This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2022.3176191, IEEE Transactions on Information Forensics and Security

PVD-FL: A Privacy-Preserving and Verifiable Decentralized Federated Learning Framework

Jiaqi Zhao, Hui Zhu, *Senior Member, IEEE,* Fengwei Wang, Rongxing Lu, *Fellow, IEEE,* Zhe Liu, *Senior Member, IEEE,* and Hui Li

***Abstract*—Over the past years, the increasingly severe data island problem has spawned an emerging distributed deep learning framework—federated learning, in which the global model can be constructed over multiple participants without directly sharing their raw data. Despite its promising prospect, there are still many security challenges in federated learning, such as privacy preservation and integrity verification. Furthermore, federated learning is usually performed with the assistance of a center, which is prone to cause trust worries and communicational bottlenecks. To tackle these challenges, in this paper, we propose a privacy-preserving and verifiable decentralized federated learning framework, named PVD-FL, which can achieve secure deep learning model training under a decentralized architecture. Specifically, we first design an efficient and verifiable cipher-based matrix multiplication (EVCM) algorithm to execute the most basic calculation in deep learning. Then, by employing EVCM, we design a suite of decentralized algorithms to construct the PVD-FL framework, which ensures the confidentiality of both global model and local update and the verification of every training step. Detailed security analysis shows that PVD-FL can well protect privacy against various inference attacks and guarantee training integrity. In addition, the extensive experiments on real-world datasets also demonstrate that PVD-FL can achieve lossless accuracy and practical performance.**

***Index Terms*—Federated learning, privacy-preserving, verification, decentralized, efficiency.**

I. Introduction

DRIVEN by the explosive growth of computational capability and data volume, deep learning (DL) [1] technique has been widely penetrated into all aspects of society, such as finance, healthcare, and transportation, and changed people's lifestyles significantly. Generally, the DL model quality benefits from large-scale fusion data. Unfortunately, due to the increasing privacy awareness, coupled with the introduction of some strict privacy laws (such as the General Data Protection Regulation (GDPR) [2] and the California Consumer Privacy Act (CCPA) [3]), it is severe to achieve data sharing over multiple institutions, which makes massive precious data isolated as the form of data islands. Consequently, the state-of-the-art concept of federated learning (FL) [4] is introduced by Google, as shown in Fig. 1, where the global DL model can be constructed over multiple participants only through exchanging the global model and local update.

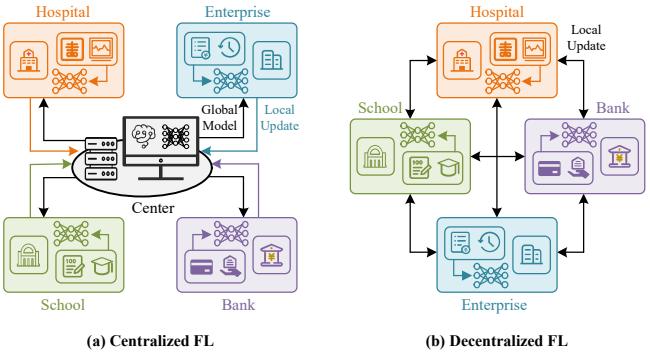

Fig. 1. The architectures of federated learning. (a) Centralized FL where a center connects with all participants. (b) Decentralized FL where participants connect with each other.

Nevertheless, there are still many challenges in FL. First, privacy concerns exist in FL since both the global model and local update may also reveal the data information [5]. On the one hand, the raw data can be recovered accurately via observing the local updates in just a few rounds [6], [7]. On the other hand, through exploiting the difference between consecutive global models, which is equivalent to the aggregation of local updates an adversary participant can also infer the data property and membership in a certain training round [8], which threatens users' privacy considerably. Second, the training integrity is always overlooked in FL. For example, a "lazy" participant or center may execute the stipulated protocol incompletely due to its limited computation resources, which will cause a model accuracy drop. Moreover, in centralized FL, it is hard to judge whether a center is trustworthy and to find one trusted by all participants. Meanwhile, since the complete global model is transferred between the center and all participants at every training round, the center's communication ability may also become the system bottleneck. Therefore, it is urgent and profound to design a privacy-preserving and verifiable decentralized FL framework, which can well guarantee data privacy (containing the protection of global model and local update) and training integrity, meanwhile, can alleviate the trust worries and communicational bottleneck caused by a center.

Aiming at tackling the above challenges, plenty of secure

This work was supported by National Key R&D Program of China (2021YFB3101300), National Natural Science Foundation of China (61972304 and 61932015), Science Foundation of the Ministry of Education (MCM20200101), and Shaanxi Provincial Key Research and Development Program (2020ZDLGY08-04). (Corresponding author: Hui Zhu.)

Jiaqi Zhao, Hui Zhu, Fengwei Wang, and Hui Li are with the School of Cyber Engineering, Xidian University, Xi'an, Shaanxi, China (e-mail: jq_zhao@stu.xidian.edu.cn, zhuhui@xidian.edu.cn, wangfengwei@xidian.edu.cn, and lihui@mail.xidian.edu.cn).

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

Zhe Liu is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China (e-mail: zhe.liu@nuaa.edu.cn).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2022.3176191, IEEE Transactions on Information Forensics and Security

FL schemes have been proposed. Based on secret sharing or homomorphic encryption, many secure aggregation algorithms [9]–[12] are proposed and used to ensure the confidentiality of local updates, but the global model is still leaked in these schemes. Differential privacy-based schemes can hide the data information of a single participant via adding random noises into the local updates [13]–[16], but its privacy guarantee will cause a trade-off of model accuracy drop. Secure multi-party computation (MPC) technique can also be used to resolve the privacy problems in FL [17]–[20], but it only supports model training with a limited number of participants and will bring unacceptable communication overhead. Moreover, a few works [21]–[23] start focusing on the verification of model aggregation by using a homomorphic hash function or Lagrange interpolation, and some decentralized FL schemes [24]–[26] are also proposed recently, but as far as we know, there is no work considering both of them.

In this paper, we propose a privacy-preserving and verifiable decentralized FL framework, namely PVD-FL, through which the global DL model can be constructed securely over multiple participants without the assistance of a center. Specifically, based on a lightweight symmetric homomorphic encryption SHE, we propose an efficient and verifiable cipher-based matrix multiplication (EVCM) algorithm to ensure training security. Then, a suite of decentralized algorithms is carefully designed for achieving high-accuracy decentralized model training. Specifically, the main contributions of this paper are three-fold as follows.

• *First, PVD-FL guarantees the security of model training process.* Benefiting from our proposed EVCM algorithm, both the local update and global model are kept confidential during the whole training process, which can strictly protect data privacy. Meanwhile, every training step of PVD-FL is verifiable, thus ensuring the training integrity. Specifically, EVCM first uses complement to give consideration to both signed ciphertext packing and calculation. Moreover, the random numbers are added to every ciphertext packing for supporting verifiable calculations.

• *Second, PVD-FL achieves high-accuracy DL model training under a decentralized architecture.* Through carefully utilizing EVCM, we design a suite of decentralized algorithms containing model initialization, model propagation, and model updating. Based on them, in PVD-FL, the global model can be constructed over multiple connected participants without the assistance of a center. Moreover, all calculations in PVD-FL will not cause a model accuracy drop.

• *Third, PVD-FL is efficient in computational cost and communication overhead.* In PVD-FL, the global model and local update are both encrypted with the lightweight SHE technique, and the ciphertext calculations can be executed in parallel, which reduce the overhead significantly. The extensive experimental results demonstrate that PVD-FL achieves the same high accuracy as centralized training, in addition, is computation and communication efficient on real-world datasets.

The rest of this paper is organized as follows. In Section II, we define the system model, security requirements, and design goals. In Section III, we outline some building blocks of PVD-FL. Section IV introduces PVD-FL detailedly, followed by the security analysis and performance evaluation in Section V and Section VI. Finally, we review the related work in Section VII and draw a conclusion in Section VIII.

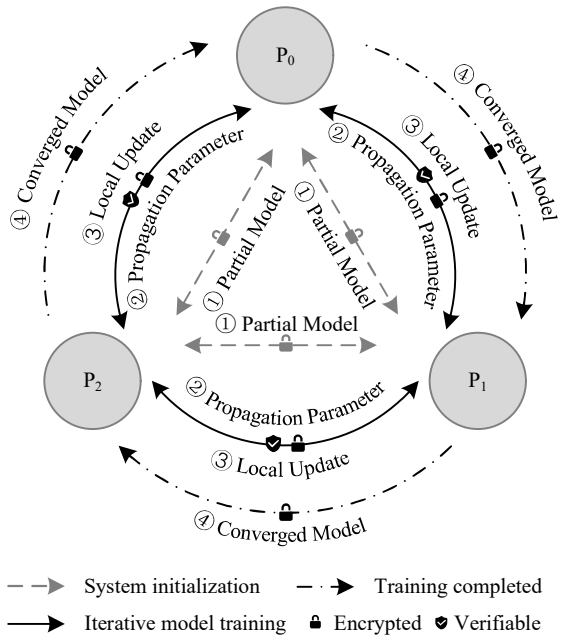

Fig. 2. System model under consideration ($N = 3$).

II. Models, Security Requirements, and Design Goals

In this section, we formally describe our system model and security model, then identify our design goals.

A. System Model

In our system model, we mainly focus on how to train a DL model over multiple connected participants. Thus our system only consists of multiple participants $\{\mathrm{P}_0, \mathrm{P}_1, \cdots, \mathrm{P}_{N-1}\}$, each equipped with a computer or workstation and can connect with others, as shown in Fig. 2.

Each participant $\mathrm{P}_n \in \{\mathrm{P}_0, \mathrm{P}_1, \cdots, \mathrm{P}_{N-1}\}$ is an institution (e.g., bank, hospital, etc.) with powerful storage and computing capability, and their training data are assumed independent and identically distributed (i.i.d.). Specifically, all participants first initialize a ciphertext global model via parameter establishment, key generation, and partial model aggregation. Then, at each training round, every P_n executes cipher-based forward and backward propagations with the collaboration of others. After that, the ciphertext local updates are calculated and shared for updating the ciphertext global model. Finally, the global model converges after multiple training rounds, and the converged model is transferred and decrypted.

B. Threat Model and Security Requirements

In our threat model, all participants are considered honest-but-curious [27], i.e., each P_n is obliged to execute the stipulated protocol process honestly, but tries to independently

infer other participants' sensitive information as much as possible. Specifically, an honest-but-curious participant may try to infer other participants' local training data from the received global model and local update. Since there are commonly conflicts of interest between different participating institutions, the participants are considered non-colluding. In addition, we assume participants may execute incomplete or incorrect calculations due to their limited computing resources or equipment failures during iterative model training. It is worth noting that there may be some other active attacks (e.g., poisoning attack, evasion attack, etc.) in FL. Since PVD-FL focuses on protecting the training data of participants and verifies the miscalculation during model training, these attacks are currently out of the scope of this paper and will be considered in future work. Under these threat assumptions, the following security requirements should be satisfied.

• *Privacy.* First, the raw local training data involves users' privacy directly. Moreover, in FL, the global model and local update also contain massive sensitive data information. Therefore, PVD-FL should protect not only the raw local training data, but also the intermediate parameters during model training.

• *Verification.* During each training round, limited by local computing resources, a participant may execute the protocol calculations incompletely, which will affect the normal model training. Meanwhile, there may be some calculation errors due to equipment failures, which will also cause wrong training results. Therefore, PVD-FL should have the ability to verify the miscalculations and ensure the training integrity.

Moreover, it is noteworthy that these two types of miscalculation are essentially different from the poisoning attacks. The purpose of poisoning attacks is to maliciously destroy the model training, but that of the incomplete or incorrect calculations is to normally participate in the model training even though the computing resources are limited or the computing equipment is error-prone.

C. Design Goals

Under the system and threat models mentioned above, aiming to achieve privacy-preserving and verifiable decentralized federated learning, PVD-FL should satisfy the following three objectives.

