Toward Secure and Privacy-Preserving Distributed Deep Learning in Fog-Cloud Computing

Yiran Li, *Graduate Student Member, IEEE*, Hongwei Li, *Senior Member, IEEE*, Guowen Xu, *Graduate Student Member, IEEE*, Tao Xiang, *Member, IEEE*, Xiaoming Huang, *Member, IEEE*, and Rongxing Lu, *Senior Member, IEEE*

***Abstract*—Fog-cloud computing promises many new vertical service areas beyond simple data communication, storing, and processing. Among them, distributed deep learning (DDL) across fog-cloud computing environment is one of the most popular applications due to its high efficiency and scalability. Compared with the centralized deep learning, DDL can provide better privacy protection with training only on sharing parameters. Nevertheless, when DDL meets fog-cloud computing, it still faces two major security challenges: 1) how to protect users' privacy from being leaked to other internal participants in the training process and 2) how to guarantee users' identities from being forged by external adversaries. To combat them, several approaches have been proposed via various technologies. Nevertheless, those approaches suffer from drawbacks in terms of security, efficiency, and functionality, and cannot guarantee the legitimacy of participants' identities during training. In this article, we propose a secure and privacy-preserving DDL (SPDDL) for fog-cloud computing. Compared with the state-of-the-art works, our proposal achieves a better tradeoff between security, efficiency, and functionality. In addition, our SPDDL can guarantee the unforgeability of users' identities against external adversaries. Extensive experimental results indicate the practical feasibility and high efficiency of our SPDDL.**

***Index Terms*—Distributed deep learning (DDL), fog-cloud computing, identity verification, privacy preserving.**

Manuscript received March 30, 2020; revised June 13, 2020 and July 7, 2020; accepted July 24, 2020. Date of publication July 28, 2020; date of current version December 11, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802300 and Grant 2017YFB0802000; in part by the National Natural Science Foundation of China under Grant 61972454, Grant 61802051, Grant 61772121, and Grant 61728102; in part by the Sichuan Science and Technology Program under Grant 2020JDTD0007 and Grant 2020YFG0298; in part by the Peng Cheng Laboratory Project of Guangdong Province under Grant PCL2018KP004; and in part by the Guangxi Key Laboratory of Cryptography and Information Security under Grant GCIS201804. *(Corresponding author: Hongwei Li.)*

Yiran Li, Hongwei Li, and Guowen Xu are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: yiranli842@foxmail.com).

Tao Xiang is with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: txiang@cqu.edu.cn).

Xiaoming Huang is with the Technology Marketing Department, CETC Cyberspace Security Research Institute Company Ltd., Chengdu 610041, China (e-mail: apride@gmail.com).

Rongxing Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton NB E3B 5A3, Canada (e-mail: rlu1@unb.ca).

Digital Object Identifier 10.1109/JIOT.2020.3012480

I. Introduction

FOG-CLOUD computing is envisioned as a preferable complement of single-cloud computing [1], [2]. For supplying Internet-of-Things (IoT) services with high scalability and low latency, fog-cloud computing deploys a part of processing and storage to fog nodes, which are closer to "things." Based on the decentralized computing architecture, distributed deep learning (DDL) has been properly applied in IoT and supplied with diverse scenarios, such as autonomous vehicles [3], [4], smart grid [5], [6], e-health [7], [8], etc. For example, the DDL-based vehicle system can timely and precisely make complex driving decisions by deploying the unified neural networks to base stations (fog nodes) and the cloud server. Besides, a DDL-based smart grid can achieve electricity billing and fault detection with higher efficiency. Assuredly, DDL has obtained remarkable achievements in IoT and is expanding the application scenarios of IoT.

Tracing back to 2016, Google company formalized the concept of DDL (also called federated learning) for addressing the issue of data privacy with a distributed learning model. With the novel framework, DDL can achieve training model only by exchanging parameters between the cloud server and users, rather than the raw data. Despite this, DDL still has some intrinsic security issues, which may prevent the application of DDL in fog-cloud computing. State-of-the-art research [9] has demonstrated that in the DDL-based framework, the user's private information may be leaked to other internal participants through even a small partition of leaked parameters. For addressing this problem, many researches have been carried out, which are mainly based on differential privacy (DP) and encryption technology. The main idea of DP-based DDL is to add noise (Laplace or Gaussian noises) [10], [11] to the exchanged parameters for concealing the real information while achieving the secure aggregation for obtaining the global weights of the neural networks. Encryption technologies fall into two main categories: 1) secure multiparty computation (SMC) and 2) homomorphic encryption (HE). The SMC-based DDL [12], [13] aims to create approaches for users to jointly calculate a summation of their gradients while keeping data privacy through Yao's garbled circuit (GC) [14] or secret key sharing (such as the Shamir secret sharing (SS) [15]). Additionally, HE-based DDL [11], [13] utilizes fully HE (FHE) or additive HE (AHE) to construct the privacy-preserving protocol.

2327-4662 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

Nevertheless, current methods still face some challenges in security, efficiency, and functionality. First, although DP-based schemes can operate with high efficiency and low consumption, they must balance accuracy and privacy [16]. Second, SMC-based methods can guarantee security and provide more functionalities, however, multiple rounds of interaction lead to large communication overhead, which is a serious challenge for the resource-constrained devices in IoT. Third, FHE-based frameworks can simultaneously support addition and multiplication operations, however, the unacceptably huge consumptions of storage and computation make it unfeasible for practical applications. Recent solutions based on additive HE seem more efficient than FHE and SMC-based ones, nevertheless, utilizing a unified private key for all the users is unable to defend against the collusion, in which internal participants (some curious users and the cloud server) may collude with each other for obtaining other users' privacy. Meanwhile, they cannot be robust for users dropping out, which will cause the training process to restart and consume many more resources. Beyond the issues above, in the practical IoT environment, some external adversaries may forge the legitimate users' identities [17] to corrupt the DDL-based framework. Unfortunately, existing solutions have no consideration of any authentication mechanism to guarantee the unforgeability of users' identities during training.

In this article, we propose a secure and privacy-preserving DDL (SPDDL) for fog-cloud computing, where the threshold Paillier encryption [18] is utilized for encrypting the local gradients and a threshold signature [19] is applied to verify the users' identities. We demonstrate that our proposal achieves a preferable tradeoff between security, efficiency, and functionality than current works. Focusing on security, our scheme holds threshold encryption property and does not need to balance accuracy and privacy, both of which make our scheme more secure than AHE-based methods [20], [21] and DP-based methods [22]–[25]. Meanwhile, our scheme performs with higher accuracy than DP-based methods. As the same functionalities as the SMC-based method [12] can provide, our SPDDL supplies privacy protection against the collusion between multiple internal participants and robustness to users exiting during the training procedure. Note that less exchanges between participants and the optimal encryption mechanism allow our SPDDL to perform with much less communication and computation overhead. In addition, an authentication scheme is conducted to defend against external adversaries from forging users' identities, which supplies our SPDDL with a higher security level.

Compared with the previous conference version [26], we have improved the security level of our scheme, where we utilize a zero-knowledge proof-based signature scheme to verify the users' identities. Besides, we have applied our proposed SPDDL into fog-cloud computing. Additionally, a more detailed analysis of security and a more comprehensive evaluation of performance have been presented in this version. Specifically, the contributions of this article can be summarized as follows.

1) *Identity Verification:* A zero-knowledge proof-based threshold signature is utilized to construct the authentication scheme, which can ensure the unforgeablity of users' identities against the adaptive-chosen-message attack, and prohibit external adversaries from compromising our system for malicious attacks.
2) *Privacy Preservation:* A public-key system is built up based on the threshold Paillier encryption. With the multikey framework, each user's data privacy is guaranteed from being deduced by the cloud server or other users while working with low overhead of key management.
3) *Resistance Against Collusion:* SPDDL can tolerate the collusion among a certain amount of internal participants, including users and the cloud server while guaranteeing users' private information from leakage.
4) *Robustness for Users Dropping out:* A robust privacy-preserving protocol is designed for supporting users exiting in any phase of the training procedure, which can avoid redundant repetitions of the training process, for alleviating the communication and computation overhead.
5) *Provable Security and High Performance:* Comprehensive security analysis is provided for demonstrating the unforgeability of each user's identity against external adversaries, as well as the security against internal participants. Besides, purposeful simulations are operated in extensive experiments to prove SPDDL with preferable feasibility, efficiency, and security.

The remainder of this article is organized as follows. We state our research problem in Section II, and review some encryption primitives in Section III. In Section IV, we perform a detailed introduction to our scheme. Then, we achieve a comprehensive security analysis and analyze performance evaluation, respectively, in Sections V and VI. Next, we discuss related works in Section VII. Finally, in Section VIII, we draw our conclusions.

II. Problem Statement

In this section, we first review the DDL. Then, we outline an overview of our system. Finally, we introduce the threat model as well as our goals.

