Secure Decentralized Image Classification with Multiparty Homomorphic Encryption

Guowen Xu, Guanlin Li, Shangwei Guo, Tianwei Zhang, Hongwei Li

***Abstract*— Decentralized image classification plays a key role in various scenarios due to its attractive properties, including tolerating high network latency and less prone to single-point failures. Unfortunately, training such a decentralized image classification model is more vulnerable to data privacy leaks compared to other distributed training frameworks. Existing efforts exclusively use differential privacy as the cornerstone to alleviate the threat to data privacy. However, differential privacy is implemented at the expense of accuracy, which goes against our motivation for designing an image classification model without loss of accuracy. To address this problem, we propose D^2-MHE, the *first* secure and efficient decentralized training framework with lossless precision. Inspired by the latest developments in the homomorphic encryption technology, we design a multiparty version of Brakerski-Fan-Vercauteren (BFV), one of the most advanced cryptosystems, and use it to implement private gradient updates of users' local models. D^2-MHE can reduce the communication complexity of general Secure Multiparty Computation (MPC) tasks from quadratic to linear in the number of users, making it very suitable and scalable for large-scale decentralized learning systems. Moreover, D^2-MHE provides strict semantic security protection even if the majority of users are dishonest with collusion. We conduct extensive experiments on MNIST, CIFAR-10, and ImageNet to demonstrate the superiority of D^2-MHE. Experimental results show that D^2-MHE achieves up to $5.5\times$ reduction in computation overhead, and at least $12\times$ reduction in communication overhead compared to existing schemes.**

***Index Terms*—Privacy Protection, Decentralized Image Classification, Homomorphic Encryption.**

I. Introduction

Image classification with deep learning technology has been widely used in various scenarios, including face recognition [1], object detection [2], and information forensics [3], [4]. To achieve satisfactory performance for complex image classification tasks, modern deep learning models need to be trained from excessive computing resources and data samples. A conventional approach is centralized training (Figure 1(a)): each user is required to upload his training samples to a third party (e.g., the cloud server), which has enough computing resources to produce the final model. However, this fashion raises widespread privacy concerns about training data [5]. Intuitively, an untrusted third party has a financial incentive to abuse sensitive data collected from different users, such as malicious dissemination, packaging, and selling them to the black market.

To alleviate the above problem, one potential way is to split and distribute the training task to the users, who only need to train the models locally and then share the gradients without disclosing their private data. As a result, untrusted third parties do not have access to users' data, and the privacy risk of training data is effectively reduced. For example, in federated learning, a central server collects the gradients from all these users, aggregates them, and distributes the new gradient to each user (Figure 1(b)). In a decentralized learning system, the central server is eliminated, so each user autonomously exchanges gradients with its neighbors and updates its model (Figure 1(c)).

Although users do not release their training samples in these distributed systems, the adversary can still infer the attributes of these samples or even reconstruct the original samples [6], [7] from the shared gradients. This threat is more severe in decentralized learning than in federated learning [8], [9], as any user connected to an honest user can access its gradients and compromise its data privacy, making the potential attack domains and methods more diversified and concealed. Since decentralized learning exhibits higher robustness to network delays and single point of failure, it becomes more promising and suitable for distributed training with large numbers of users. Therefore, it is necessary to have an efficient and privacy-preserving decentralized deep learning framework. Unfortunately, privacy protection of decentralized learning systems is still in its infancy. Although a wealth of works have been proposed to protect the privacy of federated learning [10], [5], [11], [12], they cannot be easily extended to the decentralized learning scenario due to its unique network topology and gradient propagation mechanism.

In particular, existing privacy-preserving deep learning solutions are mainly evolved from the following technologies. (1) *Differential Privacy* [13]: this approach adds controllable noise to the users' data, gradients, or intermediate values to obfuscate the adversary's observations while maintaining the training accuracy. However, its implementation enforces the need to sacrifice a certain degree of accuracy if satisfactory security is guaranteed, which is contrary to the aim of this paper to design an image classification model without loss of accuracy. (2) *Secure Multi-Party Computation* (MPC) [14]: this approach enables multiple entities to securely compute arbitrary functions without revealing their secret inputs. It has been widely used in centralized and federated learning systems [15], [16], [17]. However, it is hard to be grafted to the decentralized scenario due to the lack of central servers,

Guowen Xu, Guanlin Li, and Tianwei Zhang are with the School of Computer Science and Engineering, Nanyang Technological University. (e-mail: guowen.xu@ntu.edu.sg; guanlin001@e.ntu.edu.sg; tianwei.zhang@ntu.edu.sg). The corresponding author is *Tianwei Zhang*.

Shangwei Guo is with the College of Computer Science, Chongqing University, Chongqing 400044, China.(e-mail: swguo@cqu.edu.cn)

Hongwei Li is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China. (e-mail: hongweili@uestc.edu.cn)

This article has been accepted for publication in IEEE Transactions on Circuits and Systems for Video Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2023.3234278

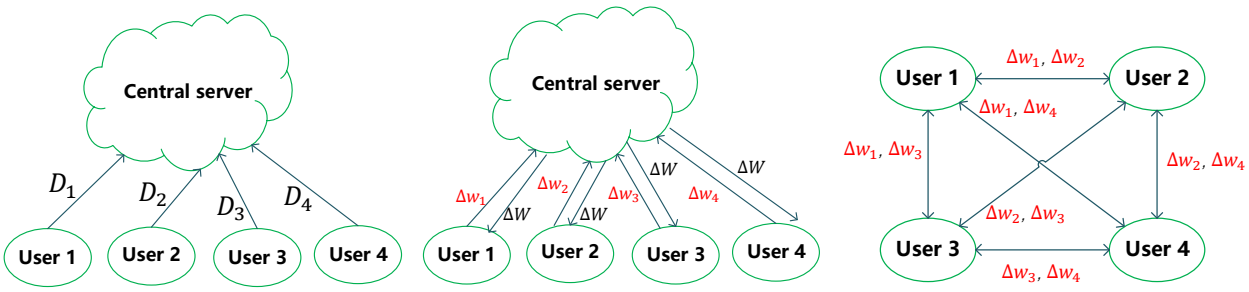


Fig. 1: Different types of training systems. (a) Centralized learning: each user i uploads its dataset D_i to a central server, which trains a specific model in a centralized manner. (b) Federated learning: each user train a local model with its own dataset. A central server is introduced to aggregate the gradients Δw_i uploaded by each user i. Then, each user updates the local model using the aggregated value (global parameter ΔW) returned by the server. (c) Decentralized learning: each user trains its own local model, and exchanges gradients with other users interconnected with it in the network. Meanwhile, it collects the gradients from the neighbors, aggregates them and updates its local model.

thus executing the secret sharing protocols across users is rather inefficient. (3) *Homomorphic Encryption* (HE) [11]: this approach enables the calculation of arbitrary (approximate) polynomial functions in a ciphertext environment without the need for decryption. It has been widely used in private deep learning [18], [19], [5]. However, it requires expensive calculations for function evaluation under ciphertext, which can significantly affect the efficiency of decentralized learning. (4) *Other Emerging Technologies*: Some emerging techniques, such as the fractal sorting method [20], [21], [22], can be explored to encrypt gradients in decentralized systems. However, in this paper, we require the underlying technology to perform ciphertext operations (including addition and multiplication) on the premise of encryption. These schemes currently have limited support for ciphertext operations, which may result in poor execution performance. More analysis of the limitations of these solutions is given in Section II.

Driven by the above limitations, our goal is to remedy the gap in the practicality of decentralized learning for protecting the training data privacy. We propose **D^2-MHE**, a practical, privacy-preserving and high-fidelity decentralized training framework. **D^2-MHE** is built based on the HE technology with innovations to address the computational bottleneck of ciphertext operation and distributed decryption. We explore the benefits of a state-of-the-art HE method, Brakerski-Fan-Vercauteren (BFV) [23], and extend it to a multiparty version for privacy-preserving decentralized learning.