• *Guarantee the security during the whole model training process.* There is massive sensitive information in users' data (e.g., electronic healthcare records, location trajectory, etc.), once leaked, it will cause a serious threat to user privacy or even a violation of relevant privacy laws. Moreover, participants' incomplete or wrong calculations will greatly affect the model training effect. Therefore, our proposed PVD-FL should guarantee privacy-preserving and integrity of model training.

• *Achieve high-accuracy decentralized training for DL model over multiple participants.* In FL, the global model is usually trained in a centralized way, i.e., the global model and local update are exchanged between a server and many participants at every training round, where the communication capability of the server becomes the main factor limiting the model training efficiency. Therefore, without an accuracy drop, we consider the decentralized model training via the collaboration of just multiple participants in PVD-FL.

• *Low computation and communication overhead.* In recent years, the explosive growth of computational and communicational capabilities has brought about the rapid development of DL technology, which is applied to all aspects of people's daily life. Nevertheless, most privacy-preserving DL schemes will cause at least two orders of magnitude overhead than origin schemes, since the sensitive data are always protected by the less efficient cryptography methods, e.g., encryption. Therefore, the proposed protocol in PVD-FL should be carefully designed for accomplishing high efficiency in terms of computation and communication.

III. Preliminaries

A. Deep Learning

Deep learning [28], [29] is a new research direction in machine learning, whose motivation is to build a deep neural network (DNN) model that can simulate the human brain for making decisions. A DNN model consists of an input layer, several hidden layers (containing fully-connected (FC), convolutional (Cov), and pooling layers), and an output layer. In supervised DL, a data sample contains its feature x and label y, and the loss function ζ is used to measure the difference between the labels and predicted outputs. For minimizing ζ, a DNN model is trained by iteratively performing the following phases [30].

1) Forward Propagation: First, given a random data batch X as input a^0, output a^L is obtained as follows, where X contains β data samples, and L denotes the layer depth.

• *Cov Layer.* The Cov layer is used to extract the data features via multiple small $d_l \times d_l$ convolutional kernels. The input can be represented as $a^{l-1} \in \mathbb{R}^{\beta \times w_{l-1} \times h_{l-1} \times c_{l-1}}$, where w_{l-1}, h_{l-1}, and c_{l-1} denote the width, height, and channel number respectively. Let $W^l \in \mathbb{R}^{d_l \times d_l \times c_{l-1} \times c_l}$ denotes the multiple convolutional kernels, the output $a^l \in \mathbb{R}^{\beta \times w_l \times h_l \times c_l}$ can be calculated as $a^l = \sigma(z^l) = \sigma(a^{l-1} * W^l + b^l)$, where $*$ denotes the convolutional operation, $b^l \in \mathbb{R}^{w_l \times h_l \times c_l}$ is the bias unit, $\sigma(\cdot)$ denotes the activation function (such as Sigmoid and ReLU), and a^l satisfies $w_l = \lceil (w_{l-1} - d_l + 1)/s_l \rceil$ and $h_l = \lceil (h_{l-1} - d_l + 1)/s_l \rceil$.

• *FC Layer.* In FC layers, every entry of $a^{l-1} \in \mathbb{R}^{h_{l-1}}$ is connected to the output $a^l \in \mathbb{R}^{h_l}$ via the weight matrix $W^l \in \mathbb{R}^{h_{l-1} \times h_l}$. Given a^{l-1}, the output a^l of FC layer can be calculated as $a^l = \sigma(z^l) = \sigma(a^{l-1} W^l + b^l)$.

• *Pooling Layer.* Analogous to the Cov layer, the pooling layer calculates the max or average value of every region as its output, which can be represented as $\rho(a^l)$.

After obtaining output a^L, the loss value is calculated as $loss = \zeta(a^L, Y)$, and ζ may be the cross-entropy error function, mean square error function, and so on. After that, the derivative of the L-th layer can be calculated as $\delta^L = \frac{\partial loss}{\partial a^L}$.

2) Backward Propagation: The backward propagation first calculates the derivative δ^l as follows.

• Cov Layer: $\delta^l = \delta^{l+1} * \mathrm{rot}(W^{l+1}) \odot \sigma'(z^l)$,
• FC Layer: $\delta^l = \delta^{l+1} [W^{l+1}]^T \odot \sigma'(z^l)$,
• Pooling Layer: $\delta^l = \mathrm{upsample}(\delta^{l+1}) \odot \sigma'(z^l)$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2022.3176191, IEEE Transactions on Information Forensics and Security

where $\odot$ is the Hadamard product, $\mathrm{rot}(W^l)$ will rotate each convolutional kernel by 180 degrees, i.e., first reverse it up and down, then reverse it left and right, and $\mathrm{upsample}(\delta^{l+1})$ will recover δ^{l+1} to the size before pooling.

After that, the gradients G_W^l and G_b^l can be calculated as:
- Cov Layer: $G_W^l = [a^{l-1}]^T \delta^l, G_b^l = \sum \delta^l$,
- FC Layer: $G_W^l = a^{l-1} * \delta^l, G_b^l = \sum \delta^l$.

3) Model Updating: After the backward propagation, W and b are updated as $W^l = W^l - \frac{\alpha}{\beta} \cdot G_W^l$ and $b^l = b^l - \frac{\alpha}{\beta} \cdot G_b^l$, where α is the learning rate.

Moreover, since the convolutional operation cannot be directly calculated over ciphertexts, in PVD-FL, it is transferred to linear matrix multiplication based on the method proposed in [31]. Specifically, each input a^{l-1} will be flattened to a matrix $\tilde{a}^{l-1} \in \mathbb{R}^{w_l h_l \times d_l^2 c_{l-1}}$, which is written as $\tilde{a}^{l-1} \leftarrow \mathrm{flattenX}(a^{l-1})$. And the convolutional kernel W^l will be flattened to $\tilde{W}^l \in \mathbb{R}^{d_l^2 c_{l-1} \times c_l}$ as $\tilde{W}^l \leftarrow \mathrm{flattenW}(W^l)$. After that, the convolutional result can be obtain by $a^{l-1} * W^l = \mathrm{clusterX}(\tilde{a}^{l-1}\tilde{W}^l)$, where function $\mathrm{clusterX}$ can cluster the $w_l h_l \times c_l$ matrix to a $w_l \times h_l \times c_l$ matrix as the output of the current layer.

B. The SHE Technique

The symmetric homomorphic encryption technique named SHE [32] can compute homomorphic addition and multiplication efficiently, which is IND-CPA secure [33], [34] and contains three functions written as $\mathrm{SHEKeyGen}, \mathrm{SHEEnc}$, and SHEDec.

- $\mathrm{SHEKeyGen}(k_0, k_1, k_2) \rightarrow (pp, key)$. Given security parameters (k_0, k_1, k_2) satisfying $k_1 < k_2 < k_0$ (denoting the plaintext, random, and key space respectively), first choose two k_0-bit prime numbers p, q, and calculate $\mathcal{N} = pq$. Then, select a k_2-bit random number $\mathcal{L}$. Finally, output the secret key $key = (p, \mathcal{L})$ and public parameter $pp = (k_0, k_1, k_2, \mathcal{N})$.
- $\mathrm{SHEEnc}(key, m) \rightarrow [\![m]\!]$. Given key and a k_1-bit message m, the cipher $[\![m]\!]$ is calculated as $[\![m]\!] = (r\mathcal{L} + m)(1 + r'p) \mod \mathcal{N}$, where r and r' are k_2-bit and k_0-bit random numbers.
- $\mathrm{SHEDec}(key, [\![m]\!]) \rightarrow m$. Given key, ciphertext $[\![m]\!]$ can be decrypted as $m = ([\![m]\!] \mod p) \mod \mathcal{L}$.

Given two ciphertexts $[\![m_1]\!]$, $[\![m_2]\!]$, and a plaintext m_3, the SHE technique supports the homomorphic calculations as $[\![m_1]\!] \oplus [\![m_2]\!] = [\![m_1 + m_2]\!]$, $[\![m_1]\!] \oplus m_3 = [\![m_1 + m_3]\!]$, $[\![m_1]\!] \otimes [\![m_2]\!] = [\![m_1 m_2]\!]$, and $[\![m_1]\!] \otimes m_3 = [\![m_1 m_3]\!]$, where $\oplus$ and $\otimes$ represent the homomorphic addition and multiplication respectively. Moreover, the SHE technique can calculate almost unlimited ciphertext additions and $\lfloor \frac{k_0}{2k_2} - 1 \rfloor$ ciphertext multiplications.

C. The SIMD Technique

The Single Instruction Multiple Data (SIMD) technique [35], [36] can be used in encryption to partition the entire plaintext space into multiple plaintext slots, which allows encrypting multiple plaintexts into one ciphertext. Moreover, the SIMD ciphertext can execute homomorphic operations in parallel. As shown in Fig. 3, a plaintext slot is 4-bit and the vector addition can be calculated over ciphertexts just by one homomorphic addition. The SIMD technique is used as a building block to construct our EVCM algorithm, which will be introduced in Section IV-A.

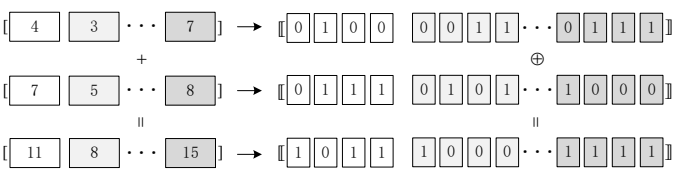

Fig. 3. An example of the SIMD technique.

IV. Proposed Scheme

In this section, we first introduce an efficient and verifiable cipher-based matrix multiplication algorithm named EVCM. Then, by utilizing EVCM, a suite of cipher-based algorithms are carefully designed, which are used to construct the privacy-preserving and verifiable decentralized FL framework PVD-FL.

A. EVCM Algorithm

Based on the SHE and SIMD techniques, the EVCM algorithm is proposed, which can calculate cipher-based matrix multiplication efficiently and verifiably over ciphertexts. In this algorithm, by converting the matrix element to its complement, signed calculations can be executed over ciphertext packing. Moreover, the random numbers are added to every ciphertext packing for verifying the calculation correctness. As shown in Fig. 4, the matrix multiplication of $A \in \mathbb{Z}^{W \times U}$ and $B \in \mathbb{Z}^{U \times V}$ is calculated with nine EVCM functions, namely KGen, PGen, Sign, Pack, Enc, Mul, Dec, Unpack, and Ver, which are introduced detailedly in the following.