A. Distributed Deep Learning

1) Neural Network: In general, SPDDL can be applied to any type of neural network. For simplicity, we take the fully connected neural network (FCNN) as an example to introduce the concept of neural networks. As shown in Fig. 1, this model acts as a bottom-up pipeline, which is constituted by the input layer, hidden layer, and output layer. Meanwhile, adjacent two layers are connected by the vector of weights. We can define this model as a function: $F(X, \boldsymbol{W}) = \widetilde{Y}$ (X: input vector, $\boldsymbol{W}$: weight, and $\widetilde{Y}$: output vector). Given a training data set $D = \{X, Y\}$, the loss function could be defined as $\mathcal{L}(Y, \widetilde{Y}) = ||Y - \widetilde{Y}||^2$. For achieving the model training, we should minimize $\mathcal{L}(Y, \widetilde{Y})$ to obtain the optimal parameter of weight $\boldsymbol{W}$. This nonlinear optimization problem can be addressed by minibatch stochastic gradient descent (SGD) [11], [27], [28].

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

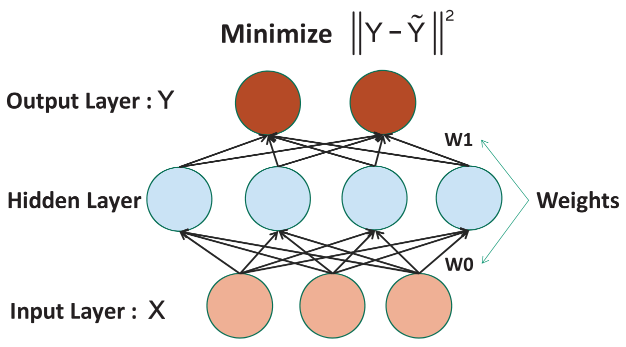

Fig. 1. Fully connected neural network.

2) *Stochastic Gradient Descent:* The entire data set consists of N pairs of training data, which can be defined as $D_N = \{D_i = (X_i, Y_i), i = 1, 2, \ldots, N\}$. In the jth iteration, some data pairs of (X_i, Y_i) will be randomly chosen from D_N to constitute the minibatch $D_j \in D_N$ for training the model. Then, the loss function is defined as follows:

$$\mathcal{L}_F(D_j, \boldsymbol{W_j}) = \frac{1}{|D_j|} \sum_{(X_i, Y_i) \in D_j} \mathcal{L}_F(X_i, Y_i, \boldsymbol{W_j})$$

where $\mathcal{L}_F(X_i, Y_i, \boldsymbol{W_j}) = \mathcal{L}(Y_i, F(X_i, \boldsymbol{W_j}))=||Y_i - \widetilde{Y}_i||^2$, and $|D_j|$ denotes the number of training pairs in D_j.

Then, for obtaining the optimal training model, the parameter of weight $\boldsymbol{W}$ will be adjusted as follows:

$$\boldsymbol{W}_{j+1} \leftarrow \boldsymbol{W}_j - \beta \nabla \mathcal{L}_F(D_j, \boldsymbol{W}_j)$$

where β denotes the learning rate, ∇ denotes the partial derivative of $\mathcal{L}_F$ with respect to $\boldsymbol{W}_j$ based on the data set D_j, $\boldsymbol{W}_j$ denotes the current weight, and $\boldsymbol{W}_{j+1}$ denotes the updated weight. The neural network performs the above-mentioned parameter updates iteratively until the preconvergence conditions are satisfied.

B. System Overview

As shown in Fig. 2, there are four entities in our application scenario, i.e., perceptual devices, fog nodes (considered as *users*), *the cloud server*, and trusted authority (TA). In brief, perceptual devices upload real-time information, while *the cloud server* cooperates with fog nodes to construct the fog-cloud computing structure, which makes complex decisions [29] and issues control instructions to perceptual devices. We specifically describe three entities applied in our scheme as follows.

1) *TA:* TA initializes our framework, generating public parameters and a public key, as well as some private keys. Besides, it broadcasts public parameters and the public key to both of *users* and *the cloud server*, while, respectively, sending each unique private key to each user through the secure tunnel based on the transport layer security (TLS) protocol [30].
2) *Users: Users* cooperate with *the cloud server* to execute the privacy-preserving training process, where they calculate local gradients with the data sent by perceptual devices and achieve the encryption of local gradients for further secure aggregation in *the cloud server*. Additionally, for verifying each user's identity, each user signs their local gradients and uploads the signatures.
3) *Cloud Server: The cloud server* aggregates these encrypted gradients, achieves the decryption for obtaining a summation of these gradients, and calculates the global parameters of weights. Besides, it executes the authentication for each user through verifying each user's signatures.

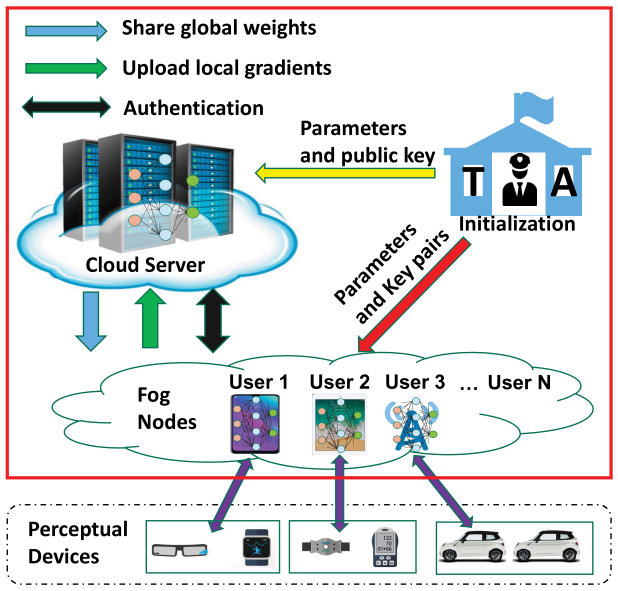

Fig. 2. System overview.

C. Threat Model and Goals

In our scenario, *users* and *the cloud server* are considered as two honest-but-curious entities [31], which means that both of them will strictly execute the protocol, but some curious ones of them may intend to infer other ones' private information. Besides, there exist some *external adversaries* trying to attack the system through forging users' identities to deceive the system, and they are considered impossible to access our system if they cannot obtain the legal identities. Additionally, we assume the TA is totally honest. That means it will strictly follow the scheme with no intention to collude with any other participants. Furthermore, a certain amount of internal participants (including *users* and *the cloud server*) are permitted to obtain the most offensive capabilities by colluding with each other.

The goals of our SPDDL are to protect users' private gradient information, even if multiple internal participants collude with each other, guarantee each user's identity unforgeable, and keep robustness to users' failure at any phase of the training procedure.

III. Cryptographic Primitives

In this section, we review the encryption scheme [18], which constructs our privacy-preserving protocol, and the signature scheme [19], which conducts our authentication framework.

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

A. Threshold Paillier Encryption

Although there are many encryption technologies, we utilize the (K, T)-threshold Paillier encryption [18] in our scheme because it has not only the additive homomorphic property but also the (K, T)-threshold property, both of which allow us to achieve the secure aggregation on users' plaintexts.

In this asymmetric cryptosystem, each user holds a unified public key Puk while the private key will be divided to K keys denoted as $(S_1, S_2, \dots, S_k)$, which are distributed to K users. With the public key Puk, each user's plaintext could be encrypted, but only the collaboration of at least T users can achieve the decryption. Considering there are K users and the ith user U_i holds a public key Puk and a private key S_i, we introduce the scheme in detail.

1) Key Generation: The public key is $Puk = n$, where $n = pq$ (n is a large positive integer, and p and q are two *primes*). For acquiring a private key S_i, we first randomly choose $\boldsymbol{x}_j$ (for $0 < j < T$) from $\{0, \dots, n * n' - 1\}$, where $n' = p'q'$, and p' and q' are two primes obtained from $p = 2p' + 1$ and $q = 2q' + 1$. Especially, $\boldsymbol{x}_0 = d$, which is calculated from $d = 0 \bmod n'$ and $d = 1 \bmod n$ through the Chinese remainder theorem. Then, we obtain the private key $S_i = \sum_{j=0}^{T-1} \boldsymbol{x}_j(i)^j \bmod nn'$ $(1 \le i \le K)$.

2) Encryption: Each user U_i can encrypt a plaintext $M \in \boldsymbol{Z}_n$ with the public key Puk, through the module exponential operation as follows:

$$C = E_{\boldsymbol{Puk}}(M) = (1 + n)^M r_i^n \bmod n^2$$

where r_i is a private random number chosen by each U_i from the multiplicative group $\boldsymbol{Z}^*_{n^2}$.

Assuming that there are two plaintexts of $M_1, M_2 \in \boldsymbol{Z}_n$ and a constant $b \in \boldsymbol{Z}_n$, the additive homomorphic property of this encryption scheme can be described as follows:

$$\begin{aligned} E_{\boldsymbol{Puk}}(M_1 + M_2) &= E_{\boldsymbol{Puk}}(M_1) \cdot E_{\boldsymbol{Puk}}(M_2) \\ &= (1 + n)^{M_1+M_2} (r_1 r_2)^n \bmod n^2 \\ E_{\boldsymbol{Puk}}(b \cdot M_1) &= (E_{\boldsymbol{Puk}}(M_1))^b = (1 + n)^{bM_1} {r_1}^{bn} \bmod n^2 \end{aligned}$$

where $r_1, r_2 \in \boldsymbol{Z}^*_{n^2}$ are the private random numbers.