Specifically, the BFV cryptosystem is a fully homomorphic encryption scheme based on the Ring Learning with Error (RLWE) problem. It supports both addition and multiplication operations in ciphertext. Compared to standard BFV, the main difference from our multiparty version is that the decryption capability is divided into N users. This means that the public key pk used for encryption is disclosed to all users, while the secret key sk is divided into N shares and can only be recovered with the collaboration of N users. As a result, to construct such a multiparty version, all algorithms that require sk as input need to be modified to meet the needs of distributed decryption. In detail, we construct four new functions ($\mathsf{MBFV \cdot SecKeyGen}$, $\mathsf{MBFV \cdot PubKeyGen}$, $\mathsf{MBFV \cdot Bootstrap}$ and $\mathsf{MBFV \cdot Convert}$) based on the standard BFV cryptosystem, to support system secret key generation, public key generation, distributed bootstrapping, and ciphertext conversion in a decentralized environment, respectively (see Section IV-B for more details). The four novel functions in **D^2-MHE** can satisfy the following properties.

First, all constructed functions are bound to the given NP-hard problem, to ensure the semantic security of the **D^2-MHE** cryptosystem. Second, the BFV cryptosystem reduces the security of the scheme to the famous NP-hard problem (i.e., Decision-RLWE [24]) by adding controllable noise in the ciphertext. Noise must be erased in the decryption process to ensure the correctness of the decryption. **D^2-MHE** follows such security guidelines, but accumulates more noise in the process of generating the public key pk (See Section IV-B). Moreover, this accumulated noise will be transferred to other operations that require pk as input. Therefore, to ensure the correctness of the decryption, we carefully control the scale of the noise added to the newly constructed functions. Third, in the standard BFV, the decryption is performed by a party holding the secret key. However, in **D^2-MHE**, this must be done without revealing sk. Obviously, once sk is revealed, all local gradients that users previously encrypted with pk will be leaked. To achieve this, we design a new method to realize ciphertext conversion [25] in BFV, i.e., converting a ciphertext originally encrypted under the public key of the system pk into a new ciphertext under the recipient's public key pk'.

To the best of our knowledge, **D^2-MHE** is the first work to accelerate the performance of decentralized learning by using cryptographic primitives. It provides the best accuracy-performance trade-off compared to existing work. Our contributions can be summarized as follows.

- We design a novel decentralized training framework **D^2-MHE** with the multiparty homomorphic encryption. Compared to existing work, it reduces the communication overhead of each round of gradient update from quadratic

to linear in the number of users without sacrificing the accuracy of the original model.
- We provide a rigorous security proof for **D^2-MHE**. Theoretical analysis shows that **D^2-MHE** can provide semantic security even if most of the users participating in the training are dishonest and collude with each other.
- We conduct extensive experiments on MNIST, CIFAR-10 and ImageNet to demonstrate the the superiority of **D^2-MHE** in performance, including the advantages of communication and computation overhead compared with existing similar schemes.

The remainder of this paper is organized as follows. Section II discusses related work on privacy-preserving solutions and limitations. In Section III, we review some basic concepts and introduce the scenarios and threat models in this paper. In Section IV, we give the details of our **D^2-MHE**. Security analysis and performance evaluation are presented in Sections V and VI, respectively. Section VII concludes the paper.

II. Related Works

We review existing privacy-preserving solutions for deep learning, which can be classified into following categories: *Differential Privacy*, *Secure Multi-Party Computation* (MPC), *Homomorphic Encryption* (HE), and *Other Emerging Technologies*. We provide a comprehensive investigation as follows.

A. Differential Privacy (DP)

Differential privacy relies mainly on the addition of controllable noise to the user's local data, gradient, or intermediate value, to realize the confusion of user data, but to ensure training performance [26], [27]. Several works [28], [8], [9], [29], [30], [31] have been designed for decentralized training scenarios. For example, *Cheng et al.*[8] propose LEASGD, which achieves a predetermined privacy budget by adding random noise to the users' local gradients and calibrates the noise scaling by analyzing the sensitivity of the update function in the algorithm. *Bellet et al.* [29] also design a completely decentralized algorithm to solve the problem of personalized optimization, and use differential privacy to protect the privacy of user data. Other works, like A$(DP)^2$SGD [31] and ADMM [9], implement the perturbation of the gradients of each user with similar tricks.

Limitations: It is still unclear whether differential neural network training can provide a satisfactory utility-privacy trade-off for common models. This stems from the inherent shortcomings of differential privacy: achieving a strong level of privacy protection requires injecting a large amount of noise during model training, which inevitably reduces the model accuracy [32], [33](See SectionVI-A).

B. Secure Multi-Party Computation (MPC)

MPC allows multiple participants to securely compute arbitrary functions without releasing their secret inputs [34], [35]. It has been widely used in conventional deep learning scenarios, including centralized learning and federated learning [15], [36], [16], [17]. Most of these efforts rely on users to secretly share (utilizing Shamir's Secret-Sharing [37] or Additive Secret-Sharing [38]) local data or gradients to two or more servers, and require an honest majority to perform deep learning training and prediction without collusion. In this way, frequent secret sharing between users is avoided, and the complexity of communication overhead is reduced from O(N^2) to O(S^2), where N and S represent the numbers of users and servers, respectively.

Limitations: It is convincing to explore MPC-based protocols in centralized or federal learning, because third-party servers naturally exist in these scenarios. However, grafting MPC to a decentralized scenario has the following limitations. (1) Decentralized learning abandons central servers to avoid a single point of failure and communication bottlenecks. As a result, it is conflicting to transplant the existing MPC-based training mechanism to a decentralized mode. A trivial idea to alleviate this problem is to execute the secret sharing protocol between users directly, which is rather inefficient since each user needs to perform $N-1$ interactions for secret sharing at each iteration [39] (refer to Section VI-C). (2) The existing MPC technology generally requires that most of the entities involved in the calculation are honest and will not collude with each other [15], [36], [16], [17]. This is done to ensure smooth execution of calculations. In other words, if the majority of entities are dishonest and collude with each other, there is a high probability that execution will be terminated or errors will occur. However, a strong security framework should be able to withstand attacks from adversarial collusion. In a decentralized scenario, the need for such a security guarantee is more urgent because any user can obtain the gradient of other users connected to it and then easily collude with some malicious users to break the privacy of the target user.

C. Homomorphic Encryption (HE)

(Fully) homomorphic encryption can achieve the calculation of arbitrary (approximate) polynomial functions in ciphertext without the need for decryption [40], [41]. Such an attractive nature makes it widely used in private deep learning [18], [19], [5], [11]. Informally, we can divide HE into the following two types with different decryption methods: (1) standard HE [42], [43] is used mainly for model inference, where the public key is released to all participants, while the secret key is only held by the decryptor (e.g., the user). (2) In threshold-based HE [44], [45], [46], the secret key is securely shared with multiple entities. As a result, each entity still performs a function evaluation under the same public key, while decryption of the result requires the participation of the number of entities exceeding the threshold. Several threshold-based HE variants [46], [44], [47] have been used in the federated learning scenario, and one of the most representative is the threshold Paillier-HE [46]. For example, *Zheng et al.* [11] propose Helen, the first secure federated training system utilizing the threshold Paillier-HE. In Helen, each user's data are encrypted with Paillier-HE and submitted to an "Aggregator", which is responsible for performing aggregation. Then, the Aggregator broadcasts the aggregated results to all the users to update the local model parameters. When the trained model reaches

the preset convergence condition, the model parameters can be decrypted by collaboration of multiple users without revealing the original private key.

Limitations: Threshold Paillier-HE requires substantial modular exponential operations for the evaluation of functions in ciphertext and requires expensive calculations among multiple users for decryption. In a decentralized scenario, each user receives gradients from neighboring users and aggregates them to update its local model. This inevitably produces worse performance if threshold Paillier-HE is simply used as its underlying architecture (please refer to Section VI for more details). Other variants like TFHE [44], are possibly applicable to the decentralized environment. However, TFHE can only encrypt one bit at a time, which is obviously unrealistic to achieve practical training.

D. Other Emerging Technologies

Some emerging techniques, such as fractal sorting method [20], [21], [22], have recently been proposed for encrypting images and have demonstrated good performance. For example, *Xian et al.* proposed fractal sorting matrix (FSM) [20], a new type of sorting matrix with fractal characteristics. Then they propose a new method of global pixel diffusion with two chaotic sequences, which offers good security and high encryption efficiency. *Xian et al.* also presented the double parameters fractal sorting matrix (DPFSM) [21], which contains self-similar structures in the ordering of both elements and sub-blocks in the matrix. The image encryption algorithm based on DPFSM is demonstrated to have semantic security and excellent encryption performance. However, since this paper we aim to perform efficient decentralized gradient aggregation on ciphertexts, which requires the underlying cryptography to enable mathematical operations between ciphertexts, a property that fractal sorting method does not possess. Therefore, we do not consider how to use these emerging techniques to perform gradient updates in this paper, leaving the exploration to the future.