- $\mathrm{KGen}(k_0, k_1, k_2) \rightarrow (pk, sk)$: Given the security parameters of SHE, first execute $\mathrm{SHEKeyGen}(k_0, k_1, k_2)$ to obtain key as the private key sk of EVCM. Then, encrypt two zero values with SHEEnc, and the public key of EVCM is written as $pk = ([\![0]\!]_0, [\![0]\!]_1)$.
- $\mathrm{PGen}(L_0, k_1) \rightarrow PP$: Given the bit length L_0 of each element in matrices A and B, the bit length of elements in C is first set to L_1 satisfying $L_1 > 2L_0 + \log(U) - 1$. Then, the bit length of each plaintext slot is set as L_2 satisfying $L_2 > 4L_0 + 3\log(U) - 3$. After that, according to the security parameter k_1 of SHE, the number of plaintext slots is determined as N_s, which satisfies $L_2 \cdot N_s < k_1$. Finally, $\{N_s, L_0, L_1, L_2\}$ is written as the public parameter PP.
- $\mathrm{Sign}(B, SV, PP) \rightarrow B'$: For verification, first generate multiple U-dimension random vectors $sv_0, sv_1, \cdots, sv_{N_p - 1}$ as signature vectors where each element $x \in sv_n$ satisfies $x \in (-2^{L_0 - 1}, 0) \cup (0, 2^{L_0 - 1})$ and $N_p = \lceil \frac{V}{N_s - 1} \rceil$. Then, insert $SV = (sv_0, sv_1, \cdots, sv_{N_p - 1})$ into B for every $N_s - 1$ column to obtain B'.
- $\mathrm{Pack}(B', PP) \rightarrow \langle \hat{B}' \rangle$: For each element $b_{u,v} \in (-2^{L_0 - 1}, 2^{L_0 - 1})$ in matrix B', it is first encoded into its L_1-bit complement form, written as $\hat{b}_{u,v}$. After that, the signed number $b_{u,v}$ is transferred into a positive integer, which can be packed and encrypted directly. Then, similar to the SIMD technique introduced in Section III-C, for each row u in

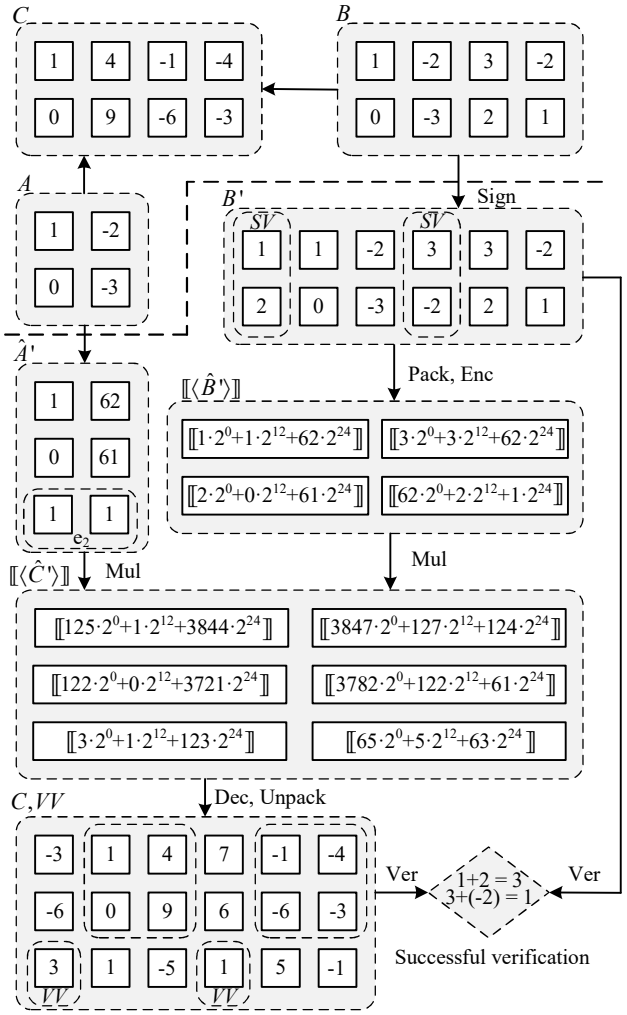

Fig. 4. An example of the EVCM algorithm ($L_0 = 3, L_1 = 6, L_2 = 12, N_s = 3$).

complement matrix $\hat{B}'$, N_s elements are packed into one plaintext $\langle \hat{b} \rangle_{u,s}$ by computing

$$\langle \hat{b} \rangle_{u,s} = \sum_{i=0}^{N_s-1} \hat{b}_{u,sN_s+i} \cdot 2^{iL_2}. \tag{1}$$

It is noteworthy that the element number in the last packing may be less than N_s if $V + N_p$ is not divisible by N_s.

• $\mathrm{Enc}(\langle \hat{B}' \rangle, pk) \rightarrow [\![\langle \hat{B}' \rangle]\!]$: Each element $\langle \hat{b} \rangle_{u,s}$ in matrix $\langle \hat{B}' \rangle$ is encrypted with pk as

$$[\![\langle \hat{b} \rangle_{u,s}]\!] = \langle \hat{b} \rangle_{u,s} \oplus (r_0 \otimes [\![0]\!]_0) \oplus (r_1 \otimes [\![0]\!]_1), \tag{2}$$

where r_0 and r_1 are two k_2-bit random numbers. Then, the ciphertext packed matrix is written as $[\![\langle \hat{B}' \rangle]\!]$.

• $\mathrm{Mul}(A, [\![\langle \hat{B}' \rangle]\!], PP) \rightarrow [\![\langle \hat{C}' \rangle]\!]$: First, a vector e_W of all ones is added into the last row of A to obtain A'. Similarly, each element $a_{w,u} \in (-2^{L_0-1}, 2^{L_0-1})$ in matrix A' is then encoded as its L_1-bit complement $\hat{a}_{w,u}$. After that, each ciphertext element $[\![\langle \hat{c} \rangle_{w,s}]\!] \in [\![\langle \hat{C}' \rangle]\!]$is calculated as

$$[\![\langle \hat{c} \rangle_{w,s}]\!] = \bigoplus_{i=0}^{U-1} \hat{a}_{w,i} \otimes [\![\langle \hat{b} \rangle_{i,s}]\!]. \tag{3}$$

• $\mathrm{Dec}([\![\langle \hat{C}' \rangle]\!], sk) \rightarrow \langle \hat{C}' \rangle$: Elements in $[\![\langle \hat{C}' \rangle]\!]$ are decrypted with SHEDec to obtain the decrypted matrix $\langle \hat{C}' \rangle$.

• $\mathrm{Unpack}(\langle \hat{C}' \rangle, PP) \rightarrow (C, VV)$: First, each element $\langle \hat{c} \rangle_{w,s}$ is unpacked with Algorithm 1. After that, we obtain a complement matrix $\hat{C} \in \mathbb{Z}^{W \times V}$ and a complement vector $\widehat{VV} \in \mathbb{Z}^N$. Finally, every complement $\hat{c}_{w,v}$ or $\hat{vv}_s$ is decoded into its true form, which forms the final matrix product C and verification vector VV.

• $\mathrm{Ver}(SV, VV) \rightarrow b$: For all $s = 0, 1, \cdots, N_p - 1$, if $vv_s = \Sigma sv_s$, the matrix product C is considered correct and b is set to 0. If else, b is set to 1.

Algorithm 1 Unpack

```
for j = N_s − 1, N_s − 2, ··· , 1 do
    ĉ_{w,s(N_s−1)+j−1} = ⟨ĉ⟩_{w,s} // 2^{jL_2}.
    ⟨ĉ⟩_{w,s} = ⟨ĉ⟩_{w,s} − ĉ_{w,s(N_s−1)+j−1} · 2^{jL_2}.
if w = W then
    v̂v_s = ⟨ĉ⟩_{w,s}.
```

Correctness. *Each element $c_{w,v} \in C$ can be calculated correctly, i.e., $c_{w,v} = \sum_{i=0}^{U-1} a_{w,i} b_{i,v}$.*

Proof. According to the homomorphic properties of SHE, Equation (3) can be deduced as:

$$[\![\langle \hat{c} \rangle_{w,s}]\!] = \bigoplus_{i=0}^{U-1} \hat{a}_{w,i} \otimes [\![\langle \hat{b} \rangle_{i,s}]\!] = [\![\sum_{i=0}^{U-1} \hat{a}_{w,i} \langle \hat{b} \rangle_{i,s}]\!].$$

Since $L_2 \cdot N_s < k_1$, the bit length of $\langle \hat{c} \rangle$ will not exceed the plaintext space and $[\![\langle \hat{c} \rangle_{w,s}]\!]$ can be decrypted correctly. According to Equation (1), $\langle \hat{c} \rangle_{w,s}$ can be represented as

$$\begin{aligned} \langle \hat{c} \rangle_{w,s} &= \sum_{i=0}^{U-1} (\sum_{j=0}^{N_s-1} \hat{a}_{w,i} \cdot \hat{b}_{i,sN_s+j} \cdot 2^{jL_2}) \\ &= \sum_{j=0}^{N_s-1} (\sum_{i=0}^{U-1} \hat{a}_{w,i} \cdot \hat{b}_{i,sN_s+j}) \cdot 2^{jL_2}. \end{aligned}$$

According to $L_2 > 4L_0 + 3\log(U) - 3$ and $L_1 > 2L_0 + \log(U) - 1$, we can obtain $L_2 > 2L_1 + \log(U) - 1$. Meanwhile, since $\hat{c}_{w,s(N_s-1)+j-1} = \sum_{i=0}^{U-1} \hat{a}_{w,i} \cdot \hat{b}_{i,sN_s+j}$ and $\hat{a}_{w,i}, \hat{b}_{i,sN_s+j}$ are L_1-bit complements, the plaintext slot is larger than the space of $\hat{c}_{w,s(N_s-1)+j-1}$. Therefore, every element $\hat{c}_{w,s(N_s-1)+j-1}$ can be recovered from its plaintext slot with Algorithm 1.

Then, according to $L_1 > 2L_0 + \log(U) - 1$ and $a_{w,u}, b_{u,v} \in (-2^{L_0-1}, 2^{L_0-1})$, i.e., $c_{w,s(N_s-1)+j-1} < 2^{L_1}$, $\hat{c}_{w,s(N_s-1)+j-1}$ can be decoded correctly into its true form $c_{w,s(N_s-1)+j-1} = a_{w,i} \cdot b_{i,s(N_s-1)+j-1}$.

Finally, the column index of C is rewritten as $v = s(N_s - 1) + j - 1$, and $c_{w,v}$ can be represented as $c_{w,v} = \sum_{i=0}^{U-1} a_{w,i} b_{i,v}$.

Therefore, $c_{w,v} \in C$ is calculated correctly as $c_{w,v} = \sum_{i=0}^{U-1} a_{w,i} b_{i,v}$. □

B. Description of PVD-FL framework

The proposed PVD-FL mainly contains four phases: 1) System initialization; 2) Cipher-based model propagation; 3) Cipher-based model updating; 4) Training completed. The

TABLE I
NOTATIONS OF PVD-FL

Notations	Definition
N	The number of participants.
k_0, k_1, k_2	Security parameters of SHE.
α, β	Learning rate and batch size.
h_l, d_l, c_l	The size of l-th layer.
ρ, ζ, σ	Pooling, loss, and activation functions.
W^l, b^l	The global model of the l-th layer.
L_0	Precision parameter.
κ	Expansion factor.
PP	Public parameter.
fpk_n, fsk_n	EVCM key pair for forward propagation.
bpk_n, bsk_n	EVCM key pair for backward propagation.
M_n^l, p_n^l	The partial model of the l-th layer.
SV, ZV	Random signature vector and zero vector.
z^l	The output of the l-th layer.
a^l	The l-th output after activation function.
δ^l	The derivative of the l-th layer.
G_W^l, G_b^l	The local update of the l-th layer.
$\langle X \rangle$	Packed X.
$[\![X]\!]_i$	Ciphertext X encrypted with fpk_n or bpk_n.

overview of PVD-FL is shown in Fig. 5. Specifically, in system initialization, the common ciphertext global model is first generated and distributed among every participant, which is invisible to any participant. Then, based on the EVCM algorithm, all participants can collaboratively perform forward and backward propagations over the ciphertext global model, and obtain the derivative of every layer. After that, the ciphertext derivatives are used to calculate ciphertext local updates, which are sent to every participant for updating their ciphertext global model locally. Finally, through multi-round model training (containing model propagation and updating), the global model is judged to converge, and then is transferred and decrypted by every participant. Moreover, the used notations are listed in TABLE I for describing PVD-FL more clearly.

1) System Initialization: In this phase, all participants first establish common system parameters. Then, every participant P_n generates two pairs of EVCM keys for forward and backward propagations, respectively. Finally, through exchanging and aggregating partial ciphertext models, participants obtain the initial ciphertext global model.

• Step 1: Parameter Establishment

In this step, N participants first agree on the security parameters (k_0, k_1, k_2) of SHE.

Then, the model structure is determined by all participants, which consists of the type (i.e., FC or Cov) and size h_l, d_l, c_l of each layer, activation function σ, pooling function ρ, loss function ζ, learning rate α, batch size β, and so on. Specifically, the model structure contains N FC and Cov layers, and the weight matrix of the l-th layer is written as W^l. For simplicity, we assume the activation function is used for every FC or Cov layer, and every Cov layer is followed by a pooling layer.

After that, determine the precision parameter L_0. Since the float numbers can not be encrypted directly by SHE, each element x in data or model is first expanded by $x = \lfloor \kappa x \rfloor$ (expansion factor κ is less than 2^{L_0-1}) before encryption and divided by κ when decryption.