3) Decryption: For decrypting the ciphertext C to obtain the plaintext M, each user U_i first utilizes the private key S_i to encrypt C for obtaining a secret share C_i as follows:

$$C_i = C^{2\Delta S_i}, \text{ where } \Delta = (K!).$$

Then, T shares of C_i will be combined. Finally, based on the combination, the plaintext M could be obtained through the Lagrange interpolation algorithm and the "extraction algorithm" [18]. Similarly, we can obtain $(M_1 + M_2)$ and $(b \cdot M_1)$ through the decryption above.

B. Threshold Signature

For verifying users' identities, we utilize the threshold signature [19], which is based on zero-knowledge proof [32]. The advantages are: 1) in the verification process, the verifier can know nothing about the signature supplier's plaintext or private key; 2) the public key and private keys utilized in the signature scheme are as same as the ones used in the threshold encryption [18], which can minimize the overhead of key management; and 3) the noninteractive property can guarantee the robustness to users' failure in the verification procedure.

Considering a security parameter k and a message M, we briefly review the scheme as follows. Specifically, the scheme falls into three parts: 1) SIG.gen; 2) SIG.signature; and 3) SIG.verify. First, with an input of security parameter k, the key generation SIG.gen(k)$\rightarrow$ (Puk, S_i) outputs the public key Puk and the private key S_i, which are as same as the ones generated in the threshold encryption [18]. Then, with the input of the private key S_i and the message M, the signing algorithm SIG.signature(S_i, M)$\rightarrow SS_i$ outputs a signature SS_i on the message M. Finally, with the input of the public key S_i, the signature SS_i, and the message M, the verifying scheme SIG.verify(Puk, SS_i, M)$\rightarrow\{0, 1\}$ outputs a result indicating whether the signature is valid. Assuming that the RSA problem [19] is hard, this scheme can guarantee the robustness and unforgeability against the adaptive-chosen-message attack [19] in the random oracle mode.

IV. Our Proposed Scheme

A. Overview of SPDDL

For protecting each user's private gradients from being leaked in the DDL training process, we utilize the threshold Pallier algorithm [18] to encrypt the gradients and achieve the secure aggregation of these gradients. Besides, for prohibiting the external adversaries from forging users' identities, we adopt the threshold signature [19] to construct our authentication scheme.

As shown in Fig. 3, six phases are performed in our SPDDL. Specifically, in phase 0, TA initializes the system, generating the public parameters, a public key, as well as private keys, and distributing them to each entity. In phase 1, each user sends his own signature to the cloud server [33], where the authentication could be achieved through, respectively, verifying each user's signature. Phases 2–5 construct our privacy-preserving training protocol, where gradient encryption is achieved in phase 2, ciphertext aggregation is done in phase 3, decryption is executed in phase 4, and the global weight is updated in phase 5. To conclude, our scheme falls into three main parts: 1) initialization; 2) authentication; and 3) privacy-preserving training, as follows.

B. Initialization

In our SPDDL, we utilize a TA for initializing the whole system in phase 0. The main tasks contain three parts: 1) parameter generation; 2) key generation; and 3) data distribution.

1) *Parameter Generation:* We assume there are totally N users constituting the set U_N. Considering the (K, T)-threshold encryption applied in our scheme, in a training process, K users will be randomly chosen as a subgroup $U_K \in U_N$ for training the DDL model, and considering instabilities of the network and users' equipment, no one can guarantee that all the users belonging to U_K can achieve the uploading tasks, therefore, E users will be chosen as a subgroup $U_E \in U_K$, who's gradients constitute the summation of gradients. To summarize, TA will

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

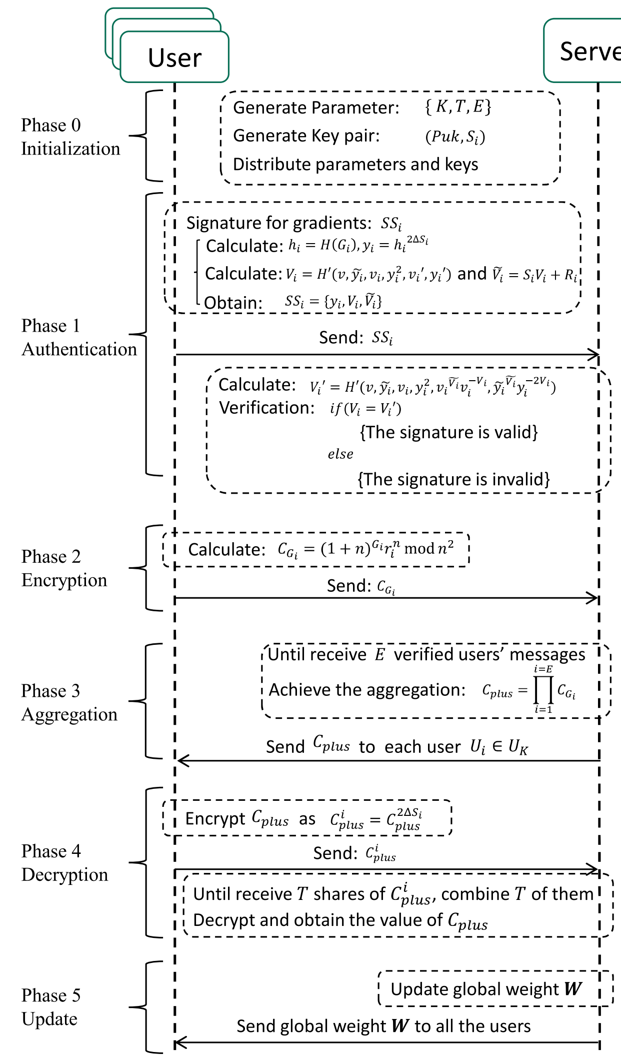
User
Server
Phase 0 Initialization
Generate Parameter: $\{K,T,E\}$
Generate Key pair: (Puk,S_i)
Distribute parameters and keys
Phase 1 Authentication
Signature for gradients: SS_i
Calculate: $h_i = H(G_i), y_i = h_i^{2\Delta S_i}$
Calculate: $V_i = H'(v, \tilde{y}_i, v_i, y_i^2, v_i', y_i')$ and $\tilde{V}_i = S_i V_i + R_i$
Obtain: $SS_i = \{y_i, V_i, \tilde{V}_i\}$
Send: SS_i
Calculate: $V_i' = H'(v, \tilde{y}_i, v_i, y_i^2, v_i^{\tilde{V}_i} v_i^{-V_i}, \tilde{y}_i^{\tilde{V}_i} y_i^{-2V_i})$
Verification: $if\,(V_i = V_i')$
{The signature is valid}
else
{The signature is invalid}
Phase 2 Encryption
Calculate: $C_{G_i} = (1+n)^{G_i} r_i^n \bmod n^2$
Send: C_{G_i}
Phase 3 Aggregation
Until receive E verified users' messages
Achieve the aggregation: $C_{plus} = \prod_{i=1}^{i=E} C_{G_i}$
Send C_{plus} to each user $U_i \in U_K$
Phase 4 Decryption
Encrypt C_{plus} as $C_{plus}^i = C_{plus}^{2\Delta S_i}$
Send: C_{plus}^i
Until receive T shares of C_{plus}^i, combine T of them
Decrypt and obtain the value of C_{plus}
Phase 5 Update
Update global weight $\boldsymbol{W}$
Send global weight $\boldsymbol{W}$ to all the users

Fig. 3. Overview of Our SPDDL

generate K, T, and E, where K and T are two parameters for the (K, T)-threshold encryption, and $K > E > T$.

2) *Key Generation:* In our SPDDL, authentication and encryption can utilize the same public key and private keys in common. The public key Puk will be set as $n = pq$, where n is a large integer, and p as well as q are two primes, while the private key S_i will be, respectively, generated for each user $U_i \in U_K$, as depicted in Section III-A.

3) *Data Distribution:* The public parameter group (K, T, E) and the public key Puk are totally open for users and the cloud server, which will be broadcasted to both of them. However, the private key S_i will be sent to each user $U_i \in U_K$ through the secure tunnel for guaranteeing the security of our framework.

C. Authentication

Since the key generation for the signature is achieved in Section IV-B, we focus on how to sign the gradient message and verify users' identities in this section. For guaranteeing the preciseness of our scheme, we utilize the "packing method" proposed in [20] to represent each user U_i's gradient message as an integer plaintext $G_i \in \mathbf{Z}_n$ needed in the threshold Paillier encryption. Considering the $G_i \in \mathbf{Z}_n$, the whole process takes as each user generates a signature SS_i on the G_i, and the cloud server justifies whether the signature is valid. Specifically, the whole process can be divided into signature and verification as follows.

1) Signature: For obtaining a signature, we first define a hash function H to map messages to the elements of $\mathbf{Z}_n^*$, where n denotes the public key Puk and $\mathbf{Z}_n^*$ is a multiplicative group. Then, we define another hash function H' with the output length of 128 b.