Remark 1: Based on the above discussions, we argue that differential privacy-based and MPC-based approaches are contrary to our motivation and the characteristics of decentralized learning systems. In contrast, threshold-based HE seems to be more promising, if it can be freed from the computational bottleneck of ciphertext operation and distributed decryption. Inspired by this, this paper focuses on exploring the benefits of a state-of-the-art HE method, Brakerski-Fan-Vercauteren (BFV), and the possibilities to extend it to a multiparty version for privacy-preserving decentralized learning.

III. Preliminaries

In this section, we first review some basic concepts about decentralized parallel stochastic algorithms and BFV homomorphic encryption. Then, we describe the threat model and privacy requirements considered in this paper.

A. Decentralized Parallel Stochastic Algorithms

As shown in Figure 1(c), a decentralized system can be represented as an undirected graph (V, E), where V denotes a set of N nodes in the graph (i.e., users in the system[1]), and E denotes a set of edges representing communication links. We have $(i, j) \in E$ if and only if node i can receive information from node j. $\mathcal{N}_i = \{j|(i, j) \in E\}$ represents the set of all nodes connected to node i. $E \in \mathbb{R}^{N\times N}$ is a doubly symmetric stochastic matrix to denote the training dependency of two nodes. It has the following two properties: (i) $E_{i,j} \in [0, 1]$ and (ii) $\sum_j E_{i,j} = 1$ for all i. Commonly for a node i, we can set $E_{i,j} = 0$ if node $j \notin \mathcal{N}_i$ and $E_{i,j} = 1/|\mathcal{N}_i|$ otherwise.

For neural network training in such a decentralized system, all nodes are required to optimize the following function [48], [49]:

$$\min_{W\in\mathbb{R}^H} G(W) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{X\sim D_i} L_i(W, X) \tag{1}$$

where $W \in \mathbb{R}^H$ denotes the parameters of the target model. The distribution of training samples for each user is denoted as D_i, and $X \in \mathbb{R}^M$ represents a training sample from the distribution. For each node i, $L_i(W, X) = L(W, X)$ denotes the loss function.

Decentralized Parallel Stochastic Gradient Decay algorithm (D-PSGD) [49] is usually used to solve the above optimization problem. Its main idea is to update the local model by requiring each user to exchange gradients with their neighbors (technical details are shown in **Algorithm 1**). In this paper, our goal is to implement D-PSGD in a privacy-preserving way.

Algorithm 1 Decentralized Parallel Stochastic Gradient Decay

Input: Initialize $W_{0,i} = W_0$, E, step length η, and K.

1: **for** $k = 0, 1, 2, \cdots K-1$ **do**

2: Each node i select sample $X_{k,i}$ and calculate $\Delta W_i = \triangledown L_i(W_{k,i}, X_{k,i})$.

3: Receive $W_{k,j}$ of all nodes in $\mathcal{N}_i$, node i calculates

$$W_{k+\frac{1}{2},i} = E_{i,i}W_{k,i} + \sum_{j\in\mathcal{N}_i} E_{i,j}W_{k,j}$$

4: Node i computes $W_{k+1,i} \leftarrow W_{k+\frac{1}{2},i} - \eta\Delta W_i$

5: Node i broadcast $W_{k+1,i}$ to all nodes in $\mathcal{N}_i$.

6: **end for**

Output: $\frac{1}{N}\sum_{i=1}^{N} W_{K,i}$.

B. BFV Homomorphic Encryption

The BFV cryptosystem [23] is a fully homomorphic encryption scheme based on the Ring-learning with error (RLWE) problem. It supports both addition and multiplication operations in ciphertext. In this section, we briefly introduce the basic principles of the standard BFV algorithm used in the centralized scenario. In Section IV, we will explain in detail how to convert this standard BFV to the multiparty version and enable decentralized training.

Suppose the ciphertext space is composed of polynomial ring $R_q = \mathbb{Z}_q[\mathtt{X}]/(\mathtt{X}^n+1)$, and the quotient ring of polynomials with coefficients in $\mathbb{Z}_q$, where $\mathtt{X}^n + 1$ is a monic irreducible polynomial with degree of $n = 2^b$. The set of integers in $(\frac{-q}{2}, \frac{q}{2}]$ is used to denote the representatives of the congruence

[1]In this paper we use the terminologies of node and user interchangeably.

classes modulo q. Similarly, the plaintext space is denoted as the ring $R_t = \mathbb{Z}_t[\mathtt{X}]/(\mathtt{X}^n+1)$ where $t < q$. We use $\Lambda = \lfloor q/t \rfloor$ to represent the integer division of q by t. Unless otherwise stated, we consider the arithmetic of R_q. Therefore, the symbol of polynomial reductions is sometimes omitted from the BFV execution. Informally, the standard BFV encryption system consists of the following five algorithms.

1. $\mathsf{BFV} \cdot \mathsf{SecKeyGen}(1^\lambda) \rightarrow sk$: Given the security parameter λ, this algorithm selects an element s uniformly on the polynomial ring $R_3 = \mathbb{Z}_3[\mathtt{X}]/(\mathtt{X}^n+1)$, where the coefficients of every polynomial in R_3 are uniformly distributed in $\{-1, 0, 1\}$. Then, it outputs the secret key $sk = s$.

2. $\mathsf{BFV} \cdot \mathsf{PubKeyGen}(sk) \rightarrow pk$: Given the secret key s, this algorithm selects an element p_1 uniformly on the polynomial ring R_q and an error term e from χ. χ is a distribution over R_q with coefficients obeying the centered discrete Gaussian with standard deviation σ and truncated to support over $[-B, B]$. Then it outputs $pk = (p_0, p_1) = (-(sp_1 + e), p_1)$. According to the current version of the homomorphic encryption standard [50], (σ, B) are set as $(\frac{8}{\sqrt{2\pi}} \approx 3.2, \lfloor 6 \cdot \sigma \rceil = 19)$. As a result, the selected noise is not only random, but its norm is controlled between $[-19, 19]$ (please refer to [23], [50] for more details of the parameter settings).

3. $\mathsf{BFV} \cdot \mathsf{Encrypt}(pk, x) \rightarrow ct$: Given the public key $pk = (p_0, p_1)$, this algorithm samples an element μ uniformly from R_3 and two error terms e_0, e_1 from χ. Then, it outputs the ciphertext $ct = (\Lambda x + \mu p_0 + e_0, \mu p_1 + e_1)$.

4. $\mathsf{BFV} \cdot \mathsf{Eval}$ $(pk, f, ct_1, ct_2, \cdots, ct_N) \rightarrow ct'$: Given the public key pk, the function f to be evaluated, and $N (N \geq 1)$ ciphertext inputs $(ct_1, ct_2, \cdots, ct_N)$, this algorithm outputs the ciphertext result ct'. Note that since BFV supports addition and multiplication operations in ciphertext, it is feasible to securely evaluate a function f that can be (approximately) parsed as a polynomial. To achieve this, BFV uses $\mathsf{BFV} \cdot \mathsf{Add}$ and $\mathsf{BFV} \cdot \mathsf{Mul}$ operations to perform homomorphic addition and multiplication, respectively, and uses *relinearization key (rlk)* to ensure consistency of the ciphertext form after each multiplication. In addition, $\mathsf{BFV} \cdot \mathsf{Bootstrap}$ is used to reduce the noise of the ciphertext back to a fresh-like one, which enables further calculations even if the noise of the current ciphertext reaches the limit of the homomorphic capacity. The reader can refer to [23] for more details.

5. $\mathsf{BFV} \cdot \mathsf{Decrypt}(sk, ct) \rightarrow x$: Given the secret key s and the ciphertext $ct = (c_0, c_1)$, this algorithm outputs the decrypted plaintext $x = \left[\left\lfloor \frac{t}{q}[c_0 + c_1 s]_q \right\rceil\right]_t$, where $[c_0 + c_1 s]_q$ denotes $c_0 + c_1 s \mod q$.