Finally, each participant executes $\mathrm{PGen}(L_0, k_1)$ to generate public parameter $PP = \{N_s, L_0, L_1, L_2\}$.

• Step 2: Key Generation

In this step, according to the security parameters of SHE, each participant P_n first executes $\mathrm{KGen}(k_0, k_1, k_2)$ to generate two EVCM key pairs, written as (fpk_n, fsk_n) and (bpk_n, bsk_n), which are used for cipher-based forward and backward propagations respectively.

Then, P_n sends its public keys fpk_n and bpk_n to other participants over a secure channel.

Finally, P_n has its private keys $\{fsk_n, bsk_n\}$, and all public keys $\{fpk_i, bpk_i\}_{i=0,\cdots,N-1}$.

• Step 3: Ciphertext Model Initialization

As shown in Algorithm 2, every P_n first generates its partial model randomly, which is then encrypted and exchanged with others, and the ciphertext global model is finally obtained through aggregating multiple partial ciphertext models. Moreover, the global model is processed by function Sign for executing subsequent training integrity verification. The detailed process is as follows.

Algorithm 2 Ciphertext Model Initialization

for $l = 1, \ldots, N$ **do**
  ▷ *Generate partial model.* ◁
  if l-th layer is FC layer **then**
    Randomly generate $M_n^l \in \mathbb{R}^{(h_{l-1} \times h_l)}$ and $p_n^l \in \mathbb{R}^{h_l}$.
    Transpose M_n^l to $[M_n^l]^T \in \mathbb{R}^{(h_l \times h_{l-1})}$.
  if l-th layer is Cov layer **then**
    Randomly generate $M_n^l \in \mathbb{R}^{(d_l^2 c_{l-1} \times c_l)}$, $p_n^l \in \mathbb{R}^{(w_l h_l \times c_l)}$.
    $\mathrm{rot}(M_n^l) \to [M_n^l]^T \in \mathbb{R}^{(d_l^2 c_l \times c_{l-1})}$.
  ▷ *Encrypt partial model for forward propagation.* ◁
  $\langle M_n^l \rangle \leftarrow \mathrm{Pack}(\mathrm{Sign}(M_n^l, ZV))$.
  for $i = 0, 1, \cdots, N-1 (i \neq n-1)$ **do**
    Send $[\![\langle M_n^l \rangle]\!]_{i+1} \leftarrow \mathrm{Enc}(\langle M_n^l \rangle, fpk_{i+1})$ to P_i.
  $\langle p_n^l \rangle \leftarrow \mathrm{Pack}(\mathrm{Sign}(p_n^l, ZV))$.
  for $i = 0, 1, \cdots, N-1$ **do**
    Send $[\![\langle p_n^l \rangle]\!]_{i+1} \leftarrow \mathrm{Enc}(\langle p_n^l \rangle, fpk_{i+1})$ to P_i.
  Generate random signature vectors SV_n^l.
  $\langle M_n^l \rangle \leftarrow \mathrm{Pack}(\mathrm{Sign}(M_n^l, SV_n^l))$.
  Send $[\![\langle M_n^l \rangle]\!]_n \leftarrow \mathrm{Enc}(\langle M_n^l \rangle, fpk_n)$ to P_{n-1}.
  ▷ *Encrypt partial model for backward propagation.* ◁
  if l-th layer is FC layer **then**
    Generate random signature vectors ${SV'}_n^l$.
    $\langle [M_n^l]^T \rangle \leftarrow \mathrm{Pack}(\mathrm{Sign}([M_n^l]^T, {SV'}_n^l))$.
    Send $[\![\langle [M_n^l]^T \rangle]\!]_n \leftarrow \mathrm{Enc}(\langle [M_n^l]^T \rangle, bpk_n)$ to P_{n+1}.
    $\langle [M_n^l]^T \rangle \leftarrow \mathrm{Pack}(\mathrm{Sign}([M_n^l]^T, ZV))$.
    for $i = 0, 1, \cdots, N-1 (i \neq n+1)$ **do**
      Send $[\![\langle [M_n^l]^T \rangle]\!]_{i-1} \leftarrow \mathrm{Enc}(\langle M_n^l \rangle, bpk_{i-1})$ to P_i.
▷ *Aggregate partial ciphertext models.* ◁
for $l = 1, \ldots, N$ **do**
  $[\![\langle W^l \rangle]\!]_{n+1} = \bigoplus_{i=0}^{N-1} [\![\langle M_i^l \rangle]\!]_{n+1}$.
  $[\![\langle b^l \rangle]\!]_{n+1} = \bigoplus_{i=0}^{N-1} [\![\langle p_i^l \rangle]\!]_{n+1}$.
  $[\![\langle [W^l]^T \rangle]\!]_{n-1} = \bigoplus_{i=0}^{N-1} [\![\langle [M_i^l]^T \rangle]\!]_{n-1}$.

• Generate partial models. For a FC or Cov layer, P_n generates a $h_{l-1} \times h_l$ or $d_l^2 c_{l-1} \times c_l$ random matrix as its partial weight M_n^l, then generates a h_l-dimensional random vector or $w_l h_l \times c_l$ random matrix as its partial bias p_n^l, where $x \in M_n^l, p_n^l$ satisfies $x \in (-\frac{1}{N}, \frac{1}{N})$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2022.3176191, IEEE Transactions on Information Forensics and Security

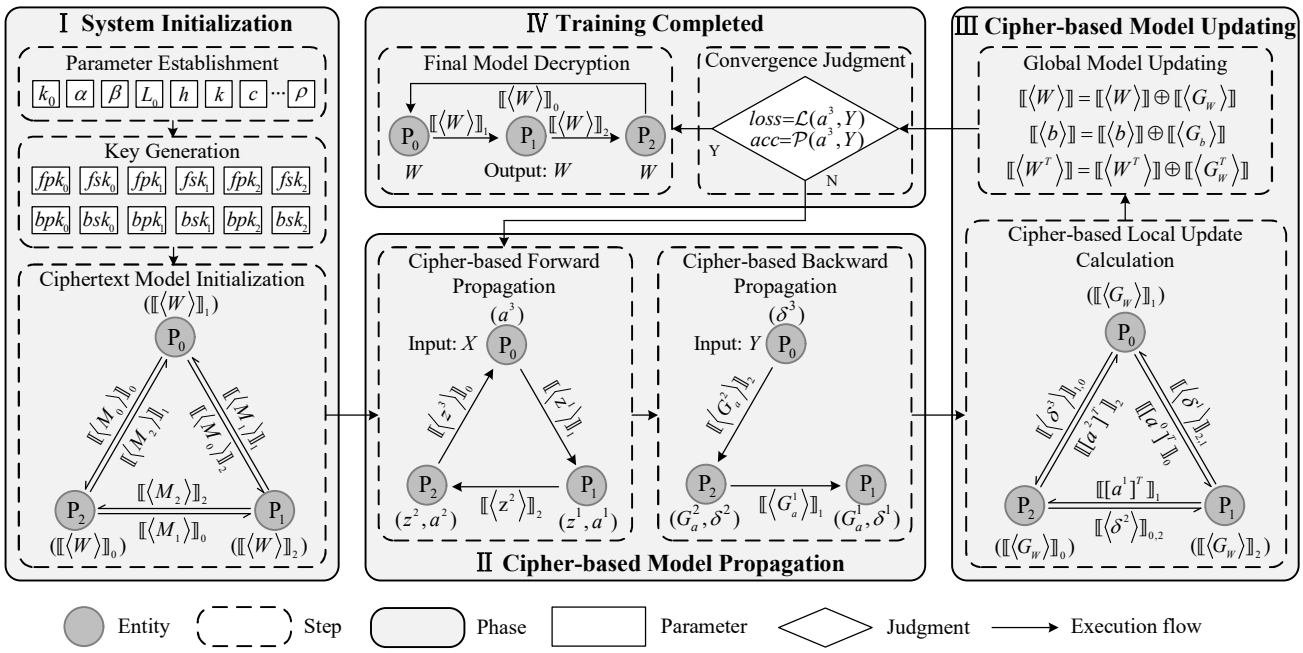

Fig. 5. The overview of PVD-FL.

• Encrypt partial models for forward propagation. In this step, M_n^l and p_n^l are signed, packed, and encrypted [1] with fpk_{i+1} for all other P_i. Specifically, $[\![\langle M_n^l \rangle]\!]_n$ signed with random signature vector SV_n^l is sent to P_{n-1} [2] for verification, and $[\![\langle M_n^l \rangle]\!]_{i+1}$ signed with zero vector ZV is sent to corresponding $\mathrm{P}_i (i \neq n-1)$.

• Encrypt partial models for backward propagation. In this step, $[M_n^l]^T$ is encrypted with bpk for backward propagation, and $[\![\langle [M_n^l]^T \rangle]\!]_n$ signed with random signature vector SV'^l_i is sent to P_{n+1}, and $[\![\langle M_n^{l\,T} \rangle]\!]_{i-1}$ signed with zero vector ZV is sent to corresponding $\mathrm{P}_i (i \neq n+1)$.

• Aggregate partial ciphertext models. After receiving all partial models from other participants, P_n calculates the global model $[\![\langle W^l \rangle]\!]_{n+1}$, $[\![\langle [W^l]^T \rangle]\!]_{n-1}$, and $[\![\langle b^l \rangle]\!]_{n+1}$ via aggregating them over ciphertexts.

In summary, after ciphertext model initialization, all participants obtain a common ciphertext model encrypted by different public keys. Specifically, W^l and b^l of each P_n are encrypted with fpk_{n+1} and signed with SV_{n+1}^l for forward propagation, and $[W^l]^T$ is encrypted with bpk_{n-1} and signed with SV'^l_{n-1} for backward propagation.

2) Cipher-based Model Propagation: At each training round, based on EVCM algorithm, all participants perform forward propagation over the ciphertext global model to obtain loss. Similarly, participants propagate the loss back for obtaining all layers' derivatives, which are used for subsequent model updating. Particularly, the calculation integrity of forward and backward propagations can be verified by EVCM.

• **Step 1: Cipher-based Forward Propagation**

[1]For simplicity, the input PP of Sign, Pack, Mul, and Unpack is omitted.

[2]The indexes of the participants are calculated with a module N, e.g., P_{1+2} is equivalent to P_0 when $N = 3$.

Algorithm 3 Cipher-based Forward Propagation

for $\mathrm{P}_{n+l} (l = 0, 1, \cdots, N)$ **do**
  if $l \neq 0$ **then**
    $a^l, VV \leftarrow \mathrm{Unpack}(\mathrm{Dec}([\![\langle z^l \rangle]\!]_{n+l}, fsk_{n+l}))$.
    if $\mathrm{Ver}(SV_{n+l}^l, VV) \neq 0$ **then**
      Propagation terminated.
    $a^l = \sigma(z^l)$.
    if the l-th layer is Cov layer **then**
      $a^l \leftarrow \rho(a^l)$.
  if $l \neq N$ **then**
    $[\![\langle z^{l+1} \rangle]\!]_{n+l+1} \leftarrow \mathrm{Mul}(a^l, [\![\langle W^{l+1} \rangle]\!]_{n+l+1}) \oplus [\![\langle b^{l+1} \rangle]\!]_{n+l+1}$
    Send $[\![\langle z^{l+1} \rangle]\!]_{n+l+1}$ to P_{n+l+1}.

In this step, every participant P_n first randomly selects β data samples X as input a^0, and the corresponding labels are written as Y. Then, as shown in Algorithm 3, given the input a^0, all participants each propagate one layer and P_n finally obtains output a^N. The forward propagation process is as follows.

• Input layer. Participant P_n propagates a^0 over the ciphertext global model $[\![\langle W^1 \rangle]\!]_{n+1}$ to obtain $[\![\langle z^1 \rangle]\!]_{n+1}$, which is sent to the next participant P_{n+1}.