Then, considering G_i, we now introduce how to generate a signature on G_i with the key pair (Puk, S_i) and the public parameter K. We first calculate $h_i = H(G_i)$, then the U_i's signature SS_i could consist of $y_i = h_i^{2\Delta S_i} \in \mathbf{Q}_n$ and a *proof of correctness*, where $\Delta = (K!)$, and $\mathbf{Q}_n$ is a subgroup of squares in $\mathbf{Z}_n^*$. For obtaining the *proof of correctness* $(V_i, \widetilde{V}_i)$, each user U_i will calculate V_i and $\widetilde{V}_i$ as follows:

$$V_i = H'\Big(v, \widetilde{y}_i, v_i, y_i^2, v_i', y_i'\Big); \quad \widetilde{V}_i = S_i V_i + R_i \tag{1}$$

where R_i is a private random number chosen from $\{0, \ldots, 2(L(n) + 2L_1 - 1)\}$, $L(n)$ is denoted as the bit length of n, $L_1 = 128$, v is the number for generating the cyclic group of squares in $\mathbf{Z}_{n^2}^*$, $v_i = v^{\Delta S_i} \bmod n^2$, $\widetilde{y}_i = h_i^{4\Delta}$, $v_i' = v^{R_i}$, and $y_i' = \widetilde{y}^{R_i}$.

Finally, each user will obtain a signature as $SS_i = (y_i, V_i, \widetilde{V}_i)$, which is uploaded to the cloud server for verification.

2) Verification: After receiving the SS_i, the cloud server calculates V_i' as follows:

$$V_i' = H'\Big(v, \widetilde{y}_i, v_i, y_i^2, v^{\widetilde{V}_i} v_i^{-V_i}, \widetilde{y}_i^{\widetilde{V}_i} y_i^{-2V_i}\Big). \tag{2}$$

Following that, the cloud server will verify whether $V_i' = V_i$. If yes, the signature could be valid, otherwise, it should be invalid. Therefore, we can verify if the right G_i encrypted with S_i is indeed uploaded by the right user U_i.

D. Privacy-Preserving Training

For better understanding of how to protect users' privacy in the training process, we first give a brief overview of our privacy-preserving scheme through the pseudocode depicted in Algorithm 1.

Considering there are total N users, and each user $U_i \in U_N$ holds the same local neural network along with a unique local training data set D_i. In an iteration, some users are randomly chosen for calculating their gradients with their local training data sets. Then, their gradients will be aggregated in the cloud server. Following that the cloud server calculates the average value of these gradients and updates the global weight $\boldsymbol{W}$. In the process, lines 7–9 are required to guarantee the privacy of users' gradients. Specifically, four phases will be executed for achieving the privacy-preserving training process, including gradient encryption, secure aggregation, decryption, and model updating, as follows.

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Privacy-Preserving Training

```
Input: Public key: $Puk$, Private key: $S_i$
    Training Data Set: $D_N = \{D_i = (X_i, Y_i), i = 1, 2...N\}$,
    Learning Rate: $\beta$,
    Loss Function: $\mathcal{L}_F(D_i, \boldsymbol{W}) = \mathcal{L}_F(X_i, Y_i, \boldsymbol{W})$.
Output: $\boldsymbol{W}$ (Weight vector)
1:  Randomly initialize $\boldsymbol{W}$;
2:  repeat
3:      Randomly choose some $D_i$ as a subset $D_s \in D_N$;
4:      for (each $D_i \in D_s$) do
5:          Compute gradient $G_i \leftarrow \nabla\mathcal{L}_F(X_i, Y_i, \boldsymbol{W})$;
6:      end for
7:      Encrypt each $G_i$ with $Puk$;
8:      Aggregate all encrypted $G_i$;
9:      Decrypt the encrypted $\sum_{D_i \in D_s} G_i$;
10:     Compute $\widetilde{G} \leftarrow \frac{1}{|D_s|}\sum_{D_i \in D_s} G_i$;
11:     Update $new\ \boldsymbol{W} \leftarrow old\ \boldsymbol{W} - \beta\widetilde{G}$;
12: until Loss function stops decreasing
13: return $\boldsymbol{W}$.
```

1) Gradient Encryption: In phase 2, considering the (K, T)-threshold encryption, K verified users constitute a subgroup $U_K \in U_N$. With the public key $Puk = n$, each user $U_i \in U_K$ will first encrypt the gradient G_i as follows:

$$C_{G_i} = \mathrm{Enc}_{\boldsymbol{Puk}}(G_i) = (1+n)^{G_i} {r_i}^n \bmod n^2 \tag{3}$$

where $n = pq$, (p and q are two primes), and r_i is privately and randomly chosen from the multiplicative group $\boldsymbol{Z}^*_{n^2}$. Then, after the encryption, all users in U_K are asked to upload their results of the ciphertext C_{G_i} to the cloud server.

2) Secure Aggregation: In phase 3, the cloud server randomly chooses E users who have uploaded their results, constituting a group U_E, and achieving the aggregation with each user's C_{G_i} ($U_i \in U_E$), as follows:

$$C_{\text{plus}} = \prod_{i=1}^{i=E} C_{G_i} = (1+n)^{\sum_{i=1}^{i=E} G_i} \left(\prod_{i=1}^{i=E} r_i\right)^n \bmod n^2 \tag{4}$$

where $K > E > T$, and K and T are two parameters for the (K, T)-threshold Paillier cryptosystem, and for simplicity, we denote $G_E = \sum_{i=1}^{i=E} G_i$ as the sum of these E users' gradients. Then, the acquired C_{plus} will be, respectively, sent to each $U_i \in U_K$.

We note that the value of $(\prod_{i=1}^{i=E} r_i)$ is still a random value, and the algorithm is based on the problem of discrete logarithm difficulty, both of which can guarantee each user's gradients from being leaked.

3) Decryption: In phase 4, we achieve decryption for obtaining the sum of the users' gradients G_E.

For decrypting the ciphertext to obtain the summation of these gradients, each user $U_i \in U_K$ will first encrypt C_{plus} with their own private key S_i to obtain a secret share C^i_{plus} as follows:

$$C^i_{\text{plus}} = C^{2\Delta S_i}_{\text{plus}} \tag{5}$$

where $\Delta = K!$. Then, each user's C^i_{plus} will be sent to the cloud server again.

After at least T users' C^i_{plus} are received by the cloud server, T users' C^i_{plus} will be randomly chosen as a set S_T, and be combined as follows:

$$C'_{\text{plus}} = \prod_{i \in S_T} {C^i_{\text{plus}}}^{2\varphi^{S_T}_{(0,i)} \bmod n^2}$$

where

$$\varphi^{S_T}_{(0,i)} = \Delta \cdot \prod_{i' \in S_T \setminus i} \frac{-i'}{i - i'}. \tag{6}$$

Then, for obtaining the plaintext G_E, the Lagrange interpolation algorithm is utilized to conclude that $C'_{\text{plus}} = (1+n)^{4\Delta^2 G_E} \bmod n^2$, then we can calculate $4\Delta^2 G_E$ through the "extraction algorithm" proposed in [18]. Finally, we obtain the value of G_E by multiplying $(4\Delta^2)^{-1} \bmod n$.

In the procedure, if and only if at least T users' cooperation can achieve the decryption for obtaining the final sum of gradients. Therefore, in our proposed framework, the internal participants have no possibility to obtain any useful information except the final summation.

4) Model Updating: In phase 5, after the cloud server obtains the sum of gradients G_E, it will calculate the newest global weight $\boldsymbol{W}$, as follows:

$$\text{new } \boldsymbol{W} \leftarrow \text{old } \boldsymbol{W} - \beta \cdot \frac{G_E}{E} \tag{7}$$

where β denotes the learning rate.

At the end of phase 5, the cloud server will send the newest $\boldsymbol{W}$ to each $U_i \in U_N$ for updating their local models. Noteworthily, phases 2–5 will be executed iteratively until the optimal training model is acquired.

V. Security Analysis

In our model, two different types of attacks are considered. In phase 1, we focus on how to prohibit the external adversaries from forging the users' identities, and during phases 2–5, we put emphasis on protecting users' local gradients from being leaked to internal curious participants. In this section, first, we will briefly discuss the correctness of the authentication. Then, we perform an analysis of how our SPDDL protects users' gradients during the privacy-preserving training procedure, and discuss the expected security level our SPDDL can achieve.

A. Correctness of Authentication

Considering the standard RSA signature to be a secure scheme, the threshold signature algorithm adapted in our SPDDL is unforgeable against the adaptive-chosen-message attack [19], and the unidirectional hash function is secure. We now verify the correctness of the authentication in our SPDDL. Assuming there exist external adversaries can forge users' identities for compromising our SPDDL through the adaptive-chosen-message attack. That means they can forge the users' signatures. However, because of the security of the standard RSA signature scheme, the signature applied in our SPDDL can be unforgeable against the adaptive-chosen-message attack in the random oracle model [19]. This contradicts our assumption, therefore, based on our authentication, no external adversary can forge users' identities, and the

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

correctness is proved. Additionally, based on the hash function unidirectionality and the difficult problem of the discrete logarithm, it can be easily proven that the cloud server cannot deduce each user's private key or gradients.

B. Security of Privacy-Preserving Training

In our settings, we consider users and the cloud server to be honest-but-curious [34], and we assume that only at most $(T-1)$ users are allowed to collude with the cloud server.

In our proposed protocol of privacy-preserving training, all exchanges are achieved between users and the cloud server, therefore, the main security threats come from these two internal entities, and our goal is to protect each user's local gradients from being deduced by other users or the cloud server.

Considering the (K,T)-threshold Paillier encryption [18] applied in our framework is secure via the security of the decisional composite residuosity assumption (DCRA) [35] proposed by Paillier, we can demonstrate that our proposed protocol can protect each user's gradients from being leaked, even if any other $(T-1)$ users are allowed to collude with the cloud server for obtaining the most offensive capabilities. We first start with intuitive analysis of our proposed protocol, as follows.