The security of the BFV cryptosystem is reduced to the famous *Decisional-RLWE Problem* [24]. Informally, given a random a, a secret key s and an error term e uniformly sampled from R_q, R_3 and χ, respectively, it is computationally difficult for an adversary to distinguish the two distributions $(sa+e, a)$ and (g, a) without the knowledge of s and e, where g is uniformly sampled from R_q.

C. Threat Model and Privacy Requirement

As shown in Figure 1(c), we consider a decentralized learning system with N users. Each user i with a local dataset D_i adopts the D-PSGD algorithm to collaboratively train a deep learning model with others. In this paper, each user is considered to be honest but curious, i.e., they follow the agreed procedure to perform the training task, but may try to obtain the private data (i.e., gradients) of other users along with the collected prior knowledge. As a result, attacks from malicious adversaries by violating the execution of the protocol are beyond the scope of this paper. Such a threat model has been widely used in existing works about privacy-preserving machine learning [18], [35], [5], [39]. Moreover, we allow the collusion of the majority of users to enhance the attack capabilities. Specifically, for the union composed of user i and its set of connected nodes $\mathcal{N}_i$, i.e., $\mathcal{U} = i \cup \mathcal{N}_i$, collusion of at most $|\mathcal{U} - 1| = |\mathcal{N}_i|$ users is allowed at any training stage, where $|\mathcal{N}_i|$ denotes the number of users in the set $\mathcal{N}_i$. Our goal is to protect the confidentiality of sensitive data (i.e., gradients) for each benign user. This means that during the training process, we should guarantee that no user i can learn the gradient ΔW_j of any benign user j, except those that can be inferred from its own input data ΔW_i.

IV. PROPOSED SCHEME

We present a novel privacy-preserving framework, Decentralized Deep learning with Multiparty Homomorphic Encryption (**D^2-MHE**), which enables N users to train the target model collaboratively under the decentralized network. We first give the overview of **D^2-MHE** for implementing the D-PSGD algorithm with the multiparty version of BFV, and then further explain the detailed algorithms.

A. Overview

Essentially, in **D^2-MHE**, users iteratively execute the D-PSGD algorithm with the Multiparty BFV (MBFV) cryptosystem. The complete algorithm is shown in **Algorithm 2**, where the newly constructed functions (marked in red) are introduced to convert the standard BFV into a multiparty version. During decentralized training, each node requires additional operations to securely execute the D-PSGD compared to the original algorithm, including generating additional variables for encryption/decryption, and modifying certain operations to implement the ciphertext calculations. We describe each step in **Algorithm 2** as well as Figure 2.

(1) In the initialization phase, each node i generates its own secret key sk_i and public key pk_i using the standard BFV. Then, for each set $\mathcal{N}_i, i \in [1, N]$, a new function $\mathsf{MBFV} \cdot \mathsf{SecKeyGen}$ is used to generate shares $(s_i, s_j | j \in \mathcal{N}_i)$ of the system secret key $sk = s$, where $s = s_i + \sum_{j \in \mathcal{N}_i} s_j$. Then, a new function $\mathsf{MBFV} \cdot \mathsf{PubKeyGen}$ is exploited to generate the public key pk corresponding to s. Note that for two different $\mathcal{N}_i$ and $\mathcal{N}_j$, we need to repeat the above key generation process to ensure that distinct key pairs are produced for each group.

(2) For each node j, instead of sending the original gradient $W_{k,j}$ to other neighboring nodes, it uses the standard $\mathsf{BFV} \cdot \mathsf{Encrypt}$ to send the ciphertext $\mathsf{E}(W_{k,j})$ (line 6 and Figure 2(a)) to all the connected users $i \in \mathcal{N}_j$.

This article has been accepted for publication in IEEE Transactions on Circuits and Systems for Video Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2023.3234278

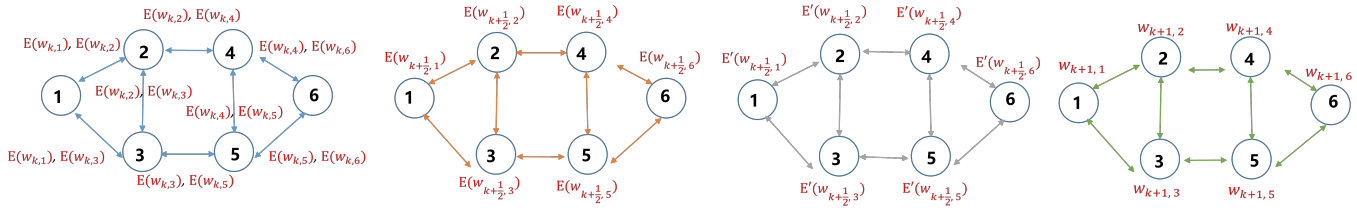


Fig. 2: A high-level view of our **D^2-MHE**. (a) Step 1: Exchange encrypted local parameters with connected nodes. (b) Step 2: Each node locally aggregates parameters from other nodes. (c) Step 3: Each node interacts with connected nodes to convert the local aggregated value into ciphertext under its own public key. (d) Step 4: Each node decrypts the aggregated value and updates the local parameters.

Algorithm 2 Privacy-preserving D-PSGD

Input: Each node i generates $sk_i \leftarrow \mathsf{BFV \cdot SecKeyGen}(1^\lambda)$ and $pk_i \leftarrow \mathsf{BFV \cdot PubKeyGen}(sk_i)$. For each set $\mathcal{N}_i, i \in [1, N]$, nodes i and $j \in \mathcal{N}_i$ generate $(s_i, s_j | j \in \mathcal{N}_i) \leftarrow \mathsf{MBFV \cdot SecKeyGen}(1^\lambda)$ where $s = s_i + \sum_{j \in \mathcal{N}_i} s_j$. Initialize $W_{0,i} = W_0$, E, η, and K.

1: Each node i cooperates with the nodes in $\mathcal{N}_i$ to generate $pk \leftarrow \mathsf{MBFV \cdot PubKeyGen}(s_i, s_j | j \in \mathcal{N}_i)$
2: Each node i broadcasts pk_i to all nodes $j \in \mathcal{N}_i$.
3: **for** $k = 0, 1, 2, \cdots K-1$ **do**
4: Each node i performs the following operations in parallel:
5: Randomly select sample $X_{k,i}$ and calculate $\Delta W_i = \nabla L_i(W_{k,i}, X_{k,i})$
6: Obtain $\mathsf{E}(W_{k,j}) \leftarrow \mathsf{BFV \cdot Encrypt}(pk, W_{k,j})$ from all $j \in \mathcal{N}_i$, and calculate $\mathsf{E}(W_{k+\frac{1}{2},i})$ with the algorithm $\mathsf{MBFV \cdot Bootstrap}(ct, s_i, s_j | j \in \mathcal{N}_i)$ as follows:

$$\mathsf{E}(W_{k+\frac{1}{2},i}) \leftarrow \mathsf{BFV \cdot Eval}(pk, f, \mathsf{E}(W_{k,i}), \mathsf{E}(W_{k,j}) | j \in \mathcal{N}_i)$$

where $f = E_{i,i} W_{k,i} + \sum_{j \in \mathcal{N}_i} E_{i,j} W_{k,j}$.
7: Broadcast $\mathsf{E}(W_{k+\frac{1}{2},i})$ to all nodes $j \in \mathcal{N}_i$.
8: Convert $\mathsf{E}(W_{k+\frac{1}{2},i})$ into a new ciphertext $\mathsf{E}'(W_{k+\frac{1}{2},i}) \leftarrow \mathsf{MBFV \cdot Convert}(\mathsf{E}(W_{k+\frac{1}{2},i}), pk_i, s_i, s_j | j \in \mathcal{N}_i)$ with pk_i.
9: Decrypt $\mathsf{E}'(W_{k+\frac{1}{2},i})$ as $W_{k+\frac{1}{2},i} \leftarrow \mathsf{BFV \cdot Decrypt}(sk_i, \mathsf{E}'(W_{k+\frac{1}{2},i}))$.
10: Update $W_{k+1,i} \leftarrow W_{k+\frac{1}{2},i} - \eta \Delta W_i$.
11: Broadcast $\mathsf{E}(W_{k+1,i}) \leftarrow \mathsf{BFV \cdot Encrypt}(pk, W_{k+1,i})$ to all nodes in $\mathcal{N}_i$.
12: **end for**

Output: $\frac{1}{N} \sum_{i=1}^{N} W_{K,i}$.