• Hidden layer. Participant $\mathrm{P}_{n+l} (l \neq 0, N)$ first decrypts and unpacks received $[\![\langle z^l \rangle]\!]_{n+l}$. Then, since the calculations of the pool function ρ and activation function σ do not need the model parameters, they are calculated directly by P_{n+l}. After that, $\mathrm{P}_{n+l} (l \neq 0, N)$ executes function Mul with $[\![\langle W^{l+1} \rangle]\!]_{n+l+1}$ to obtain $[\![\langle z^{l+1} \rangle]\!]_{n+l+1}$ and sends it to the next participant P_{n+l+1}.

• Output layer. P_n decrypts $[\![\langle z^N \rangle]\!]_n$ and obtains a^N.

It is noteworthy that, the functions flattenX and clusterX introduced in Section III-A will be executed before and after a convolutional operation, which are omitted in the algorithm for simplicity. Moreover, the function Ver will be executed after every propagation to prevent miscalculations. If verification

fails, the forward propagation will terminate and no changes will be made to the global model.

• **Step 2: Cipher-based Backward Propagation**

In this step, as shown in Algorithm 4, P_n first calculates the derivative δ^N based on labels Y. Then, each participant $\mathrm{P}_{n-l}(l = 0, 1, \cdots, N-1)$ propagates back one layer and obtains the derivative G_a^{N-l} of this layer. The backward propagation process is as follows.

• Output layer. Based on $[\![\langle [W^N]^T \rangle]\!]_{n-1}$, P_n calculates $[\![\langle G_a^{N-1} \rangle]\!]_{n-1}$ by function Mul and sends it to P_{n-1}.

• Hidden layer. Participant $\mathrm{P}_{n-l}(l \neq 0)$ first decrypts and unpacks received $[\![\langle G_a^{N-l} \rangle]\!]_{n-l}$. Then, P_{n-l} calculates the derivative δ^{N-l}. After that, $[\![G_a^{N-l}]\!]$ is also calculated with $[\![\langle [W^{N-l-1}]^T \rangle]\!]_{n-l-1}$ by function Mul. Finally, $[\![\langle G_a^{N-l-1} \rangle]\!]_{n-l-1}$ is sent to the previous participant P_{n-l-1}.

• Input layer. Participant P_{n+1} decrypts $[\![\langle G_a^1 \rangle]\!]_{n+1}$ and obtains δ^1.

Similarly, VV' is also used to ensure the calculation integrity of backward propagation.

Algorithm 4 Cipher-based Backward Propagation

for $\mathrm{P}_{n-l}(l = 0, 1, \cdots, N-1)$ **do**
  if $l \neq 0$ **then**
    $G_a^{N-l}, VV' \leftarrow \mathrm{Unpack}(\mathrm{Dec}([\![\langle G_a^{N-l} \rangle]\!]_{n-l}, bsk_{n-l}))$.
    if $\mathrm{Ver}(SV'^{N-l}_{n-l}, VV') \neq 0$ **then**
      Propagation terminated.
    if the $(N-l)$-th layer is Cov layer **then**
      $G_a^{N-l} \leftarrow \mathrm{upsample}(G_a^{N-l})$.
    $\delta^{N-l} = G_a^{N-l} \odot \sigma'(z^{N-l})$.
  else
    $\delta^N \leftarrow \zeta'(a^N, Y)$.
  if $j \neq N-1$ **then**
    $[\![\langle G_a^{N-l-1} \rangle]\!]_{n-l-1} \leftarrow \mathrm{Mul}(\delta^{N-l}, [\![\langle [W^{N-l}]^T \rangle]\!]_{n-l-1})$.
    Send $[\![\langle G_a^{N-l-1} \rangle]\!]_{n-l-1}$ to P_{n-l-1}.

3) Cipher-based Model Updating: After obtaining the derivative of every layer, the ciphertext local updates are calculated between every two neighbor participants. Then, each P_n sends its local updates to others for executing ciphertext global model updating.

• **Step 1: Cipher-based Local Update Calculation**

After cipher-based model propagation, each participant $\mathrm{P}_{n+l}(l = 0, 1, \cdots, N-1)$ obtains δ^l and a^l.

In this step, as shown in Algorithm 5, P_{n+l} first encrypts packed $\langle \delta^l \rangle$ with all $fpk_i (i \neq n+l-1)$ and send them to P_{n+l-1}. Similarly, $\langle a^l \rangle$ is encrypted with all $bpk_i (i \neq n+l+1)$, which is then sent to P_{n+l+1}. Then, unpacked $[\delta^l]^T$ and $[a^l]^T$ are encrypted with fpk_{n+l} and bpk_{n+l}, and are sent to P_{n+l-1} and P_{n+l+1} respectively.

After receiving all ciphertexts from others, P_{n+l} calculates local updates $[\![\langle G_W^{l+1} \rangle]\!]_i$ and $[\![\langle [G_W^l]^T \rangle]\!]_i$, and sends them to corresponding P_{i-1} and P_{i+1}.

Finally, G_b^l is calculated with δ^l by a single participant P_{n+l}, which then is packed and encrypted with all fpk_i and sent to P_{i-1}.

Algorithm 5 Cipher-based Local Update Calculation

for $\mathrm{P}_{n+l}(l = 0, 1, \cdots, N-1)$ **do** ▷ *Encrypt δ^l and a^l.*
  $\langle \delta^l \rangle \leftarrow \mathrm{Pack}(\mathrm{Sign}(\delta^l, ZV))$.
  for $i \neq n+l-1$ **do**
    Send $[\![\langle \delta^l \rangle]\!]_i \leftarrow \mathrm{Enc}(\langle \delta^l \rangle, fpk_i)$ to P_{n+l-1}.
  Send $[\![[\delta^l]^T]\!]_{n+l} \leftarrow \mathrm{Enc}([\delta^l]^T, bpk_{n+l})$ to P_{n+l-1}.
  $\langle a^l \rangle \leftarrow \mathrm{Pack}(\mathrm{Sign}(a^l, ZV))$.
  for $i \neq n+l+1$ **do**
    Send $[\![\langle a^l \rangle]\!]_i \leftarrow \mathrm{Enc}(\langle a^l \rangle, bpk_i)$ to P_{n+l+1}.
  Send $[\![[a^l]^T]\!]_{n+l} \leftarrow \mathrm{Enc}([a^l]^T, fpk_{n+l})$ to P_{n+l+1}.
for $\mathrm{P}_{n+l}(l = 0, 1, \cdots, N-1)$ **do** ▷ *Calculate local updates.*
  for $i \neq n+l$ **do**
    $[\![\langle G_W^{l+1} \rangle]\!]_i \leftarrow \mathrm{Mul}(\frac{\alpha}{\beta N}[a^l]^T, [\![\langle \delta^{l+1} \rangle]\!]_i)$.
    Send $[\![\langle G_W^{l+1} \rangle]\!]_i$ to P_{i-1}.
    $[\![\langle [G_W^l]^T \rangle]\!]_i \leftarrow \mathrm{Mul}(\frac{\alpha}{\beta N}[\delta^l]^T, [\![\langle a^{l-1} \rangle]\!]_i)$.
    Send $[\![\langle [G_W^l]^T \rangle]\!]_i$ to P_{i+1}.
  $[\![\langle G_W^l \rangle]\!]_{n+l-1} \leftarrow \mathrm{Mul}([\![[a^{l-1}]^T]\!]_{n+l-1}, \frac{\alpha}{\beta N}\langle \delta^l \rangle)$.
  Send $[\![\langle G_W^l \rangle]\!]_{n+l-1}$ to P_{n+l-2}.
  $[\![\langle [G_W^{l+1}]^T \rangle]\!]_{n+l+1} \leftarrow \mathrm{Mul}([\![[\delta^{l+1}]^T]\!]_{n+l+1}, \frac{\alpha}{\beta N}\langle a^l \rangle)$.
  Send $[\![\langle [G_W^{l+1}]^T \rangle]\!]_{n+l+1}$ to P_{n+l+2}.
  for $i = 0, 1 \cdots, N-1$ **do**
    $G_b^l = \sum \delta^l$.
    Send $[\![\langle G_b^l \rangle]\!]_i \leftarrow \mathrm{Enc}(\langle G_b^l \rangle, fpk_i)$ to P_{i-1}.

After Step 1, each P_{n+l} obtains the local updates $[\![\langle G_W \rangle]\!]$, $[\![\langle [G_W]^T \rangle]\!]$, and $[\![\langle G_b \rangle]\!]$ of all layers. Then, based on the local updates of all layers, its global model is updated as

$$\begin{cases} [\![\langle W^l \rangle]\!] = [\![\langle W^l \rangle]\!] \oplus [\![\langle G_W^l \rangle]\!], \\ [\![\langle [W^l]^T \rangle]\!] = [\![\langle [W^l]^T \rangle]\!] \oplus [\![[G_W^l]^T]\!], \\ [\![\langle b^l \rangle]\!] = [\![\langle b^l \rangle]\!] \oplus [\![\langle G_b^l \rangle]\!], (l = 1, 2, \cdots, N). \end{cases}$$

Moreover, the correct calculation can also be ensured during the model updating phase. Once a participant executes a wrong or incomplete calculation, the updated global model will not satisfy the verification in the next round of model propagation.

So far the global model is trained with the data of P_n for one round. Notably, we only introduce the training process of P_n and others train the model in the same way, which can be executed parallelly.

4) Training Completed: On finishing certain rounds of model training, every P_n will judge whether the global model converges based on the loss and accuracy. Then, when is judged to converge by all participants, the ciphertext global model is decrypted for providing services.

• **Step 1: Convergence Judgment**

Since the global model is kept confidential during the whole training process, it is difficult to execute convergence judgment with test data. According to the i.i.d. assumption of training data, we judge the model convergence simply by loss value or prediction accuracy with training data.

For each participant P_n, it will obtain the loss value and prediction accuracy of its one data batch after every forward propagation. Since a data batch may not be representative, the average loss and prediction accuracy in multi-round training is used to judge the model convergence.

When P_n judges the global model has converged, it will stop model training with its data but execute model updating as usual. Until every participant judges the model has converged,

the model training process stops and the final ciphertext global model is obtained by all participants.

• **Step 2: Final Model Decryption**

In this step, every participant P_n sends its ciphertext global model $[\![W^l]\!]_{n+1}$ and $[\![b^l]\!]_{n+1}(l = 1, 2, \cdots, N)$ to the next participant P_{n+1}. Finally, P_{n+1} decrypts it with its private key fsk_{n+1} and obtains the plaintext global model.

V. Security Analysis

In this section, we first prove the verification correctness briefly. Then, we prove the confidentiality of PVD-FL under the honest-but-curious assumption. Finally, we show that PVD-FL can well resist the existing and potential inference attacks.

A. Verification Correctness

During iterative model training, a participant may execute incomplete or incorrect calculations, which will impact the integrity of the model training. Benefiting from the verification ability of the EVCM algorithm, PVD-FL allows every participant to verify the calculation correctness of every model propagation or updating step. Here we just prove the verification correctness of every forward propagation in the following, and the proofs of backward propagation and model updating are similar to it and are omitted.

Theorem 1. *During each forward propagation, output* $b = 0$ *of function* Ver *proves that the result* $[\![\langle z\rangle]\!]_n$ *is calculated correctly by* P_{n-1}.

Proof. In the system initialization phase, for P_{n-1}, only $[\![\langle M_n\rangle]\!]_n$ is signed with random vector SV, and others are signed with zero vector ZV, where SV is generated and saved by P_n. The global model is calculated by $[\![\langle W\rangle]\!]_n = \bigoplus_{i=0}^{N-1}[\![\langle M_i\rangle]\!]_n = [\![\langle\sum_{i=0}^{N-1} M_i\rangle]\!]_n$, so $[\![\langle W\rangle]\!]_n$ is also signed with SV.