The main idea of the protocol is to utilize the (K, T)-threshold Paillier encryption for achieving the secure aggregation [36] of users' gradients. As mentioned in Section IV-D, before the secure aggregation, each user's G_i will be encrypted as $C_{G_i} = \text{Enc}_{\boldsymbol{Puk}}(G_i) = (1+n)^{G_i} {r_i}^n \text{mod } n^2$. After being sent to the cloud server, it may try to deduce G_i, however, based on the (K, T)-threshold Paillier scheme applied in our SPDDL cryptosystem, the cloud server cannot infer any useful information of G_i, unless it can collude with other T users. Additionally, the aggregation will be achieved under the ciphertext mode. Therefore, under our settings, our protocol can be secure against the curious cloud server, and the only plaintext acquired by the cloud server is the summation result of users' gradients. Our protocol has the same security against curious users. According to our protocol, there is no direct exchange among users, so that if and only if at least T curious users simultaneously colluding with the cloud server can infer other users' gradients.

Then, considering K and T are two parameters for the (K,T)-threshold Paillier cryptosystem, each user U_i holds the gradient G_i, E users constitute a group $U_E \in U_K$, and U_E's gradients constitute the aggregation value G_E, we formally introduce our theorem and proof as follows.

Theorem 1: Suppose $K > E > T \geq 2$. Assuming users and the cloud server are honest-but-curious, and at most $(T-1)$ users are allowed to collude with the cloud server. After the execution of our privacy-preserving training protocol, each user U_i's $(U_i \in U_E)$ gradient G_i will not be leaked to the cloud server or other users.

Proof: Supposing there exists an attack algorithm, where the cloud server holds the most offensive ability to collude with at most $(T-1)$ users belonging to U_E, and during the training procedure, as the plaintexts input of this algorithm, (Puk, K, T, E) can be known by the cloud server. Then, based on our assumption, U_i's gradient value G_i could be disclosed to other curious users or the cloud server by calculating with these input values. However, each user's G_i will be encrypted as a ciphertext, and based on the (K,T)-threshold Paillier cryptosystem, if and only if at least T users' secret shares can decrypt the ciphertext for obtaining G_i. This is a contradiction. Therefore, there is no such attack algorithm.

We give an intuitive example as follows. In our settings, there are at least two honest users $U_1, U_2 \in U_E$, who can give two different gradients of G_1 and G_2. Supposing there exists an attack algorithm, where the cloud server can deduce U_1's gradient G_1 through calculating with these inputs of (Puk, K, T, E). For verifying if the cloud server can deduce the U_1's gradient value G_1, we first exchange the observed values of G_1 and G_2. Specifically, after the exchange, U_1 will hold the gradient value of G_2, and U_2 will hold the gradient value of G_1. Then, we run the training procedure again. According to our scheme, after the execution, there is no change of these inputs recognized by the cloud server. That means, there is no change in the input values of the attack algorithm. Therefore, via this algorithm, the cloud server would still recognize U_1's gradient value as G_1. However, now U_1's gradient value has been changed from G_1 to G_2. Obviously, this is contradictory to our assumption. Thereby, there will not exist such an attack algorithm and users' gradients will not be inferred by the cloud server in our framework. Similarly, we can use the same method above to prove that each user's gradient cannot be deduced by other users. ■

VI. Performance Evaluation

In this part, we first give an introduction to our experiment settings, then we evaluate the performance for each user and the cloud server, focusing on the overheads of computation and communication. Next, some state-of-the-art works are compared with our proposed SPDDL, in terms of functionality, resource overhead, and accuracy.

A. Experimental Setting

To build "the cloud server," we utilize a server loaded with Ubuntu 18.04 operating system, and the hardware is configured as "RAM: 16 GB, SSD: 256, CPU: 2.10 GHz." Additionally, our SPDDL is evaluated on the MNIST database and simulated in Java 1.7.0. Moreover, we utilize $(K; \lfloor K/2 \rfloor)$-threshold Paillier cryptosystem in the experiment, and the key size is set as 512 b. Furthermore, for comparison, we setup an equal experiment environment for the PPML [12] and DPDL [22], where we adopt the same neural network and data sets utilized in our SPDDL.

B. Performance Analysis for Single user

In our experiment, there are three significant factors, including the number of chosen users $|U| = E = 3K/4$, number of gradients per user $|G| = \boldsymbol{l}$, and drop rate $|R|$. Based on the different values of these three factors, we will evaluate the performance of one single user from both computation and communication overhead. In the procedure, we will fix two

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

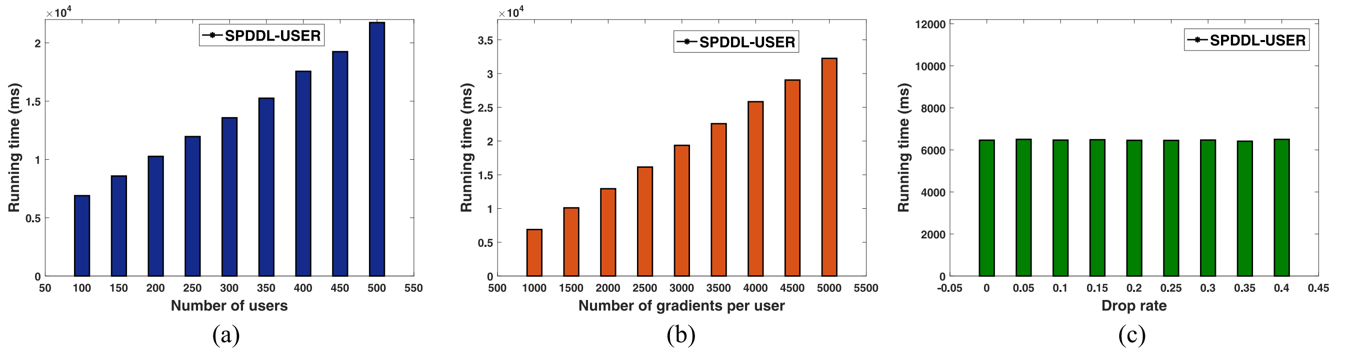

Fig. 4. Computation overhead of each user. (a) $|G| = 1000$, $|R| = 0.1$ changing with different numbers of users. (b) $|U| = 100$, $|R| = 0.1$ changing with different numbers of gradients per user. (c) $|U| = 100$, $|G| = 1000$ keeping constant with different drop rates.

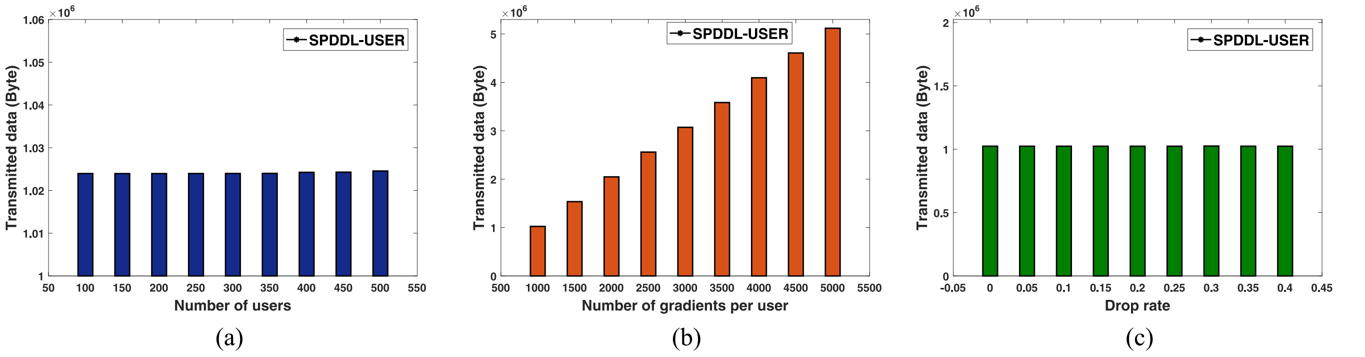

Fig. 5. Communication overhead of each user. (a) $|G| = 1000$, $|R| = 0.1$ keeping constant with different numbers of users. (b) $|U| = 100$, $|R| = 0.1$ changing with different numbers of gradients per user. (c) $|U| = 100$, $|G| = 1000$ keeping constant with different drop rates.

factors of them, and analyze the performance changing with another factor.

1) Computation Overhead $O(E \cdot \boldsymbol{l})$: Each user's computation overhead falls into three parts: 1) generating the signature, which takes $O(E \cdot \boldsymbol{l})$ time; 2) encrypting the local gradients with the public key, which takes $O(\boldsymbol{l})$ time; and 3) encrypting the ciphertext with the private key for decryption, which takes $O(E \cdot \boldsymbol{l})$ time.

As depicted in Fig. 4(a), as the number of users increases, the computation overhead of each user increases linearly. It is because in the processes of signature generation and decryption, the parameter $\Delta = K! = (4E/3)!$ will be used for achieving the exponential operation, specifically, $v_i = v^{\Delta S_i} \bmod n^2$, $\widetilde{y_i} = h_i^{4\Delta}$, and $v_i' = v^{R_i}$ for authentication and $C^i_{\text{plus}} = C^{2\Delta S_i}_{\text{plus}}$ for decryption.