(3) Each node i calculates the encrypted weighted average $\mathsf{E}(W_{k+\frac{1}{2},i})$ with the help of a new function $\mathsf{MBFV \cdot Bootstrap}$ (Figure 2(b)), which is the multiparty bootstrapping procedure. $\mathsf{MBFV \cdot Bootstrap}$ can reduce the noise of a ciphertext (such as the intermediate value ct in line 6) back to a fresh-like one, which enables further calculations even if the noise of the current ciphertext reaches the limit of homomorphic capacity.

(4) To securely decrypt $\mathsf{E}(W_{k+\frac{1}{2},i})$, we construct a new function $\mathsf{MBFV \cdot Convert}$, which can obliviously re-encrypt $\mathsf{E}(W_{k+\frac{1}{2},i})$ that is originally encrypted under the system public key pk, into a new ciphertext $\mathsf{E}'(W_{k+\frac{1}{2},i})$ under the recipient's public key pk_i (line 8 and Figure 2(c)).

(5) As a result, each node i can decrypt $\mathsf{E}'(W_{k+\frac{1}{2},i})$ with its secret key sk_i, and then update its local model parameters (lines 9-10 and Figure2(d)).

To sum up, in **D^2-MHE**, we construct four new functions ($\mathsf{MBFV \cdot SecKeyGen}$, $\mathsf{MBFV \cdot PubKeyGen}$, $\mathsf{MBFV \cdot Bootstrap}$, and $\mathsf{MBFV \cdot Convert}$) based on the standard BFV cryptosystem, which are used to support system secret key generation, public key construction, distributed bootstrapping procedure, and ciphertext conversion in a decentralized learning environment. Note that the functions $\mathsf{MBFV \cdot SecKeyGen}$ and $\mathsf{MBFV \cdot PubKeyGen}$ are only executed once during the entire training process. Besides, all nodes encrypt their local gradients under the same public key pk and broadcast to other nodes. As a result, compared with existing MPC-based works, where each node needs to secretly share its gradients to all neighbors, our method only requires each user to broadcast a ciphertext to all users. Therefore, from the perspective of the whole system, **D^2-MHE** reduces the communication overhead of each round of gradient update from quadratic to linear without sacrificing the accuracy of the original model.

Remark 2: Our **D^2-MHE** is inspired by work [38], which proposes a cryptographic primitive called multiparty homomorphic encryption from ring-learning-with-errors. However, there are three major differences between [38] and our work, which makes [38] incompatible with our scenario. (1) [38] focuses on the construction of relinearization keys for multiparty HE, thereby ensuring the correctness of multiplication between ciphertexts. It is hard to apply it to our system which mainly consists of ciphertext aggregation operations rather than multiplications. (2) [38] considers to convert a ciphertext originally encrypted under the system secret key into a new ciphertext under the recipient's secret key which is securely shared with other users. In our system, we need access to the recipient's

public key instead of the secret key. (3) [38] is mainly designed for scenarios such as private information retrieval and private set intersection, while our scheme is tailored to decentralized learning. Due to the above differences, compared to [38], our proposed algorithms are more concise and efficient and are more suitable for decentralized deep learning.

B. Detailed Implementations of the Four Functions

We provide the details of the above four newly constructed functions in **D^2-MHE**. For readability, we take the variables in **Algorithm 2** as inputs/outputs of the functions.

1. $\mathsf{MBFV}\cdot\mathsf{SecKeyGen}(1^\lambda)\rightarrow(s_i,s_j|j\in\mathcal{N}_i)$: Given a security parameter λ, this function generates shares $(s_i,s_j|j\in\mathcal{N}_i)$ of the system secret key $sk=s$ for each set $\mathcal{N}_i, i\in[1,N]$. In this paper, we focus on additive secret sharing [38] of the key, i.e., $s=s_i+\sum_{j\in\mathcal{N}_i}s_j$, which can also be replaced by Shamir's threshold secret sharing [37] with less strict requirements. We propose a simple method to implement $\mathsf{MBFV}\cdot\mathsf{SecKeyGen}(1^\lambda)$, i.e., each user independently generates s_i through the standard $\mathsf{BFV}\cdot\mathsf{SecKeyGen}(1^\lambda)$. Then, we can simply set $s=s_i+\sum_{j\in\mathcal{N}_i}s_j$. This means that s is not set beforehand but is determined when all nodes have generated their own shares. The advantage of generating s in this way is that each node i does not need to share its s_i with others. It is a common way to generate a collected key and has been proven to be secure [46], [51][2]. Note that (pk,sk) is bound to each $\mathcal{N}_i=\{j|(i,j)\in E\}$, rather than generated for all nodes in the system. Given two different $\mathcal{N}_i$ and $\mathcal{N}_j$, the key pairs generated for these two groups do not need to be the same, since this will not affect the subsequent homomorphic calculation and decryption.

2. $\mathsf{MBFV}\cdot\mathsf{PubKeyGen}(s_i,s_j|j\in\mathcal{N}_i)\rightarrow pk$: This process is to emulate the standard $\mathsf{BFV}\cdot\mathsf{PubKeyGen}$ procedure, i.e., generating the public key $pk=(p_0,p_1)$ corresponding to s. To achieve this, all users should agree on a public polynomial p_1, which is uniformly sampled in the distribution R_q. Then, each node i and all nodes $j\in\mathcal{N}_i$ independently sample e_i,e_j over the distribution χ, and compute $p_{0,i}=-(p_1s_i+e_i)$, $p_{0,j}=-(p_1s_j+e_j)$, $j\in\mathcal{N}_i$. Next, each node $k\in i\cup\mathcal{N}_i$ broadcasts $p_{0,i}$ to other nodes. Therefore, each node $k\in i\cup\mathcal{N}_i$ can construct the public key of the system by performing the following operations:

$$pk=([\sum_{k\in i\cup\mathcal{N}_i}p_{0,k}]_q,p_1)=([-(p_1\sum_{k\in i\cup\mathcal{N}_i}s_k+\sum_{k\in i\cup\mathcal{N}_i}e_k)]_q,p_1)$$

We observe that pk generated with this way has the same form as the public key generated by the standard $\mathsf{BFV}\cdot\mathsf{PubKeyGen}$, but with a larger norm of $||s||$ and $||e||$, where $||s||$ and $||e||$ denote the 2-Norm for vectors s and e, respectively. The growth of norms is linear with $|\mathcal{N}_i|$, therefore, it is not a concern (proved in [44], [47]), even for a large number of $|\mathcal{N}_i|$ (See discussion below).

[2]The s generated in this way may not conform to the property of being uniformly distributed under R_3. However, this is not a problem because our security proof (refer to Section V) does not rely on this property. Furthermore, there are many other ways [11], [52] to generate uniform keys that are subject to the distribution R_3, which require private channels between users.

3.$\mathsf{MBFV}\cdot\mathsf{Convert}(\mathsf{E}(W_{k+\frac{1}{2},i}),pk_i,s_i,s_j|j\in\mathcal{N}_i)\rightarrow\mathsf{E}'(W_{k+\frac{1}{2},i})$: This function is used to convert $\mathsf{E}(W_{k+\frac{1}{2},i})$ into a new ciphertext $\mathsf{E}'(W_{k+\frac{1}{2},i})$ under the public key pk_i. As a result, node i can decrypt it with its secret key without accessing the system secret key s. To achieve this, given node i's public key $pk_i=(p_{0,i},p_{1,i})$, and $\mathsf{E}(W_{k+\frac{1}{2},i})=(c_0,c_1)$, each node $k\in i\cup\mathcal{N}_i$ samples μ_k, $e_{0,k}$, $e_{1,k}$ over the distribution χ and executes the following operations:

$$(h_{0,k},h_{1,k})=(s_kc_1+\mu_kp_{0,i}+e_{0,k},\mu_kp_{1,i}+e_{1,k}) \tag{2}$$

Then, each $(h_{0,k},h_{1,k})$ is submitted to node i. Subsequently, node i first computes $h_0=\sum_{k\in i\cup\mathcal{N}_i}h_{0,k}$, and $h_1=\sum_{k\in i\cup\mathcal{N}_i}h_{1,k}$, and then generates the new ciphertext $\mathsf{E}'(W_{k+\frac{1}{2},i})=(c_0',c_1')$=$(c_0+h_0,h_1)$.