In the cipher-based model updating phase, $[\![\langle W\rangle]\!]_n$ will be updated by $[\![\langle W\rangle]\!]_n = [\![\langle W\rangle]\!]_n \oplus [\![\langle G_W\rangle]\!]_n = [\![\langle W + G_W\rangle]\!]_n$. Since $[\![\langle G_W\rangle]\!]_n$ is signed with zero vector ZV, $[\![\langle W\rangle]\!]_n$ is always signed with SV during the whole training process. Similarly, $[\![\langle b\rangle]\!]_n$ is always signed with ZV.

In PVD-FL, $[\![\langle W\rangle]\!]_n$ and $[\![\langle b\rangle]\!]_n$ are obtained by P_{n-1}. At each forward propagation, $[\![\langle z\rangle]\!]_n$ is calculated by $[\![\langle z\rangle]\!]_n \leftarrow \mathrm{Mul}(a, [\![\langle W\rangle]\!]_n) \oplus [\![\langle b\rangle]\!]_n$. Therefore, each element $[\![\langle z\rangle_{v,s}]\!]_n \in [\![\langle z\rangle]\!]_n$ is calculated as

$$
\begin{aligned}
&[\![\langle z\rangle_{v,s}]\!]_n \\
=&\bigoplus_{i=0}^{h-1}[\![\langle W\rangle_{i,s}]\!]_n \otimes a_{v,i} \oplus [\![\langle b\rangle_n]\!]_n \\
=&[\![\sum_{i=0}^{h-1} a_{v,i}(\sum_{j=1}^{N_s-1} w_{i,sN_s+j}\cdot 2^{jL_2} + sv_{i,s}) + \sum_{j=1}^{N_s-1} b_{sN_s+j}\cdot 2^{jL_2}]\!]_n \\
=&[\![\sum_{j=1}^{N_s-1}(\sum_{i=0}^{h-1} a_{v,i}w_{i,sN_s+j} + b_{sN_s+j})\cdot 2^{jL_2} + \sum_{i=0}^{h-1} a_{v,i}sv_{i,s}]\!]_n,
\end{aligned}
$$

where h is the column number of a. According to $L_2 > 4L_0 + 3log(U) - 3$ and $L_1 > 2L_0 + log(U) - 1$, each $[\![\langle z\rangle_{v,s}]\!]_n$ can be correctly decrypted with fsk_n and unpacked by P_n. After that, P_n obtains the verification value $vv_s = \sum_{i=0}^{h-1} a_{\beta,i}sv_{i,s} = \sum_{i=0}^{h-1} sv_{i,s}$.

Considering the condition of incomplete calculation, we assume that only one element $[\![\langle W\rangle_{k,s}]\!]_n \otimes a_{v,k}$ is omitted while calculating $[\![\langle z\rangle_{v,s}]\!]_n$. Then, the verification value vv_n is obtained as $vv_s = \sum_{i=0,i\neq k}^{h-1} sv_{i,s}$, and the verification will fail according to $sv_{k,s} \neq 0$. Therefore, even though only one element is omitted while executing our cipher-based forward propagation, the incomplete calculation result will not satisfy the verification.

Considering the other condition of incorrect calculation, similarly, we assume that only one element is changed while calculating $[\![\langle z\rangle_{v,s}]\!]_n$. If $[\![\langle W\rangle_{k,s}]\!]_i$ is changed, $sv_{j,s}$ will change to a random number, and the verification will fail. Therefore, the incorrect calculation result will also not satisfy the verification.

In summary, if $vv_s = \sum_{i=0}^{h-1} sv_{i,s}$ is verified by P_n, i.e., $b = 0$, this forward propagation is verified as correct execution. □

B. The Security of PVD-FL framework

In this subsection, we prove that the global model and local update are confidential in PVD-FL under the honest-but-curious assumption.

Theorem 2. *Local update* G_W *is confidential at each round of cipher-based model updating.*

Proof. At each cipher-based model updating, ciphertext local update $[\![\langle G_W\rangle]\!]$ is calculated between neighbor participants.

As shown in Algorithm 5, for the l-th layer' local update $[\![\langle G_W^l\rangle]\!]$, P_{n+l} sends $[\![\langle\delta^l\rangle]\!]_i$ to P_{n+l-1}, which is encrypted with $fpk_i (i \neq n + l - 1)$. Through calculating $\mathrm{Mul}(\frac{\alpha}{\beta N}[a^{l-1}]^T, [\![\langle\delta^l\rangle]\!]_i)$, $[\![\langle G_W^l\rangle]\!]_i$ is obtained. Since it is equivalent to solving $(\mathcal{L}, p)$-based decision hard problem to distinguish ciphertexts encrypted by SHE, the SHE technique is semantically secure against chosen-plaintext attack (CPA) [33], [34]. Moreover, according to the non-collusion assumption defined in our threat model, P_{n+l-1} cannot obtain private keys of other participants. Thus, if the $(\mathcal{L}, p)$-based decision problem is intractable in polynomial time, $[\![\langle\delta^l\rangle]\!]_i$ and $[\![\langle G_W^l\rangle]\!]_i$ are confidential for P_{n+l-1}.

Similarly, $[\![\langle G_W^l\rangle]\!]_{n+l-1}$ is calculated by P_{n+l} with $[\![[a^{l-1}]^T]\!]_{n+l-1}$ and $\frac{\alpha}{\beta N}\langle\delta^l\rangle$, which is also confidential for P_{n+l} based on the IND-CPA security of SHE.

Therefore, G_W of each layer is confidential at each cipher-based model updating. □

Theorem 3. *Global model* W *is confidential during the system initialization and cipher-based model training phases.*

Proof. During system initialization, each participant P_n receives ciphertext partial models $[\![\langle M_i\rangle]\!]_{n+1}(i = 0, \cdots, N-1)$, and obtains $[\![\langle W\rangle]\!]_{n+1}$ by aggregating them over ciphertexts, which based on SHE's IND-CPA security is also confidential for P_n. Thus, after system initialization, each participant obtains a ciphertext global model $[\![\langle W\rangle]\!]$.

Then, at each model training round, the ciphertext global model $[\![\langle W\rangle]\!]_{n+1}$ of P_n is updated with $[\![\langle G_W\rangle]\!]_{n+1}$, which

is also confidential for P_n according to Theorem 2. Thus, the ciphertext global model $[\![\langle W \rangle]\!]_{n+1}$ is always updated over ciphertexts until convergence.

Therefore, global model W is always confidential during the whole system initialization and model training phases. □

C. Discussion of Inference Attacks

In this subsection, we first introduce the existing inference attacks briefly, which contain the inference of properties, membership, training inputs, and labels [5]. Then, we analyze our PVD-FL against these inference attacks. Moreover, we analyze in detail the possibility of launching a potential inference attack from the parameters leaked in PVD-FL.

• *Property inference.* The property inference attack allows an adversary participant to infer others' data property with observed local updates and its auxiliary training data [8]. Specifically, the adversary participant may train a binary property classifier, which could classify a data sample into its corresponding property according to its calculated local updates.

• *Membership inference.* The membership inference attack can infer whether a certain data sample is used to train the model [37]. Specifically, given a data batch, the parameters of an embedding layer are updated only when certain data appear in this batch, which reveals the information of training data.

• *Inferring training inputs and labels.* There is an optimization algorithm called Deep Leakage from Gradients (DLG) [6] that can infer the training inputs and labels with the observed local updates and model parameters in just a few rounds. Specifically, the adversary participant will generate dummy data and optimize them iteratively to fit the observed local updates calculated by others' training data, and the accurate raw data will be recovered finally.

Analysis of existing inference attacks. The existing attacks infer the data information mainly based on the local update, global model, or both of them. However, in PVD-FL, as proved above, the local updates and global model are kept confidential in the whole iterative training process. Therefore, it can be simply proved that these inference attacks are all invalid in PVD-FL.

Analysis of potential inference attacks. In PVD-FL, the leaked parameters contain the model training hyperparameters, model size parameters, and intermediate parameters z and G_a calculated during model propagation.

First, the first two types of parameters are considered non-sensitive since they do not contain any data information.

Then, for intermediate parameters z^l of P_{n+l}, they are calculated with $z^l = a^{l-1}W^l + b^l$, where W^l and b^l are confidential and changing in the whole model training process. Meanwhile, a^{l-1} is held by P_{n+l-1} and is different at each training round, which is unknown for P_{n+l} under the non-collusion assumption. Thus, leaked z^l cannot be used to infer any data information. Similarly, for P_{n-l}, G_a^{n-l} is just a matrix product with unknown and changing δ^{n-l+1} and $[W^{n-l+1}]^T$, which also cannot launch inference attacks.

Moreover, we consider another stronger inference attack similar to DLG. The adversary participant may also generate dummy data to fit z and G_a. However, this inference attack requires propagating the dummy data over the global model for multiple rounds, which is impracticable in PVD-FL. Specifically, since the global model is confidential in PVD-FL, it is impossible to execute model propagation locally. Meanwhile, even if the adversary participant inputs dummy data for collaborative propagation at every training round, the calculated z and G_a with dummy data are not at the same layer as that observed, which cannot be used to update the dummy data for fitting other's training data.

In summary, PVD-FL can resist the existing and potential inference attacks well.

VI. Performance Evaluation

In this section, we first evaluate the efficiency of our proposed EVCM algorithm. Then, we present the experimental results of PVD-FL on four real-world datasets, and analyze its performance in terms of the model accuracy, computational cost, and communication overhead. Moreover, an MPC-based two-party neural network training scheme, namely QUOTIENT [19], is used to compare with PVD-FL from the above aspects.

A. Experimental Environment

In order to evaluate the performance of PVD-FL, the experiments are conducted over Dell Precision 7920 machines equipped with 256.0 GB RAM and Intel(R) Xeon(R) Gold 6226R 2.90GHz processor, running Ubuntu 20.04. The machines are connected in a LAN environment with an average latency of 0.3ms and a bandwidth of 1.8GB/s.

It should be noted that this network configuration is to facilitate a fair performance comparison with QUOTIENT, and indeed PVD-FL has little dependence on communication bandwidth due to its low communication overhead, which will be explained in Section VI-C4.

We use Python 3.8.1 and the Numpy 1.17.0 package to implement EVCM algorithm and evaluate the time costs of different functions with different SHE security parameters (k_0, k_1, k_2), bit length L_0 and matrix size D. For simplicity, EVCM is evaluated with random square matrices of size $D = 64, 128, 256$, or 512. We set the bit length L_0 of elements in matrices as $10, 15, 20$, or 25, and set the security parameters (k_0, k_1, k_2) as $(3072, 600, 768)$ and $(6144, 1300, 1536)$.

Moreover, we implement our PVD-FL framework based on EVCM, and evaluate the model accuracy, computational cost, and communication overhead on four real-world datasets. In PVD-FL, the parameters are set as $k_0 = 6144, L_0 = 15, L_1 = 38, L_2 = 84$, and $N_s = 15$, and we use multiple machines to execute protocol process of different participants.

B. Data-independent benchmarking of EVCM

In this section, we present the time costs of the EVCM algorithm in Fig. 6, which will be used to execute the most basic calculations of model training in PVD-FL.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2022.3176191, IEEE Transactions on Information Forensics and Security

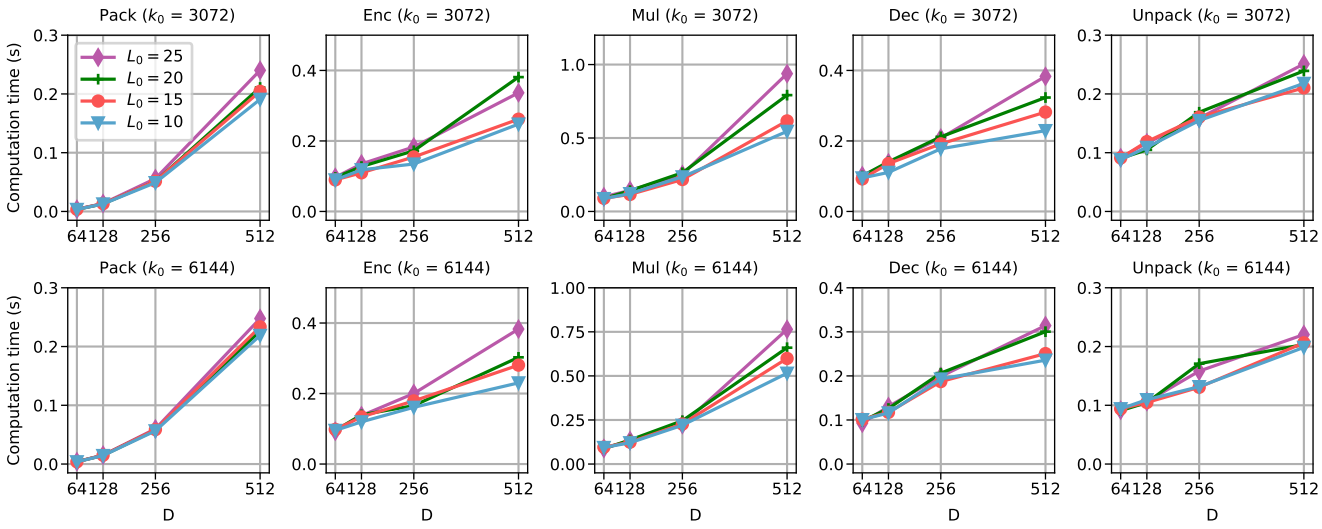

Fig. 6. Time cost of EVCM.