As described in Fig. 4(b), as the number of gradients per user increases, the computation overhead of each user increases linearly. The reason is that in the process of encryption, each user's gradient G_i will be utilized for exponential operation, to be specific, $C_{G_i} = \text{Enc}_{\boldsymbol{Puk}}(G_i) = (1+n)^{G_i} {r_i}^n \bmod n^2$. Meanwhile, each user holds $\boldsymbol{l}$ gradients so that the number of operations will depend on the size of $\boldsymbol{l}$. That means the more gradients of each user, the more exponential operations will be executed.

As shown in Fig. 4(c), although the drop rate increases, the computation overhead per user keeps constant. The increasing drop rate will change the number of existing users as well as the number of existing users, however, the calculation for each user has no relation with either of them, but with the number of users (E). Therefore, the computation overhead keeps constant.

2) Communication Overhead $O(\boldsymbol{l})$: Each user's communication overhead can be broken up as: 1) sending the signature and receiving the result of verification, which takes $O(\boldsymbol{l})$; 2) sending $\boldsymbol{l}$ encrypted gradients (C_{G_i}) and receiving encrypted aggregation, which takes $O(\boldsymbol{l})$; and 3) sending $\boldsymbol{l}$ encrypted ciphertext of aggregation (C^i_{plus}) and receiving updated weights, which takes $O(\boldsymbol{l})$.

As depicted in Fig. 5, the communication overhead keeps constant when the number of users (E) and the drop rate increase. But the increasing number of gradients per user will cause the linear increase of the communication overhead. The reason is that all the sizes of messages sent or received by each user are fixed, the communication overhead is affected by the number of each user's gradients. Furthermore, we note that the message received from the cloud server could just slightly affect the overhead. Therefore, the larger number of gradients per user, the more data should be sent, which consumes more communication overhead.

C. Performance Analysis for the Cloud Server

1) Computation Overhead $O(E \cdot \boldsymbol{l})$: The cloud server's computation overhead consists of: 1) achieving the aggregation of the gradients under a secure model, which takes $O(E \cdot \boldsymbol{l})$ and 2) achieving the decryption for obtaining the final summation of the gradients, which takes $O(E)$. Additionally, we note

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

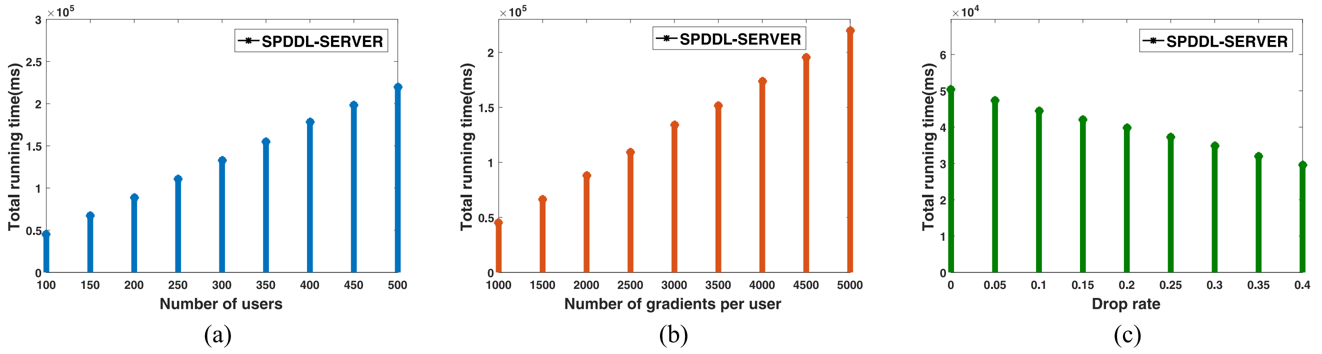

Fig. 6. Computation overhead of the cloud server. (a) $|G| = 1000$, $|R| = 0.1$ increasing linearly with the increasing number of users. (b) $|U| = 100$, $|R| = 0.1$ increasing linearly with the increasing number of gradients per user. (c) $|U| = 100$, $|G| = 1000$ decreasing linearly with the increasing drop rate.

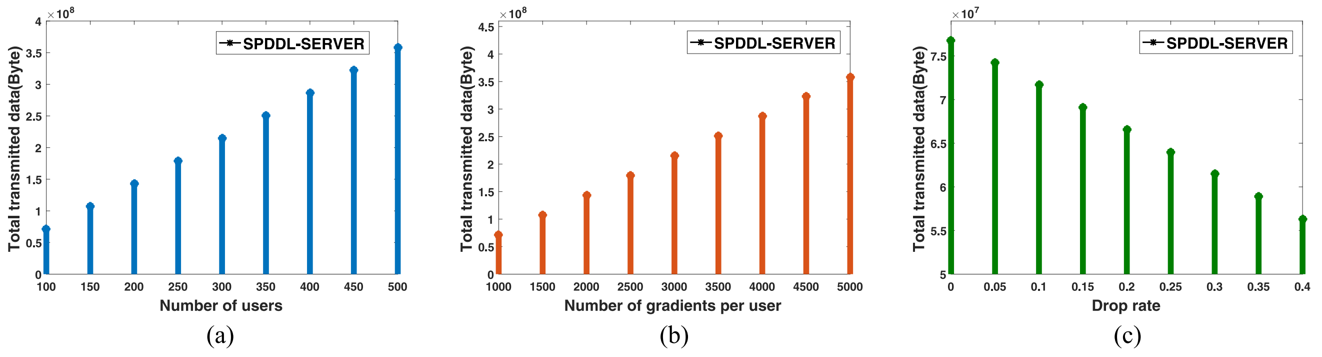

Fig. 7. Communication overhead of the cloud server. (a) $|G| = 1000$, $|R| = 0.1$ increasing linearly with the increasing number of users. (b) $|U| = 100$, $|R| = 0.1$ increasing linearly with the increasing number of gradients per user. (c) $|U| = 100$, $|G| = 1000$ decreasing linearly with the increasing drop rate.

that verifying users' identities with gradients can only slightly increase the computation overhead.

As depicted in Fig. 6(a), as the number of users increases, the server's computation overhead increases linearly, because the more users, the more gradients should be aggregated in the cloud server. Another reason is that $\Delta = K!$ will act as the exponent for the exponential operation in the procedure of decryption.

As described in Fig. 6(b), as the number of each user's gradients increases, the server's computation overhead increases linearly. Similarly, the more gradients per user, the more gradients should be aggregated in the cloud server.

As shown in Fig. 6(c), the server's computation overhead decreases linearly with the increasing drop rate. Because the larger the drop rate, the less existing users, which will cause less gradients being aggregated by the cloud server in the process of aggregation.

2) Communication Overhead $O(E \cdot \boldsymbol{l})$: The cloud server's computation overhead consists of: 1) receiving the proof of correctness from E users and sending the result of verification to all of them, which takes $O(E \cdot \boldsymbol{l})$; 2) receiving encrypted gradients (C_{G_i}) from at least $K/2$ users and sending encrypted aggregation to E users, which takes $O(E \cdot \boldsymbol{l})$; and 3) receiving encrypted ciphertext of aggregation (C^i_{plus}) and sending updated weights, which takes $O(E \cdot \boldsymbol{l})$).

As depicted in Fig. 7(a), as the number of users increases, the communication overhead increases linearly, because the more users, the more messages should be sent and received by the cloud server.

As described in Fig. 7(b), the communication overhead increases linearly, as the number of gradients per user increases. Similarly, it is because of that the more gradients per users, the more messages should be sent and received by the cloud server.

As shown in Fig. 7(c), the communication overhead decreases linearly with the increasing drop rate. Because the larger the drop rate, the less existing users, which will cause less messages being sent from the cloud server, and the less messages being received by the cloud server.

D. Comparison Between SPDDL and Others

For comprehensively evaluating our scheme, we first compare our SPDDL with several related works in terms of functionality, then we compare SPDDL with PPML [12], focusing on resource overhead, and finally, we compare the accuracy between SPDDL and DPDL [22].

1) Functionality: For evaluating the functionality, we compare our SPDDL with the state-of-the-art works of DPDL [22], LDPRLM [23], privacy-preserving deep learning (PPDL) [20], PDLM [21], PPML [12], and SecureML [37], whose main works are similar to ours.

As shown in Table I, all of the schemes above have the capability of protecting users' data privacy. For these two methods via DP: 1) DPDL [22] and 2) LDPRLM [23], they can supply with high efficiency and support multiple functionalities, however, they must balance accuracy and privacy because of the intrinsic property (adding noise for protecting

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COMPARISON OF FUNCTIONALITY

	Privacy Protection	Resistance to Collusion	Robustness to Dropout	Authentication
DPDL	✓	✓	✓	×
LDPRLM	✓	✓	✓	×
PPDL	✓	×	×	×
PDLM	✓	×	×	×
PPML	✓	✓	✓	×
SecureML	✓	×	✓	×
SPDDL	✓	✓	✓	✓

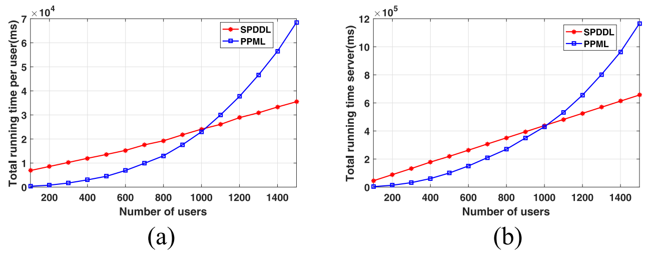

Fig. 8. Computation overhead comparison between SPDDL and PPML, $|G| = 1000$, $|R| = 0.1$, changing with the number of users. (a) Per user. (b) Cloud server.