The correctness of $\mathsf{MBFV}\cdot\mathsf{Convert}$ is shown as follows: Given node i's public key $pk_i=(p_{0,i},p_{1,i})$, and $\mathsf{E}(W_{k+\frac{1}{2},i})=(c_0,c_1)$, where $c_0+sc_1=\Delta m+e_{ct}$, $p_{0,i}=-(sk_ip_{1,i}+e_{cl})$, we have

$$\begin{aligned}
&\mathsf{BFV}\cdot\mathsf{Decrypt}(sk_i,\mathsf{E}'(W_{k+\frac{1}{2},i}))\\
&=\lfloor\frac{t}{q}[c_0+\sum_{k\in i\cup\mathcal{N}_i}(s_kc_1+\mu_kp_{0,i}+e_{0,k})\\
&+sk_i\sum_{k\in i\cup\mathcal{N}_i}(\mu_kp_{1,i}+e_{1,k})]_q\rceil\\
&=\lfloor\frac{t}{q}[c_0+sc_1+\mu p_{0,i}+sk_i\mu p_{1,i}+e_0+sk_ie_1]_q\rceil\\
&=\lfloor\frac{t}{q}[\Lambda W_{k+\frac{1}{2}}+e_{ct}+e_{Covt}]_q\rceil\\
&=W_{k+\frac{1}{2}}
\end{aligned} \tag{3}$$

where $e_d=\sum_{k\in i\cup\mathcal{N}_i}e_{d,k}$ for $d=0,1$. $\mu=\sum_{k\in i\cup\mathcal{N}_i}\mu_k$. Therefore, the additional noise involved in $\mathsf{MBFV}\cdot\mathsf{Convert}$ is $e_{Covt}=e_0+sk_ie_1+\mu e_{cl}$, which must satisfy the condition of $||e_{ct}+e_{Covt}||<q/(2t)$ for the correctness of the decryption. Note that e_{ct} and e_{cl} are the noises introduced in the process of encryption and key generation respectively, their sizes are small because the norm of the initialized noise is controlled between $[-B,B]$. $\mu=\sum_{k\in i\cup\mathcal{N}_i}\mu_k$, where each μ_k is uniformly selected from polynomial ring $R_3=\mathbb{Z}_3[\mathrm{X}]/(\mathrm{X}^n+1)$, i.e., coefficients of every polynomial in R_3 are uniformly distributed in $\{-1,0,1\}$. Therefore, the norm of $||\mu e_{cl}||$ is much smaller than $q/(2t)$, where q is much larger than t. Furthermore, since each sk_i is selected from the polynomial ring R_3, we have $e_0+sk_ie_1$ smaller than $q/(2t)$. Based on this, it is easy to ensure that $||e_{ct}+e_{Covt}||$ is smaller than $q/(2t)$.

4. $\mathsf{MBFV}\cdot\mathsf{Bootstrap}(ct,s_i,s_j|j\in\mathcal{N}_i)\rightarrow ct'$: This is the multiparty bootstrapping procedure. It can reduce the noise of a ciphertext ct to a fresh-like one ct', and then it allows for further calculations if the noise of the current ciphertext reaches the limit of homomorphic capacity. Specifically, given a ciphertext $ct=(c_0,c_1)$ with noise variance σ_{ct}^2, a common random polynomial α, each node $k\in i\cup\mathcal{N}_i$ samples M_k over R_t, $e_{0,k}$, $e_{1,k}$ over χ, and executes the following operations:

$$(\eta_{0,k},\eta_{1,k})=(s_kc_1-\Lambda M_k+e_{0,k},-s_k\alpha+\Lambda M_k+e_{1,k}) \tag{4}$$

Then, each $(\eta_{0,k}, \eta_{1,k})$ is submitted to node i. Afterwards, node i first computes $\eta_0 = \sum_{k\in i\cup\mathcal{N}_i} \eta_{0,k}$, and $\eta_1 = \sum_{k\in i\cup\mathcal{N}_i} \eta_{1,k}$, and then generates the new ciphertext $ct'=(\left[\left\lfloor \frac{t}{q}[c_0+\eta_0]_q \right\rceil\right]_t \Lambda + \eta_1, \alpha)$ with noise variance $N\sigma^2$.

C. Discussions

Noise analysis. We observe that the noise incurred is $\sum_{k\in i\cup\mathcal{N}_i} e_k$ in $\mathsf{MBFV}\cdot\mathsf{PubKeyGen}$, and $e_{Covt} = e_0 + sk_i e_1 + \mu e_{cl}$ in $\mathsf{MBFV}\cdot\mathsf{Convert}$. All noise is controllable since we can preset the range of e, sk_i, and μ (please refer to the literature [23] for a more theoretical analysis). This comes from our carefully constructed noise mechanism. Since the ciphertext is large in size and is difficult to remove during the decryption process, our criterion is to keep the accumulated noise items without ciphertext.

The utility of $\mathsf{MBFV}\cdot\mathsf{Bootstrap}$. The implementation of $\mathsf{MBFV}\cdot\mathsf{Bootstrap}$ requires interaction between multiple nodes, which will increase the communication overhead of each user. However, $\mathsf{MBFV}\cdot\mathsf{Bootstrap}$ is rarely used in our scenarios. In detail, according to the standard BFV, a ciphertext is correctly decrypted if the noise contained in the ciphertext satisfies $||e_{ct}|| < q/2t$, where q and t denote the spaces of ciphertext and plaintext, respectively. In comparison, the noise of a ciphertext involved in **D^2-MHE** is $e_{ct} + e_{Covt} \approx M \times e_{ct}$, where M can be roughly parsed as a linear function of the average number of adjacent nodes of each node in the system. The noise scale will be further increased to $M \times T \times e_{ct}$, if T times of homomorphic addition operations are performed without utilizing bootstrapping. Since the size of $||e_{ct}||$ is usually smaller than 1, for the correctness of the decryption, it is enough to ensure that $M \times T < q/2t$. Hence, given a 64-bit plaintext space and a 512-bit ciphertext space, we only need to guarantee $M \times T < \frac{2^{512}}{2^{64}} = 2^{448}$. Therefore, assuming $M = 1024$, **D^2-MHE** can still perform 2^{438} consecutive homomorphic additions without the assistance of bootstrapping.

In summary, $\mathsf{MBFV}\cdot\mathsf{Bootstrap}$ provides a trade-off between computation overhead and communication overhead. It is very practical to calculate a function without knowing the complexity of the computation complexity in advance.

V. Security Analysis

We now discuss the security of **D^2-MHE**. Compared with the standard BFV, **D^2-MHE** constructs four new functions: $\mathsf{MBFV}\cdot\mathsf{SecKeyGen}$, $\mathsf{MBFV}\cdot\mathsf{PubKeyGen}$, $\mathsf{MBFV}\cdot\mathsf{Bootstrap}$, and $\mathsf{MBFV}\cdot\mathsf{Convert}$. Since the implementation of $\mathsf{MBFV}\cdot\mathsf{SecKeyGen}$ is essentially calling the standard $\mathsf{BFV}\cdot\mathsf{SecKeyGen}$ multiple times, it inherits the security of the original algorithm. Therefore, this section focuses on the security of the other three functions. In addition, in **D^2-MHE**, each user i interacts with the connected nodes $\mathcal{N}_i$, while being separated from other users in the system. Therefore, we take the set $\mathcal{U} = i\cup\mathcal{N}_i$ as the object of discussion. In brief, the security of **D^2-MHE** is mainly tied to the *Decisional-RLWE Problem* [24] and the property of Additive Secret-Sharing [38]. Here, we provide arguments in a real/ideal simulation formalism [53].