1) Time costs with different security parameters k_0: Through comparing the subfigures in the first and second rows, we find that the time costs of $k_0 = 3072$ are markedly higher than that of $k_0 = 6144$. Specifically, the increased security parameter k_0 brings linear growth of time cost, but it also brings more plaintext slots, which causes a decrease of the total time costs. Therefore, in PVD-FL, we choose $k_0 = 6144$ to reach both security and efficiency.

2) Time costs with different bit length L_0: From a single subfigure in Fig. 6, we find that the time costs increase with the growth of the bit length L_0. In EVCM algorithm, smaller L_0 means smaller plaintext space L_2 and more plaintext slots, which will reduce the number of homomorphic additions and multiplications and thus reduce the total time costs. In PVD-FL, the bit length L_0 denotes the precision of training data and model parameters, and we choose $L_0 = 15$ to balance the accuracy and efficiency.

3) Time costs of different functions: Through observing the time costs of different functions, we can find that there is a difference in the time consumption of different functions. Specifically, functions Pack and Unpack are less time-consuming and even can be ignored, but function Mul is time-consuming. Therefore, in PVD-FL, the training protocols should be carefully designed, and the calculation times of function Mul should be minimized.

C. Experiments of PVD-FL on real-world datasets

In this section, we evaluate the model accuracy, computational cost, and communication overhead of PVD-FL on four real-world datasets, then make a comparison with QUOTIENT.

1) Datasets and model architectures: Our PVD-FL is evaluated with MNIST [38], Thyroid [39], Breast cancer [40], and German credit [41] datasets. The training task on all datasets is to train a DNN model that could make accurate classifications to corresponding data samples, and the training data is divided evenly to N participants. The introduction of these datasets is as follows, and the corresponding model architecture and parameter setting are shown in TABLE II.

TABLE II
MODEL ARCHITECTURE AND PARAMETER SETTING IN PVD-FL.

dataset	architecture [1]	α	β	N
MNIST	32 Cov 5 $\rightarrow$ MP 2 $\rightarrow$ 64 Cov 5 $\rightarrow$ MP 2 $\rightarrow$ 512 FC $\rightarrow$ 10 CEE	1.0	32	4
	$3 \times$ (128 FC) $\rightarrow$ 10 CEE	1.0	32	4
	$3 \times$ (512 FC) $\rightarrow$ 10 CEE	1.0	32	4
Thyroid	$2 \times$ (100 FC) $\rightarrow$ 3 MSE	0.01	32	3
Breast cancer	$3 \times$ (512 FC) $\rightarrow$ 2 MSE	0.01	32	4
German credit	$2 \times$ (124 FC) $\rightarrow$ 2 CEE	0.001	32	3

• *MNIST.* The handwritten digits recognition (MNIST) dataset contains 60000 training and 10000 testing 28×28 gray images of different handwritten digits from 0 to 9, and each label is a 10-dimension one-hot vector.

• *Thyroid.* The thyroid dataset contains 3772 training and 3428 testing data, where each data sample consists of 21 features and a label (normal, hyperfunction, and subnormal).

• *Breast cancer.* The breast cancer dataset contains 5547 50×50 RGB images of the breast cancer histopathology, and each data sample of a patient is classified into invasive ductal or non-invasive ductal carcinoma. In our experiments, the dataset is divided into 5000 training and 1547 testing data.

• *German credit.* The German credit dataset contains 1000 data of bank account holders, which are classified to have good or bad credit, and each data sample has 20 attributes. We use the $8 : 2$ split for training and testing, respectively, and the data are normalized before training.

2) Model Accuracy: For analyzing the model accuracy, we plot the model convergence curves of PVD-FL in Fig. 7, and also plot the curves of QUOTIENT, centralized training, and local training for comparison. Specifically, centralized training means executing model training in a center that collects all training data. In local training, the model is trained with just the local training data owned by one participant.

[1]MP, CEE, and MSE denote max pooling, cross entropy error, and mean square error respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2022.3176191, IEEE Transactions on Information Forensics and Security

TABLE III
MODEL ACCURACY OF DIFFERENT DATASETS AND DNN ARCHITECTURES AFTER 1, 5, AND 10 TRAINING EPOCHS USING PVD-FL AND QUOTIENT.

Epoch	MNIST						Thyroid		Breast cancer		German credit	
	$3 \times (128\,\mathrm{FC})$		$3 \times (512\,\mathrm{FC})$		CNN		$2 \times (100\,\mathrm{FC})$		$3 \times (512\,\mathrm{FC})$		$2 \times (124\,\mathrm{FC})$	
	PVD-FL	QUO.	PVD-FL	QUO.	PVD-FL	QUO.	PVD-FL	QUO.	PVD-FL	QUO.	PVD-FL	QUO.
1	91.49%	90.23%	94.63%	94.24%	96.54%	88.16%	91.86%	24.80%	56.12%	49.40%	31.00%	41.00%
5	96.84%	95.36%	97.54%	97.45%	97.54%	98.72%	94.06%	93.41%	74.77%	73.60%	46.00%	72.50%
10	97.37%	96.04%	97.94%	98.12%	98.47%	98.89%	94.90%	94.53%	77.14%	76.00%	80.05%	79.00%

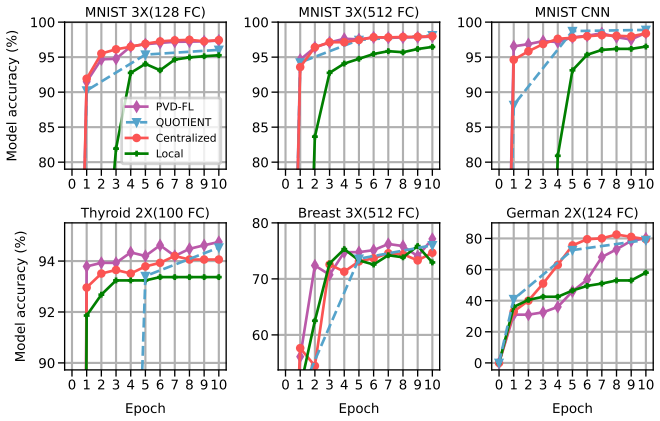

Fig. 7. Model accuracy comparison with PVD-FL, QUOTIENT, centralized training, and local training.

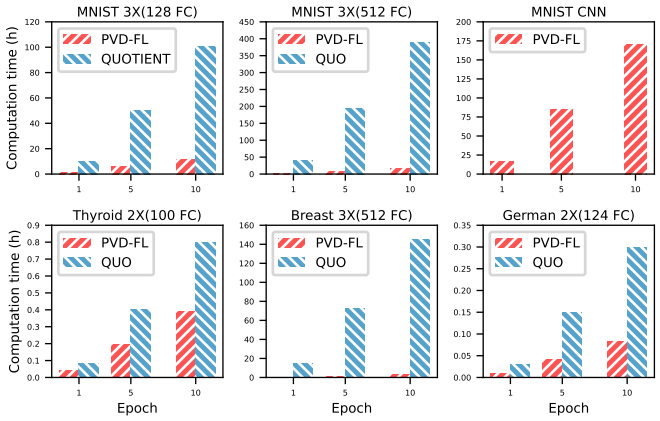

Fig. 8. Computational cost comparison of PVD-FL and QUOTIENT.

From Fig. 7, we find that the model accuracy after ten epochs is almost the same for PVD-FL, QUOTIENT, and centralized training, which is significantly higher than the accuracy under local training. Moreover, the model under PVD-FL, QUOTIENT, and centralized training converge at almost the same speed. Therefore, the experimental results show that our PVD-FL causes no loss of model accuracy and convergence speed, which further demonstrates that PVD-FL can be used in practical environments. For showing more clearly, the model accuracy comparison results of different datasets are listed in TABLE III.

3) Computational Cost: Fig. 8 shows the computational costs of our PVD-FL and QUOTIENT on different datasets and DNN architectures.

For training the same $3 \times (512\,\mathrm{FC})$ DNN models, the model training speed on MNIST dataset is lower than on breast cancer dataset, since the training data size of MNIST dataset is more than that of breast cancer dataset. For different DNN architectures on the same MNIST dataset, the computational cost of $3 \times (512\,\mathrm{FC})$ is almost 4 times that of $3 \times (128\,\mathrm{FC})$, which shows that the computational cost increases linearly with the growth of the model scale.

Through observing each subfigure, we find that the computational costs increase linearly with the growth of training epochs for both PVD-FL and QUOTIENT. Moreover, we can find that PVD-FL's computational costs are markedly lower than QUOTIENT's when training FC models, e.g., for a $3 \times (128\,\mathrm{FC})$ model, it only takes one hour to perform one epoch model training in PVD-FL but takes more than 10 hours in QUOTIENT under the same LAN environment. For CNN model training, there is no total computational cost provided by QUOTIENT, but it only takes 17 hours to execute one epoch training for a LeNet-5 model in PVD-FL, which first makes it practical to execute privacy-preserving decentralized model training for the state-of-the-art CNN model.

4) Communication Overhead: Fig. 9 shows PVD-FL's communication overhead of different steps for a single participant. For the same MNIST dataset, the communication overhead of model initialization and decryption for training CNN models is less than that for training DNN models, since the number of CNN's model parameters (i.e., convolutional kernel) is less than DNN's parameters. But the model training communication overhead of CNN models is higher than that of DNN models, since the convolutional operation brings massive intermediate parameters during model propagation. For different datasets, the communication overhead of MNIST dataset is markedly higher than others for all steps, which is because the feature number (784) of MNIST dataset is markedly more than that of other datasets.

Overall, the communication overhead of PVD-FL is indeed low, e.g., it just takes a participant less than 7 MB or 16 MB for executing one round of $3 \times (512\,\mathrm{FC})$ or CNN model progagation, which shows that PVD-FL has low requirements for the network bandwidth and can be used in most environments.

VII. RELATED WORK

In this section, we briefly introduce some related works about privacy-preserving, verifiable, and decentralized FL.