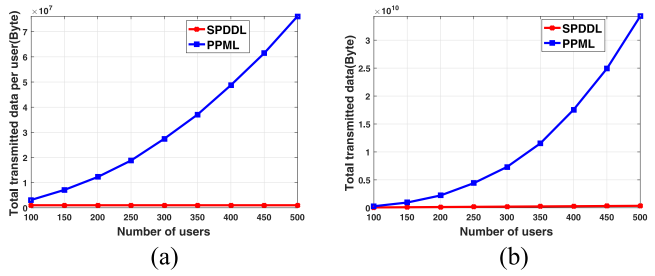

Fig. 9. Communication overhead comparison between SPDDL and PPML, $|G| = 1000$, $|R| = 0.1$, changing with the number of users. (a) Per user. (b) Cloud server.

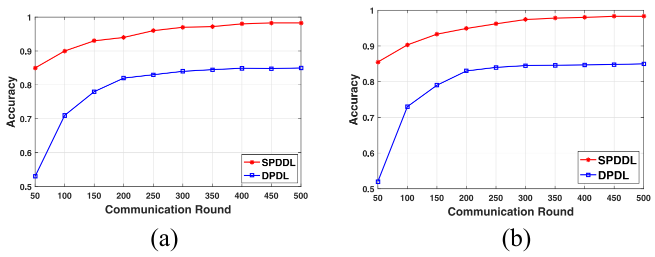

Fig. 10. Accuracy comparison between SPDDL and DPDL. (a) Training accuracy ($\epsilon = 2$). (b) Testing accuracy ($\epsilon = 2$).

privacy) of DP technology, which constrains them within limited security, while our proposed SPDDL, utilizing encryption for privacy protection, will not consider this balance, which makes our scheme more secure than DPDL and LDPRLM. For PPDL [20] and PDLM [21], which are based on HE, users hold the same secret key, which makes it vulnerable if multiple users collude with the cloud server. Additionally, PPDL and PDLM cannot run smoothly, if the user fails during the execution. SMC-based methods of PPML [12] and SecureML [37] can support more functionalities. Nevertheless, since PPML is primarily exploited for privacy protection, how to verify users' identities is not considered in their method. Additionally, SecureML needs two noncolluding servers, which cannot be secure against the collusion between these two servers.

Compared with these schemes, our proposed SPDDL can achieve authentication for verifying users' identities while guaranteeing each user's data privacy. Besides, as same as PPML, the SPDDL has the robustness to users dropping out and keeps secure against the collusion attack.

2) Resource Overhead: For evaluating the computation and communication overhead, the SMC-based PPML [12] is compared with our SPDDL.

As depicted in Fig. 8, the computation cost of our SPDDL increases linearly with the number of users, however, the curve of PPML [12] shows the feature of the quadratic function. According to the analysis in PPML, PPML's computation cost takes $O(\boldsymbol{l}^2+E\cdot\boldsymbol{l})$ for each user and $O(E\cdot\boldsymbol{l}^2)$ for the cloud server, while SPDDL obtains the corresponding results of $O(E\cdot\boldsymbol{l})$ and $O(E\cdot\boldsymbol{l})$. That is why our SPDDL takes lower computation overhead, when the number of users reaches near 1000. Obviously, that makes our SPDDL more suitable for the fog IoT constituted by thousands of fog nodes.

As depicted in Fig. 9, we intuitively found that PPML will cost more communication overhead than SPDDL, both for users and the cloud server. To be specific, in PPML, the more number of users, the faster the communication overhead increases. The root reason is that for achieving the privacy-preserving DL training process, PPML will run much more rounds to exchange keys and encrypted secret shares among users, and many more messages should be transmitted by the cloud server and users.

3) Accuracy: For further analysis of the performance of our SPDDL, we compare the accuracy between our SPDDL and DPDL [22], which is based on DP.

According to the analysis in [10], for guaranteeing the security of DPDL [22] as much as possible, we fix $\epsilon = 2$ for the (ϵ,δ)-DP [22], which requires a medium perturbation noise. In spite of this, in Fig. 10, the experiment results show that the accuracy of DPDL is much lower than our SPDDL. That is because adding perturbation noise will reduce the accuracy of the deep learning model while our scheme keeps the accuracy as same as the original deep learning model without losing privacy.

VII. Related Work

For addressing privacy issues of DDL, some approaches have been proposed, mainly based on DP, SMC, and HE.

A. Differential Privacy-Based Methods

Shokri and Shmatikov [38] proposed the first privacy-preserving DDL model. The main idea is to add noise into gradients uploaded to the cloud server. Similar to Shokri and Shmatikov's scheme [38], Abadi *et al.* [22] proposed a DPDL, which can protect the private information even if the stronger adversary can obtain the training mechanism and access to the parameters of the training model. In their scheme, they

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

exploit the moment accountant for tracking the cumulative privacy loss, which can be utilized for estimating the accumulated privacy loss.

However, there are privacy threats [39], [40] existed in both of their proposed methodologies [22], [38]. Shokri and Shmatikov's scheme [38] have been proven that a little gradient leakage could leak the user's local original data [20] even if the local gradients uploaded by users can be obfuscated via DP. Besides, their schemes [22], [38] are vulnerable if adversaries are allowed to train a generative adversarial network (GAN) [39] to attack the protocol. Additionally, because of excessive injected noise, the learning procedure might be significantly hampered for making high utility.

For supplying better security as well as feasibility, many researches [23]–[25], [41], [42] focus on improving the traditional DP for adjusting to the DDL. However, through extensive practical experiments, Jayaraman and Evans [10] demonstrated that current DP-based DDL methods can rarely offer an acceptable tradeoff between privacy and accuracy.

B. Secure Multiparty Computation-Based Methods

Generic SMC-based methods are mainly based on Yao's GC [14], [43]–[46] and SS [47], [48]. GC-based methods are suitable for two- or three-party SMC. SS-based frameworks can adjust to multiple users, but the communication overhead is unacceptable for practical applications.

For providing an efficient secure learning model, Mohassel and Zhang [37] proposed a privacy-preserving machine learning, where they exploit two noncolluding servers to achieve secure two-party computation (2PC). In their method, oblivious transfer, GC, and SS are combined to construct their 2PC framework. Compared with previous works, their method can supply more efficiency and effectiveness. However, their method is not scalable for more parties.

Additionally, addressed on the weakness of generic SMC protocols, Bonawitz *et al.* [12] presented an approach of practical secure aggregation via SMC. The main concept of their scheme is to utilize the SS and masking method to construct the basis of SMC while achieving the aggregation of gradients in a secure setting. Nevertheless, their scheme must consume many rounds to exchange keys and encrypted secret shares among users, which will cause much more communication overhead.

Hence, for these SMC-based solutions, there still needs much improvement in efficiency and functionality.

C. Homomorphic Encryption-Based Methods

Currently, FHE-based methods [49], [50] cannot be implemented in practical applications due to the unbearable resource overhead. Based on additive HE (AHE), Phong *et al.* [20] proposed an approach of PPDL via the asynchronous stochastic gradient. They utilize the AHE for encrypting the gradients and achieving the secure aggregation, additionally, the cloud server is considered as honest-but-curious, and the users are completely honest. However, in the practical environment, curious users may collude with the cloud server for inferring other users' private information, which makes it vulnerable for their proposed scheme, where users hold the same secret key. Furthermore, their scheme has no robustness for users' failure.

For obtaining a more efficient PPDL, Ma *et al.* [21] designed a PPDL model on the cloud with multiple keys (PDLM) based on the public-key cryptosystem with distributed two trapdoors (DT-PKC) [7], which is constructed by the original Paillier HE [35] and Bresson *et al.*'s cryptosystem [51]. In their method, two noncolluding entities of the service provider (SP) and cloud platform (CP) are asked to jointly achieve the secure training process. However, in the real-world setting, no one can ensure the impossibility of the collusion between the two servers, regardless of whether they are from the same operator. Besides, their scheme is incapable of keeping robustness to users exiting in the training process.

As discussed above, these state-of-the-art works suffer from either low security or low efficiency or weak functionality. Besides, all of them have no consideration of the user's identity authenticity. For improving current works, our SPDDL utilizes the threshold encryption to protect users' privacy, which can better protect data confidentiality than DP-based and AHE-based approaches, and supply with higher accuracy than DP-based methods. Similar to SMC-based approaches, our scheme can defend against the collusion between the cloud server and multiple users, and be robust to users dropping out in the training process. Noteworthily, our scheme takes less communication and computation due to fewer rounds of exchange and the optimal encryption scheme. These properties allow our SPDDL to perform with a better tradeoff between security, efficiency, and functionality. In addition, a conducted authentication scheme makes our SPDDL more secure.

