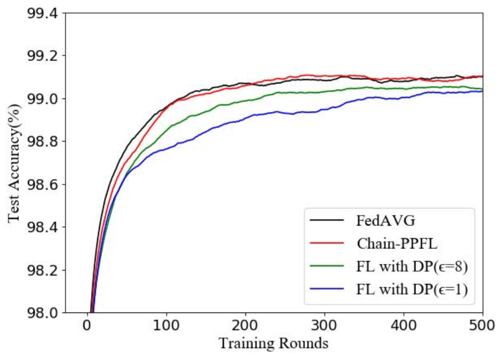
V. Experiment

A. Experiment Setup

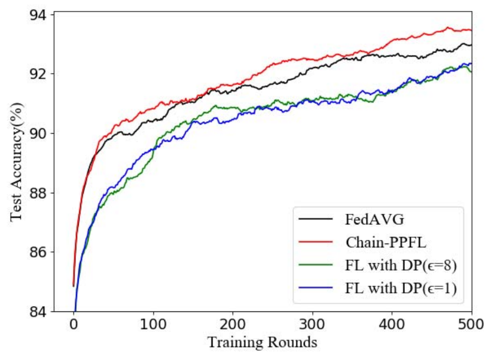
We simulate and compare the training accuracy of three schemes, including the proposed chain-PPFL algorithm, FedAVG algorithm, and the DP-based algorithm with the Laplace mechanism in [9] experimentally.



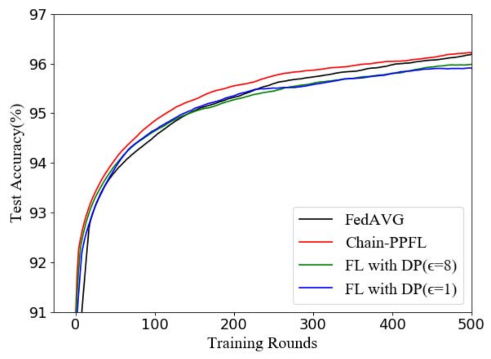
(a)



(b)



(c)



(d)

Fig. 3. Comparison of test accuracy within 500 rounds. (a) MNIST CNN NonIID. (b) MNIST CNN IID. (c) MNIST MLP NonIID. (d) MNIST MLP IID.

In accordance to [1] and [7], the federated setting of our experimental scenario is as follows.

1) Number of users $K = 100$.
2) User fraction rate $C = 0.1$.
3) Local minibatch size $B = 10$.
4) Number of local epochs $E = 5$.
5) Learning rate $\eta = 0.1$.
6) DP-based FL, respectively, fixes $\epsilon = 8$ and $\epsilon = 1$.

In the comparison test of training accuracy, we use the public data set MNIST digit recognition. For the comprehensive federated setting, we use two ways of partitioning the MNIST data over users [1]: 1) IID, the data are shuffled and partitioned into 100 users each receiving 600 examples and 2) NonIID, the data are sorted by digit labels and divided into 200 shards of size 300. We assign each of 100 users two shards. Besides, we use the two training models (similar to [1] and [41]): 1) MLP, a simple multilayer-perception with two-hidden layers with 200 units each using ReLU activations and 2) CNN, a CNN with two 5×5 convolution layers, a fully connected layer with 512 units and ReLU activation, and a final softmax output layer. We use the DLG method given by [5] to verify the analysis of leak defence experimentally. In the comparison test of leak defence, we use the public data set CIFAR-100 image classification and the L-BFGS model with learning rate 1 (the same in [5]). We use PyTorch [42] as the experiment platform. The simulation experiments run in the computer with the processor of Intel Core i5-4570 CPU @ 3.20 GHz, installed RAM 8.0 GB and without GPU.