A. Privacy-preserving FL.

Although FL makes it possible to construct a global model without collecting participants' raw training data, the ex-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2022.3176191, IEEE Transactions on Information Forensics and Security

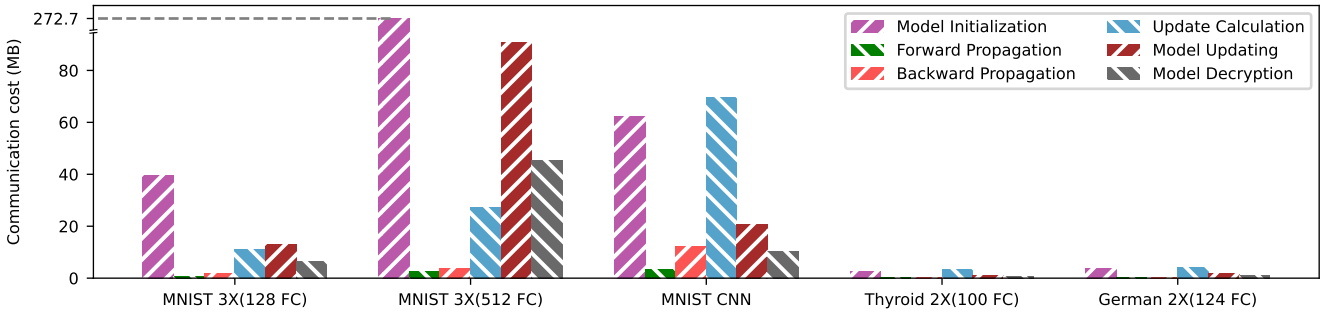

Fig. 9. Communication overhead of a single participant.

changed global model and local updates still contain massive data information. Based on the double mask method, Bonawitz *et al.* [9] designed a secure aggregation algorithm to aggregate the local updates without leaking any single local updates, which can protect the local updates from inference attacks, but the global model is still exposed to every participant in plaintext. Truex *et al.* [15] proposed a scheme named LDP-Fed to ensure privacy-preserving through adding random noises into local updates, which can provide a formal differential privacy guarantee for iterative collection of local updates in FL, but it will bring a trade-off of accuracy loss. Moreover, Agrawal *et al.* [19] proposed a two-party secure DL scheme named QUOTIENT through designing novel MPC protocols, and improved its efficiency by applying layer normalization and adaptive gradient methods, but its computation and communication costs are still high and unacceptable.

B. Verifiable FL.

During each training round of FL, the center may falsify the aggregation results deliberately or unintentionally, which will affect the model accuracy. Considering this problem, Xu *et al.* [21] designed VrifyNet based homomorphic hash function and double mask method, which enables participants to verify whether the center aggregates correctly. Based on Lagrange interpolation and homomorphic encryption, Fu *et al.* [22] also designed a scheme named VFL to verify the correctness of the aggregation results and can keep constant regardless of the number of participants. Similarly, Zhang *et al.* [23] utilized the Chinese Remainder Theorem and the Paillier encryption to achieve secure aggregation, and used bilinear signature technology to verify the aggregation correctness.

C. Decentralized FL.

In centralized FL, it is difficult to judge whether a center is trustworthy and to find one trusted by all participants, meanwhile, the communication ability between the center and participants may become the system bottleneck. Therefore, Weng *et al.* [42] proposed a privacy-preserving decentralized FL framework named DeepChain to force the participants to behave correctly based on Blockchain. Lyu *et al.* [24] proposed FPPDL to achieve fair and privacy-preserving model training under a decentralized framework, where the local updates are protected by three-layer onion-style encryption. Jeon *et al.* [25] proposed a privacy-preserving decentralized aggregation protocol based on the novel group-based communication pattern, which can provide a privacy guarantee under an honest-but-curious model, meanwhile, control the communication overhead during each training round.

Different from the above-mentioned schemes, PVD-FL can balance privacy and efficiency, and can guarantee training integrity under a decentralized architecture. Specifically, based on EVCM, PVD-FL protects the global model and local updates well during the whole model training process, which provides strong privacy preservation. Moreover, PVD-FL achieves integrity verification of every model training step.

VIII. Conclusion

In this paper, we have proposed PVD-FL, a privacy-preserving and verifiable decentralized FL framework, which first guarantees both privacy-preservation and training integrity under a decentralized architecture. Security analysis demonstrates that PVD-FL can achieve the confidentiality and integrity of model training, and can resist various existing and potential inference attacks. Meanwhile, experimental results on real-world datasets show the lossless accuracy and practical performance of PVD-FL. In addition, the poisoning attack will also destroy the model integrity and should be verified, which might be explored in future studies.

References

[1] Y. Bengio, Y. LeCun, and G. E. Hinton, "Deep learning for AI," *Commun. ACM*, vol. 64, pp. 58–65, 2021.
[2] European Parliament and Council of the European Union. (2016, April) General Data Protection Regulation. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj/
[3] California State Legislature. (2018, June) California Consumer Privacy Act of 2018. [Online]. Available: https://oag.ca.gov/privacy/ccpa/
[4] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, pp. 1–18, 2016.
[5] L. Lyu, H. Yu, J. Zhao, and Q. Yang, "Threats to federated learning," in *Federated Learning*, 2020, vol. 12500, pp. 3–16.
[6] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *NeurIPS*, 2019, pp. 14 747–14 756.
[7] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *CVPR*, 2021, pp. 16 337–16 346.
[8] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *IEEE Symposium on Security and Privacy*, 2019, pp. 691–706.
[9] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *CCS*, 2017, pp. 1175–1191.

[10] X. Zhang, S. Ji, H. Wang, and T. Wang, "Private, yet practical, multiparty deep learning," in *ICDCS*, 2017, pp. 1442–1452.

[11] F. Wang, H. Zhu, R. Lu, Y. Zheng, and H. Li, "Achieve efficient and privacy-preserving disease risk assessment over multi-outsourced vertical datasets," *IEEE Trans. Dependable Secur. Comput.*, pp. 1–14, 2020.

[12] F. Wang, H. Zhu, R. Lu, Y. Zheng, and H. Li, "A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent," *Inf. Sci.*, vol. 552, pp. 183–200, 2021.

[13] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*, 2016, pp. 308–318.

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

[15] S. Truex, L. Liu, K. H. Chow, M. E. Gursoy, and W. Wei, "LDP-Fed: federated learning with local differential privacy," in *EuroSys*, 2020, pp. 61–66.

[16] M. Kim, O. Günlü, and R. F. Schaefer, "Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication," in *ICASSP*, 2021, pp. 2650–2654.

[17] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *IEEE Symposium on Security and Privacy*, 2017, pp. 19–38.

[18] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "Deepsecure: scalable provably-secure deep learning," in *DAC*, 2018, pp. 1–6.

[19] N. Agrawal, A. S. Shamsabadi, M. J. Kusner, and A. Gascón, "QUOTIENT: two-party secure neural network training and prediction," in *CCS*, 2019, pp. 1231–1247.

[20] H. Chaudhari, R. Rachuri, and A. Suresh, "Trident: Efficient 4pc framework for privacy preserving machine learning," in *NDSS*, 2020, pp. 1–26.

[21] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 911–926, 2020.

[22] A. Fu, X. Zhang, N. Xiong, Y. Gao, and H. Wang, "VFL: A verifiable federated learning with privacy-preserving for big data in industrial iot," *IEEE Trans. Industr. Inform.*, pp. 1–11, 2020.

[23] X. Zhang, A. Fu, H. Wang, C. Zhou, and Z. Chen, "A privacy-preserving and verifiable federated learning scheme," in *ICC*, 2020, pp. 1–6.

[24] L. Lyu, J. Yu, K. Nandakumar, Y. Li, X. Ma, J. Jin, H. Yu, and K. S. Ng, "Towards fair and privacy-preserving federated deep models," *IEEE Trans. Parallel Distributed Syst.*, vol. 31, pp. 2524–2541, 2020.

[25] B. Jeon, S. M. Ferdous, M. R. Rahman, and A. Walid, "Privacy-preserving decentralized aggregation for federated learning," in *INFOCOM Workshops*, 2021, pp. 1–6.

[26] L. Cui, Y. Qu, G. Xie, D. Zeng, R. Li, S. Shen, and S. Yu, "Security and privacy-enhanced federated learning for anomaly detection in iot infrastructures," *IEEE Trans. Ind. Informatics*, pp. 1–10, 2021.

[27] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 307–328.

[28] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[29] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[30] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination press San Francisco, CA, 2015, vol. 25.

[31] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *Tenth international workshop on frontiers in handwriting recognition*. Suvisoft, 2006, pp. 1–6.

[32] H. Mahdikhani, R. Lu, Y. Zheng, J. Shao, and A. A. Ghorbani, "Achieving o ($\log^3$n) communication-efficient privacy-preserving range query in fog-based iot," *IEEE Internet of Things J.*, vol. 7, no. 6, pp. 5220–5232, 2020.

[33] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity range query over encrypted time series data," *IEEE Trans. Dependable Secur. Comput.*, pp. 1–15, 2021.

[34] Y. Guan, R. Lu, Y. Zheng, S. Zhang, J. Shao, and G. Wei, "Toward privacy-preserving cybertwin-based spatiotemporal keyword query for ITS in 6G era," *IEEE Internet Things J.*, vol. 8, pp. 16 243–16 255, 2021.

[35] Z. Brakerski, C. Gentry, and S. Halevi, "Packed ciphertexts in lwe-based homomorphic encryption," in *Public Key Cryptography*, vol. 7778, 2013, pp. 1–13.

[36] N. P. Smart and F. Vercauteren, "Fully homomorphic SIMD operations," *Des. Codes Cryptogr.*, vol. 71, pp. 57–81, 2014.

[37] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symposium on Security and Privacy*, 2017, pp. 3–18.

[38] Y. LeCun. (1998) The mnist database of handwritten digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[39] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, pp. 81–106, 1986.

[40] A. Janowczyk and A. Madabhushi, "Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases," *Journal of pathology informatics*, vol. 7, pp. 1–19, 2016.

[41] H. Hofmann. (2000) Statlog (german credit data) dataset. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

[42] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, pp. 2438–2455, 2021.

Jiaqi Zhao was born in China, in 1997. He received the B.Eng. degree in information security from Xidian University, Xi'an, Shaanxi, China, in 2020. He is currently pursuing the Ph.D. degree in cyberspace security at Xidian University, Xi'an, Shaanxi, China.

His research has been concerned with privacy-preserving machine learning.

Hui Zhu (M'13–SM'19) received the B.Sc. degree from Xidian University, Xi'an, Shaanxi, China, in 2003, the M.Sc. degree from Wuhan University, Wuhan, Hubei, China, in 2005, and the Ph.D. degree from from Xidian University, Xi'an, Shaanxi, China, in 2009. He was a Research Fellow with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore, in 2013.

Since 2016, he has been a Professor with the School of Cyber Engineering, Xidian University. His current research interests include applied cryptography, data security, and privacy.

Fengwei Wang received his B.Sc. degree from Xidian University in 2016, and Ph.D. degree from Xidian University in 2021. In 2019, he was a visiting scholar with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Since 2021, he has been the lecturer in the School of Cyber Engineering, Xidian University, China.

His research interests include the areas of applied cryptography, big data security, and privacy protection.

Rongxing Lu (S'09–M'11–SM'15–F'21) is an Associate Professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. His research interests include applied cryptography, privacy enhancing technologies, and the IoT-big data security and privacy. He was the Winner of the 2016–2017 Excellence in Teaching Award from the FCS, UNB. He was the recipient of nine best (student) paper awards from some reputable journals and conferences. Currently, he serves as the Chair for the IEEE ComSoc Communications and Information Security Technical Committee (CIS-TC), the Founding Co-Chair for the IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC), and the Chair of Mastercard IoT Research.

Zhe Liu (SM'16) received the B.S. and M.S. degrees from Shandong University in 2008 and 2011, respectively, and the Ph.D. degree from University of Luxembourg in 2015. He is a Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. His research interests include security, privacy, and cryptography solutions for the Internet of Things.

Prof. Liu was a recipient of the prestigious FNR Awards-Outstanding Ph.D. Thesis Award in 2016, ACM CHINA SIGSAC Rising Star Award in 2017 as well as DAMO Academy Young Fellow in 2019. He served as general co-chair of CHES 2020 and CHES 2021.

Hui Li (M'10) Received his B.Sc. degree from Fudan University in 1990, M.Sc. and Ph.D. degrees from Xidian University in 1993 and 1998, respectively.

Since 2005, he has been the professor in the school of Telecommunication Engineering, Xidian University, China. His research interests are in the areas of cryptography, wireless network security, information theory and network coding.

Dr. Li served as TPC co-chair of ISPEC 2009 and IAS 2009, general co-chair of E-Forensic 2010, ProvSec 2011 and ISC 2011, honorary chair of NSS 2014, ASIACCS 2016.