Before explaining the details of the proof, we define some variables which are useful for subsequent descriptions. Specifically, suppose that the security parameter of **D^2-MHE** is λ, the adversary set is $\mathcal{A} \subseteq \mathcal{U}$, and $|\mathcal{A}| \leq |\mathcal{U}| - 1$. $\mathsf{REAL}_{\mathcal{U}}^{\mathcal{U},\lambda}$ is a random variable used to refer to the joint view of all users in $x_{\mathcal{U}}$, which contains all users' input in **D^2-MHE** and information received from other users. Since there is at least one honest user in the set $\mathcal{U}$, we define this honest user as g_h for convenience of description. The set $\mathcal{H} = \mathcal{U}\backslash(\mathcal{A}\cup g_h)$ represents other honest users. With these symbols, the sketch of our proof is that for any adversary set $\mathcal{A}$, when only the input and output of $\mathcal{A}$ are provided, there exists a simulator SIM with Probabilistic Polynomial Time (PPT) computation ability, which can simulate the view of $\mathcal{A}$, and make $\mathcal{A}$ unable to distinguish the real view from the simulated ones.

A. Analysis of $\mathsf{MBFV}\cdot\mathsf{PubKeyGen}$

We consider an adversary set $\mathcal{A}$ to attack $\mathsf{MBFV}\cdot\mathsf{PubKeyGen}$ defined in Section IV-B. For each user $k \in \mathcal{U}$, its private inputs are s_k and e_k, and the output received from the function is the public key of the system pk. Therefore, given $\mathcal{A}$'s inputs $\{s_k, e_k\}, k \in \mathcal{A}$ and $pk = (p_0, p_1)$, the simulator needs to construct a simulated view which is indistinguishable from the adversary's view under the implementation of the real protocol.

Theorem 1. *Given the security parameter λ, user set $\mathcal{U}$, adversary set $\mathcal{A} \subseteq \mathcal{U}$,$|\mathcal{A}| \leq |\mathcal{U}| - 1$, $\mathcal{A}$'s inputs $\{s_k, e_k\}_{k\in\mathcal{A}}$, $pk = (p_0, p_1)$, honest user g_h, and $\mathcal{H} = \mathcal{U}\backslash(\mathcal{A}\cup g_h)$, there exists a PPT simulator* SIM*, whose output is indistinguishable from the real* $\mathsf{REAL}_{\mathcal{U}}^{\mathcal{U},\lambda}$ *output.*

$$\mathsf{SIM}_{\mathcal{A}}^{\mathcal{U},\lambda}(\{s_k, e_k\}_{k\in\mathcal{A}}, pk)$$
$$\stackrel{c}{\equiv}\mathsf{REAL}_{\mathcal{U}}^{\mathcal{U},\lambda}(\{s_j, e_j\}_{j\in\mathcal{U}}, pk)$$

Proof. Since SIM has $\mathcal{A}$'s inputs $\{s_k, e_k\}, k \in \mathcal{A}$, and the output $pk = (p_0, p_1)$ of $\mathsf{MBFV}\cdot\mathsf{PubKeyGen}$, it needs to simulate all $p_{0,j} = [-(p_1 s_j + e_j)]_q, j \in \mathcal{U}$ under two constraints: (i) the sum of all simulated $p_{0,j}$ and those generated by $\mathcal{A}$ must be equal to p_0, and (ii) the simulated $p_{0,j}$ for $\mathcal{A}$ must be equal to the real ones, otherwise the adversary can easily distinguish them. We use the symbol $\tilde{p_{0,j}}$ to denote the simulated shares of p_0. SIM can generate $\tilde{p_{0,j}}$ in the following ways:

$$\tilde{p_{0,j}} = \begin{cases} -[(p_1 s_j + e_j)]_q : \text{ if user } j \in \mathcal{A} \\ sample\ from\ R_q : \text{ if user } j \in \mathcal{H} \\ [p_0 - \sum_{j\in\mathcal{A}\cup\mathcal{H}} \tilde{p_{0,j}}]_q : \text{ if user } j = g_h \end{cases}$$

We explain how the above simulation guarantees the indistinguishability between $(\tilde{p_{0,1}}, \tilde{p_{0,2}}, \cdots \tilde{p_{0,|\mathcal{U}|}})$ and $(p_{0,1}, p_{0,2}, \cdots p_{0,|\mathcal{U}|})$. Specifically, for each user $j \in \mathcal{A}$, since SIM has inputs $\{s_j, e_j\}$ from $\mathcal{A}$, it can generate the share $p_{0,j} = [-(p_1 s_j + e_j)]_q$, which is exactly the same as the real value. For each user $j \in \mathcal{H}$, SIM simulates $p_{0,j}$ by sampling an element uniformly in the distribution R_q. *Decisional-RLWE Problem* [24] ensures that the sampled value is indistinguishable from the real $p_{0,j}$. In addition, the property of Additive Secret Sharing [38] makes it a negligible probability to restore s_j and e_j of the honest user,

even if the collusion of multiple users. For user $j = g_h$, we consider the following two cases: (i) When $\mathcal{H} \neq \emptyset$, $\tilde{p_{0,j}}$ is uniformly random in the distribution R_q, since $\sum_{j\in\mathcal{A}\cup\mathcal{H}} \tilde{p_{0,j}}$ is a random value distributed in R_q. As a result, the same indistinguishability is achieved as described above. (ii) When $\mathcal{H} = \emptyset$, it means that $|\mathcal{U} - 1|$ users are adversaries. Since pk is open to all users, adversaries can reconstruct the share of user g_h through pk and their own knowledge. Therefore, SIM calculates and outputs the real value for the share of g_h. □

B. Analysis of MBFV · Convert

Similar to the above analysis, we consider an adversary set $\mathcal{A}$ to attack the function MBFV · Convert. The goal of $\mathcal{A}$ is to derive honest users' shares $\{h_{0,k}, h_{1,k}\}_{k\in\mathcal{H}\cup g_h}$. Hence, given $\mathcal{A}$'s inputs $\{s_k, \mu_k, e_{0,k}, e_{1,k}\}, k \in \mathcal{A}$, public key $pk_i = (p_{0,i}, p_{1,i})$, and original ciphertext $\mathsf{E}(W_{k+\frac{1}{2},i}) = (c_0, c_1)$, the simulator must construct a simulated view that is indistinguishable from the adversary's view under the implementation of the real protocol.

Theorem 2. *Given the security parameter λ, user set $\mathcal{U}$, adversary set $\mathcal{A} \subseteq \mathcal{U}, |\mathcal{A}| \leq |\mathcal{U}| - 1$, $\mathcal{A}$'s inputs $\{s_k, \mu_k, e_{0,k}, e_{1,k}\}_{k\in\mathcal{A}}$, $pk_i = (p_{0,i}, p_{1,i})$, original ciphertext $\mathsf{E}(W_{k+\frac{1}{2},i}) = (c_0, c_1)$, honest user g_h, and $\mathcal{H} = \mathcal{U}\backslash(\mathcal{A} \cup g_i)$, there exists a PPT simulator SIM, whose output is indistinguishable from the real $\mathsf{REAL}_{\mathcal{U}}^{\mathcal{U},\lambda}$ output.*

$$\begin{aligned}&\mathsf{SIM}_{\mathcal{A}}^{\mathcal{U},\lambda}(\{s_k, \mu_k, e_{0,k}, e_{1,k}\}_{k\in\mathcal{A}})\\ \overset{c}{\equiv}&\mathsf{REAL}_{\mathcal{U}}^{\mathcal{U},\lambda}(\{s_j, \mu_j, e_{0,j}, e_{1,j}\}_{j\in\mathcal{A}})\end{aligned}$$

Proof. We know that $\mathcal{A}$'s inputs $\{s_k, \mu_k, e_{0,k}, e_{1,k}\}_{k\in\mathcal{A}}$ are accessible to SIM. Based on this, SIM is required to simulate all $\{h_{0,j}, h_{1,j}\}_{j\in\mathcal{U}}$ under two constraints: (i) the sum of all simulated $h_{0,j}$ and $h_{1,j}$ must be equal to h_0 and h_1, if the recipient (i.e., user i) of the converted ciphertext is malicious, and (ii) the simulated $\{h_{0,k}, h_{1,k}\}$ for $\mathcal{A}$ must be equal to the real ones. Otherwise, the adversary can easily distinguish them. We use the symbols $\tilde{h_{0,j}}$ and $\tilde{h_{1,j}}$ to denote the simulated shares. SIM can generate $\tilde{h_{0,j}}$ and $\tilde{h_{1,j}}$ in the following ways:

$$(\tilde{h_{0,j}}, \tilde{h_{1,j}}) = \begin{cases} ([s_j c_1 + \mu_j p_{0,i} + e_{0,j}]_q, [\mu_k p_{1,i} + e_{1,j}]_q), \\ \qquad\qquad \text{if user } j \in \mathcal{A}. \\ sample\ from\ R_q, \text{if user } j \in \mathcal{H}. \\ sample\ from\ R_q, \text{if user } j = g_h \&\& \text{user } i \notin \mathcal{A} \\ ([h_0 - \sum_{j\in\mathcal{A}\cup\mathcal{H}} \tilde{h_{0,j}}]_q, [h_1 - \sum_{j\in\mathcal{A}\cup\mathcal{H}} \tilde{h_{1,j}}]_q), \\ \qquad\qquad \text{if user } j = g_h \&\& \text{user } i \notin \mathcal{A}. \end{cases}$$