B. Experimental Results

The comparative results on the test accuracy of training models are shown in Fig. 3. The DP-based FL schemes preserve privacy by adding noises into original gradients. In order to improve the level of privacy protection, it is necessary to add noises with a larger variance, i.e., increasing the value of ϵ. However, noises with a large variance have a more significant negative impact on training accuracy. From Fig. 3, the experimental results show that the accuracy of models trained by chain-PPFL is close to that of FedAVG and outperforms DP-based FL by 0.1%–2%, especially for models on MNIST with NonIID partition. For the MLP model on MNIST with IID partition, Fig. 3(d) shows that the accuracy of chain-PPFL is close to FedAVG and slightly better than that of the DP-based FL by 0.2% ($\epsilon = 8$) to 0.4% ($\epsilon = 1$). Since the mask information can be removed entirely, the training accuracy given by chain-PPFL is very close to FedAVG on the whole and better than that of the DP-based FL algorithm from the experimental evaluation results as shown in Fig. 3. In addition, the smaller the ϵ value is, the larger the variance of added noises is, and the more significant the accuracy deteriorates.

On the other hand, due to the added noises during the process of model training, DP-based FL schemes unavoidably increase the number of iteration rounds significantly, and results in a slower convergence rate, as shown in Table III. In comparison, the number of consumed training rounds to achieve about the same level of accuracy by chain-PPFL is roughly the same as of the FedAVG algorithm, which is much smaller than the number of rounds consumed by the DP-based FL algorithms.

In the comparison test of leak defence, we use L-BFGS to match gradients from all parameters of the training model

TABLE III
COMPARISON OF TRAINING ROUNDS (MEAN)

Model & Accuracy on Data Distribution	Fed-AVG	Chain-PPFL	FL with DP($\epsilon = 8$)	FL with DP($\epsilon = 1$)
CNN on Non-IID (98%)	1.2×	1.2×	1.5×	2.5×
CNN on IID (99%)	1.0×	1.0×	1.8×	3.0×
MLP on Non-IID (92%)	1.2×	1.2×	2.0×	2.0×
MLP on IID (95%)	1.0×	1.0×	1.2×	1.2×
				(magnitude: 10^2 rounds)

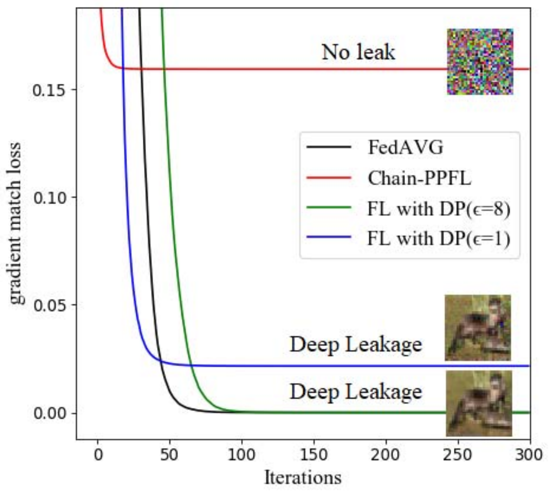


Fig. 4. Effectiveness of preventing privacy leakage.

in FL, and the experiment uses randomly initialized weights. The gradient matching loss reflects the degree of discrimination for the raw samples. The smaller the value of gradients matching loss, the more information leaked. When the value of the matching loss function is larger than 0.15, no information is leaked. Fig. 4 gives the experimental results on the effectiveness of leak defence. From Fig. 4, we observe that both FedAVG and the DP-based FL schemes cannot prevent the leakage of sensitive information, but our method successfully achieves privacy preservation.

In summary, the results of all the above experiments demonstrate that chain PPFL can achieve much better accuracy than that of DP-based FL. Due to its ability to perfectly cancel the added noises with the masked information, its performance is even very close to that of the original FedAVG algorithm, however, which has not taken privacy preservation into account.