VIII. Conclusion

In this article, we have proposed an SPDDL for fog-cloud computing. We have conducted a privacy-preserving DDL framework to protect users' privacy, and an authentication scheme to prohibit external adversaries from forging users' identities. We claimed that our SPDDL is robust to users' dropping out in the training procedure, and has a preferable balance between efficiency, security, and functionality.

References

[1] B. Demin, S. Parlati, P. F. Spinnato, and S. Stalio, "U-lite, a private cloud approach for particle physics computing," *Int. J. Cloud Appl. Comput.*, vol. 9, no. 1, pp. 1–15, 2019.

[2] J. A. Jeba, S. Roy, M. O. Rashid, S. T. Atik, and M. Whaiduzzaman, "Towards green cloud computing an algorithmic approach for energy minimization in cloud data centers," *Int. J. Cloud Appl. Comput.*, vol. 9, no. 1, pp. 59–81, 2019.

[3] M. M. Hussain and M. S. Beg, "Using vehicles as fog infrastructures for transportation cyber-physical systems (T-CPS): Fog computing for vehicular networks," *Int. J. Softw. Sci. Comput. Intell.*, vol. 11, no. 1, pp. 47–69, 2019.

[4] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin, "The security of autonomous driving: Threats, defenses, and future directions," *Proc. IEEE*, vol. 108, no. 2, pp. 357–372, Feb. 2020.

[5] L. Zhao *et al.*, "Shielding collaborative learning: Mitigating poisoning attacks through client-side detection," *IEEE Trans. Depend. Secure Comput.*, early access, Apr. 14, 2020, doi: 10.1109/TDSC.2020.2986205.

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

[6] N. Saxena, S. Grijalva, V. Chukwuka, and A. V. Vasilakos, "Network security and privacy challenges in smart vehicle-to-grid," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 88–98, Aug. 2017.

[7] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted eHealth systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4101–4112, Sep. 2018.

[8] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing fog computing for Internet of Things applications: Challenges and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 601–628, 1st Quart., 2017.

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

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

[11] S. Chang and C. Li, "Privacy in neural network learning: Threats and countermeasures," *IEEE Netw.*, vol. 32, no. 4, pp. 61–67, Jul./Aug. 2018.

[12] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM CCS*, 2017, pp. 1175–1191.

[13] H. Bae, J. Jang, D. Jung, H. Jang, H. Ha, and S. Yoon, "Security and privacy issues in deep learning," 2018. [Online]. Available: arXiv:1807.11655.

[14] A. C. Yao, "Protocols for secure computations," in *Proc. IEEE 23rd Annu. Symp. Found. Comput. Sci. (SFCS)*, 1982, pp. 160–164.

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

[16] X. Liu, H. Li, G. Xu, S. Liu, Z. Liu, and R. Lu, "PADL: Privacy-aware and asynchronous deep learning for IoT applications," *IEEE Internet Things J.*, early access, Mar. 17, 2020, doi: 10.1109/JIOT.2020.2981379.

[17] S. Kaushik and C. Gandhi, "Ensure hierarchal identity based data security in cloud environment," *Int. J. Cloud Appl. Comput.*, vol. 9, no. 4, pp. 21–36, 2019.

[18] I. Damgård and M. Jurik, "A generalisation, a simpli. cation and some applications of Paillier's probabilistic public-key system," in *Proc. Int. Workshop Public Key Cryptography*, 2001, pp. 119–136.

[19] V. Shoup, "Practical threshold signatures," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2000, pp. 207–220.

[20] 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, no. 5, pp. 1333–1345, May 2018.

[21] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao, "PDLM: Privacy-preserving deep learning model on cloud with multiple keys," *IEEE Trans. Services Comput.*, early access, Sep. 5, 2018, doi: 10.1109/TSC.2018.2868750.

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

[23] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," 2017. [Online]. Available: arXiv:1710.06963.

[24] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1486–1500, Sep. 2019, doi: 10.1109/TIFS.2019.2939713.

[25] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," 2019. [Online]. Available: arXiv:1904.02200.

[26] Y. Li, H. Li, G. Xu, S. Liu, and R. Lu, "EPPs: Efficient privacy-preserving scheme in distributed deep learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[27] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Trans. Ind. Informat.*, vol. 16, no. 10, pp. 6532–6542, Oct. 2019.

[28] 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.

[29] H. Li, D. Liu, Y. Dai, T. H. Luan, and S. Yu, "Personalized search over encrypted data with efficient and secure updates in mobile clouds," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 97–109, Jan.–Mar. 2018.

[30] F. Fischer *et al.*, "Stack overflow considered harmful? The impact of copy&paste on android application security," in *Proc. IEEE Symp. Security Privacy (SP)*, 2017, pp. 121–136.

[31] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. Deng, "Privacy-preserving federated deep learning with irregular users," *IEEE Trans. Depend. Secure Comput.*, early access, Jun. 30, 2020, doi: 10.1109/TDSC.2020.3005909.

[32] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Conf. Theory Appl. Cryptograph. Techn.*, 1986, pp. 186–194.

[33] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, "Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 1, pp. 127–138, Mar. 2015.

[34] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu, "Efficient and privacy-preserving truth discovery in mobile crowd sensing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3854–3865, Apr. 2019.

[35] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 1999, pp. 223–238.

[36] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 870–885, Apr. 2019.

[37] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE S&P*, 2017, pp. 19–38.

[38] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. ACM CCS*, 2015, pp. 1310–1321.

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

[40] 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, Jul. 2019, doi: 10.1109/TIFS.2019.2929409.

[41] Z. Huang, R. Hu, Y. Guo, E. Chan-Tin, and Y. Gong, "DP-ADMM: ADMM-based distributed learning with differential privacy," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1002–1012, Jul. 2019, doi: 10.1109/TIFS.2019.2931068.

[42] B. Jayaraman, L. Wang, D. Evans, and Q. Gu, "Distributed learning without distress: Privacy-preserving empirical risk minimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6343–6354.

[43] Y. Lindell, B. Pinkas, N. P. Smart, and A. Yanai, "Efficient constant round multi-party computation combining BMR and SPDZ," in *Proc. Annu. Cryptol. Conf.*, 2015, pp. 319–338.

[44] Y. Lindell, E. Oxman, and B. Pinkas, "The IPs compiler: Optimizations, variants and concrete efficiency," in *Proc. Annu. Cryptol. Conf.*, 2011, pp. 259–276.

[45] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proc. ACM 19th Annu. ACM Symp. Theory Comput.*, 1987, pp. 218–229.

[46] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proc. 20th Annu. ACM Symp. Theory Comput.*, 1988, pp. 1–10.

[47] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Proc. Annu. Cryptol. Conf.*, 2012, pp. 643–662.

[48] E. Boyle, K.-M. Chung, and R. Pass, "Large-scale secure computation: Multi-party computation for (parallel) RAM programs," in *Proc. Annu. Cryptol. Conf.*, 2015, pp. 742–762.

[49] P. Li *et al.*, "Multi-key privacy-preserving deep learning in cloud computing," *Future Gener. Comput. Syst.*, vol. 74, pp. 76–85, Sep. 2017.

[50] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 1–40, 2009.

[51] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2003, pp. 37–54.

Yiran Li (Graduate Student Member, IEEE) received the M.S. degree in communication and information system from the University of Electronic Science and Technology of China, Chengdu, China, in 2009, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering.

His research interests include cryptography, privacy-preserving deep learning, and data security.

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.

Hongwei Li (Senior Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2008.

He is currently the Head and a Professor with the Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China. From October 2011 to October 2012, he worked as a Postdoctoral Fellow with the University of Waterloo, Waterloo, ON, Canada. His research interests include network security and applied cryptography.

Prof. Li won Best Paper Awards from IEEE MASS 2018 and IEEE HEALTHCOM 2015. He currently serves as the Secretary of IEEE ComSoc CIS-TC. He serves as an Associate Editor for the IEEE INTERNET OF THINGS JOURNAL and *Peer-to-Peer Networking and Applications*, and the Guest Editor of IEEE NETWORK, the IEEE INTERNET OF THINGS JOURNAL and the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.

Guowen Xu (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His research interests include privacy-preserving deep learning, watermarking deep learning, and applied cryptography.

Tao Xiang (Member, IEEE) received the Ph.D. degree in computer science from Chongqing University, Chongqing, China, in 2008.

He is currently a Professor with the College of Computer Science, Chongqing University. His research interests include cryptography, multimedia security, cloud security, and data privacy.

Xiaoming Huang (Member, IEEE) received the M.S. degree from Chengdu University of Information Technology, Chengdu, China, in 2020.

He is currently a General Manager with the Technology Marketing Department, CETC Cyberspace Security Research Institute Company Ltd., Chengdu. His research interests include cognitive domain security, cryptography and information security theory, trusted computing and trusted network technology, and computer and communication security issues.

Dr. Huang is a member of Sichuan electronic information expert group.

Rongxing Lu (Senior Member, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2012.

He is currently an Associate Professor with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada. Before that, he worked as an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, from April 2013 to August 2016. He worked as a Postdoctoral Fellow with the University of Waterloo from May 2012 to April 2013.

Dr. Lu is currently a Senior Member of IEEE Communications Society. He currently serves as the Vice-Chair (Publication) of IEEE ComSoc CIS-TC.

Authorized licensed use limited to: University of New Brunswick. Downloaded on May 21,2021 at 17:55:38 UTC from IEEE Xplore. Restrictions apply.