VI. CONCLUSION

In this article, we have proposed a privacy-preserving FL framework based on the chained SMC technique: chain-PPFL. The results of analyses and experiments show that the chain-PPFL scheme can achieve a better level of privacy preservation in this case that the loss of accuracy and the computational complexity of our scheme are almost close to the FedAVG algorithm. Moreover, the architecture of chain-PPFL is relatively concise and suitable for some practical scenarios with a better communication environment, such as edge cloud, intelligent IoT, or smart city. In the future, our research work will focus on the improvement of chain-PPFL and application in decentralized FL settings.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016. [Online]. Available: arXiv:1602.05629.

[2] 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.

[3] 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.

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

[5] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems 32*. Vancouver, BC, Canada: Curran Assoc., Inc., 2019, pp. 14774–14784.

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

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

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

[9] 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)*, San Francisco, CA, USA, 2020, pp. 466–479.

[10] T. Wang, Y. Mei, W. Jia, X. Zheng, G. Wang, and M. Xie, "Edge-based differential privacy computing for sensor–cloud systems," *J. Parallel Distrib. Comput.*, vol. 136, pp. 75–85, Feb. 2020.

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

[12] R. Nock *et al.*, "Entity resolution and federated learning get a federated resolution," 2018. [Online]. Available: arXiv:1803.04035.

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

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

[15] K. Mandal, G. Gong, and C. Liu, "Nike-based fast privacy-preserving high-dimensional data aggregation for mobile devices," Dept. Elect. Comput. Eng., Univ. Waterloo, Waterloo, ON, Canada, Rep. CACR2018–10, 2018.

[16] 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.

[17] Q. Li and M. G. Christensen, "A privacy-preserving asynchronous averaging algorithm based on shamir's secret sharing," in *Proc. IEEE 27th Eur. Signal Process. Conf. (EUSIPCO)*, A Coruña, Spain, 2019, pp. 1–5.

[18] 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.

[19] J. Yuan and S. Yu, "Privacy preserving back-propagation neural network learning made practical with cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 212–221, Jan. 2014.

[20] K. Umapavankumar, S. Srinivasu, S. S. N. Rao, and S. T. Rao, "Machine learning usage in Facebook, Twitter and Google along with the other tools," in *Emerging Research in Data Engineering Systems and Computer Communications*. Singapore: Springer, 2020, pp. 465–471.

[21] S. Deep, X. Zheng, A. Jolfaei, D. Yu, P. Ostovari, and A. Kashif Bashir, "A survey of security and privacy issues in the Internet of Things from the layered context," *Trans. Emerg. Telecommun. Technol.*, Mar. 2020.

[22] A. K. Sangaiah, J. S. A. Dhanaraj, P. Mohandas, and A. Castiglione, "Cognitive IoT system with intelligence techniques in sustainable computing environment," *Comput. Commun.*, vol. 154, pp. 347–360, Mar. 2020.

[23] X. Xu, K. Lin, H. Lu, L. Gao, and H. T. Shen, "Correlated features synthesis and alignment for zero-shot cross-modal retrieval," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1419–1428.

[24] H. Lu, M. Wang, and A. K. Sangaiah, "Human emotion recognition using an EEG cloud computing platform," *Mobile Netw. Appl.*, vol. 25, no. 3, pp. 1023–1032, 2020.

[25] 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.

[26] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Proc. Annu. Cryptol. Conf.*, 2012, pp. 868–886.

[27] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," IACR Cryptol. ePrint Archive, Lyon, France, Rep. 2012/144, 2012.

[28] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 1–36, 2014.

[29] J.-S. Coron, T. Lepoint, and M. Tibouchi, "Scale-invariant fully homomorphic encryption over the integers," in *Proc. 17th Int. Workshop Public Key Cryptogr.*, 2014, pp. 311–328.

[30] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.

[31] Z. Min, G. Yang, A. K. Sangaiah, S. Bai, and G. Liu, "A privacy protection-oriented parallel fully homomorphic encryption algorithm in cyber physical systems," *EURASIP J. Wireless Commun. Netw.*, vol. 6958, no. 1, p. 15, 2019.

[32] 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.

[33] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[34] G. Ács and C. Castelluccia, "I have a DREAM! (DiffeRentially privatE smArt metering)," in *Proc. Int. Workshop Inf. Hiding*, 2011, pp. 118–132.

[35] T. Elahi, G. Danezis, and I. Goldberg, "PrivEx: Private collection of traffic statistics for anonymous communication networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2014, pp. 1068–1079.

[36] S. Goryczka and L. Xiong, "A comprehensive comparison of multiparty secure additions with differential privacy," *IEEE Trans. Depend. Secure Comput.*, vol. 14, no. 5, pp. 463–477, Sep./Oct. 2017.

[37] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smithy, "FedDANE: A federated newton-type method," in *Proc. IEEE 53rd Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, CA, USA, 2019, pp. 1227–1231.

[38] M. Vela, N. Saxena, and M. Irizarry, "Efficient neighbor list creation for cellular networks," U.S. Patent 8 086 237, Dec. 2011.

[39] H. Zhao, C. Wang, Y. Zhu, and W. Lin, "P2P network based on neighbor-neighbor lists," *J. Phys. Conf. Series*, vol. 1168, no. 3, 2019, Art. no. 032072.

[40] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," 2019. [Online]. Available: arXiv:1905.06641.

[41] S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang, "Learning private neural language modeling with attentive aggregation," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, Budapest, Hungary, 2019, pp. 1–8.

[42] P. Adam *et al.*, "Automatic differentiation in pytorch," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 8026–8037.

Yong Li (Member, IEEE) received the M.S. degree from Jilin University, Changchun, China, in 2004, where he is currently pursuing the Ph.D. degree.

He is an Associate Professor with the Changchun University of Technology, Changchun. He is currently a Visiting Scholar with Macquarie University, Sydney, NSW, Australia. His research interests include federated learning, edge computing, information security, and privacy preserving.

Yipeng Zhou (Member, IEEE) received the bachelor's degree in 2006, and the M.S. and Ph.D. degrees from the Information Engineering Department, 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. His current research areas include multimedia systems, social multimedia, vehicular networks, Internet content distribution, and data mining.

Alireza Jolfaei (Senior Member, IEEE) received the Ph.D. degree in applied cryptography from Griffith University, Gold Coast, QLD, Australia, in 2016.

He is a Lecturer with the Department of Computing, Macquarie University, Sydney, NSW, Australia. His current research areas include cyber security, IoT security, human-in-the-loop CPS security, cryptography, AI, and machine learning for cyber security.

Dongjin Yu (Senior Member, IEEE) received the bachelor's and master's degrees in computer science and technology from Zhejiang University, Hangzhou, China, in 1993 and 2004, respectively, and the Doctoral degree in management from Zhejiang Gongshang University, Hangzhou, in 2010.

He is currently a Professor with Hangzhou Dianzi University, Hangzhou. He is the Director of Big Data Institute, and the Computer Software Institute, Hangzhou Dianzi University. His research efforts include service computing, data engineering, and intelligent software engineering.

Prof. Yu is also a member of Technical Committee of Software Engineering China Computer Federation (CCF) and the Technical Committee of Service Computing CCF. He is a senior member of CCF.

Gaochao Xu (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the College of Computer Science and Technology, Jilin University, Changchun, China, in 1988, 1991, and 1995, respectively.

He is currently a Professor and the Ph.D. Supervisor with the College of Computer Science and Technology, Jilin University. His main research interests include distributed system, grid computing, cloud computing, Internet of Things, information security, software testing, and software reliability assessment.

Xi Zheng (Member, IEEE) received the Ph.D. degree in software engineering from UT Austin, Austin, TX, USA, in 2015.

He is a Lecturer with the Department of Computing, Macquarie University, Sydney, NSW, Australia. His current research areas include software modeling, formal verification, security analysis, systematic testing for cyber–physical system, cloud computing, robotics and autonomous systems, real-time systems, and hybrid systems.