We explain how the above simulation guarantees the indistinguishability between $(\tilde{h_{0,j}}, \tilde{h_{1,j}})_{j\in\mathcal{U}}$ and $\{h_{0,j}, h_{1,j}\}_{j\in\mathcal{U}}$. Specifically, for each user $j \in \mathcal{A}$, since SIM has $\mathcal{A}$'s inputs $\{s_k, \mu_k, e_{0,k}, e_{1,k}\}$, it can generate the share $p(\tilde{h_{0,j}}, \tilde{h_{1,j}}) = ([s_j c_1 + \mu_j p_{0,i} + e_{0,j}]_q, [\mu_k p_{1,i} + e_{1,j}]_q)$, which is exactly the same as the real value. For each user $j \in \mathcal{H}$, SIM simulates $\{h_{0,j}, h_{1,j}\}$ by sampling an element uniformly in the distribution χ. *Decisional-RLWE Problem* [24] ensures that the sampled value is indistinguishable from the real $\{h_{0,j}, h_{1,j}\}$. Besides, the property of adding secret sharing [38] makes it a negligible probability to restore $\{s_j, \mu_j, e_{0,j}, e_{1,j}\}$ to the honest user, even under the collusion of $|\mathcal{U}| - 2$ users. For user $j = g_h$, we consider the following two cases: (i) When user $i \notin \mathcal{A}$, $\tilde{h_{0,j}}, \tilde{h_{1,j}})$ is uniformly random on the distribution χ. This happens because the adversary cannot access the final values h_0 and h_1. Therefore, in this case, it is not necessary to ensure that the sum of all the simulated $h_{0,j}$ and $h_{1,j}$ is equal to h_0 and h_1. In addition, the same indistinguishability is achieved as described above. (ii) When $i \in \mathcal{A}$, it means that h_0 and h_1 are submitted to $\mathcal{A}$. Therefore, $(\tilde{h_{0,j}}, \tilde{h_{1,j}})$ can be constructed as $([h_0 - \sum_{j\in\mathcal{A}\cup\mathcal{H}} \tilde{h_{0,j}}]_q, [h_1 - \sum_{j\in\mathcal{A}\cup\mathcal{H}} \tilde{h_{1,j}}]_q)$. As a result, the sum of all simulated $h_{0,j}$ and $h_{1,j}$ is equal to h_0 and h_1. This guarantees the indistinguishability between the simulated view and the real view. □

C. Analysis of MBFV · Bootstrap

Finally, we discuss the security of the function MBFV · Bootstrap. Specifically, given the adversary set $\mathcal{A}$, the goal of $\mathcal{A}$ is to derive honest users' shares $\{\eta_{0,k}, \eta_{1,k}\}_{k\in\mathcal{H}\cup g_h}$. The simulator needs to construct a simulated view that is indistinguishable from the adversary's view with $\mathcal{A}$'s inputs $\{s_k, M_k, e_{0,k}, e_{1,k}\}, k \in \mathcal{A}$ and the public ciphertext $ct = (c_0, c_1)$.

Theorem 3. *Given the security parameter λ, user set $\mathcal{U}$, adversary set $\mathcal{A} \subseteq \mathcal{U}, |\mathcal{A}| \leq |\mathcal{U}| - 1$, $\mathcal{A}$'s inputs $\{s_k, M_k, e_{0,k}, e_{1,k}\}_{k\in\mathcal{A}}$, original ciphertext $ct = (c_0, c_1)$, honest user g_h, and $\mathcal{H} = \mathcal{U}\backslash(\mathcal{A} \cup g_i)$, there exists a PPT simulator SIM, whose output is indistinguishable from the real $\mathsf{REAL}_{\mathcal{U}}^{\mathcal{U},\lambda}$ output.*

$$\begin{aligned}&\mathsf{SIM}_{\mathcal{A}}^{\mathcal{U},\lambda}(\{s_k, M_k, e_{0,k}, e_{1,k}\}_{k\in\mathcal{A}})\\ \overset{c}{\equiv}&\mathsf{REAL}_{\mathcal{U}}^{\mathcal{U},\lambda}(\{s_j, M_j, e_{0,j}, e_{1,j}\}_{j\in\mathcal{A}})\end{aligned} \tag{5}$$

Proof. Given $\mathcal{A}$'s inputs $\{s_k, M_k, e_{0,k}, e_{1,k}\}_{k\in\mathcal{A}}$, SIM is required to simulate all $\{\eta_{0,j}, \eta_{1,j}\}_{j\in\mathcal{U}}$ under two constraints: (i) the sum of all simulated $\eta_{0,j}$ and $\eta_{1,j}$ must be equal to η_0 and η_1, if the recipient (i.e., user i) of the new ciphertext ct' is malicious, and (ii) the simulated $\{\eta_{0,j}, \eta_{1,j}\}$ for $\mathcal{A}$ must be equal to the real ones. Otherwise, the adversary can easily distinguish them. We use the symbols $\tilde{\eta_{0,j}}$ and $\tilde{\eta_{1,j}}$ to denote the simulated shares. SIM can generate $\tilde{\eta_{0,j}}$ and $\tilde{\eta_{1,j}}$ in the following ways:

$$(\tilde{\eta_{0,j}}, \tilde{\eta_{1,j}}) = \begin{cases} ([s_j c_1 - \Lambda M_j + e_{0,j}]_q, [-s_j \alpha + \Lambda M_j + e_{1,j}]_q), \\ \qquad\qquad \text{if user } j \in \mathcal{A}. \\ sample\ from\ R_q, \text{if user } j \in \mathcal{H} \\ sample\ from\ R_q,\ \text{if user } j = g_h \&\& \text{user } i \notin \mathcal{A}. \\ ([\eta_0 - \sum_{j\in\mathcal{A}\cup\mathcal{H}} \tilde{\eta_{0,j}}]_q, [\eta_1 - \sum_{j\in\mathcal{A}\cup\mathcal{H}} \tilde{\eta_{1,j}}]_q), \\ \qquad\qquad \text{if user } j = g_h \&\& \text{user } i \notin \mathcal{A}. \end{cases}$$

We explain how this simulation guarantees the indistinguishability between $(\tilde{\eta_{0,j}}, \tilde{\eta_{1,j}})_{j\in\mathcal{U}}$ and $\{\eta_{0,j}, \eta_{1,j}\}_{j\in\mathcal{U}}$. Specifically, for each user $j \in \mathcal{A}$, since SIM has inputs $\{s_j, M_j, e_{0,j}, e_{1,j}\}$ from $\mathcal{A}$, it can generate the share $p(\tilde{\eta_{0,j}}, \tilde{\eta_{1,j}}) = ([s_j c_1 - \Lambda M_j + e_{0,j}]_q, [-s_j \alpha + \Lambda M_j + e_{1,j}]_q)$, which is exactly the

same as the real value. For each user $j \in \mathcal{H}$, SIM simulates $\{\eta_{0,j}, \eta_{1,j}\}$ by sampling an element uniformly in the distribution R_q. The *Decisional-RLWE Problem* [24] ensures that the sampled value is indistinguishable from the real $\{\eta_{0,j}, \eta_{1,j}\}$. Besides, the property of Additive Secret-Sharing [38] makes it a negligible probability to restore $\{s_j, M_j, e_{0,j}, e_{1,j}\}$ to the honest user, even under the collusion of $|\mathcal{U}|-2$ users. For user $j = g_h$, we consider the following two cases: (i) When user $i \notin \mathcal{A}$, $\eta_{\tilde{0},j}, \eta_{\tilde{1},j})$ is uniformly random on the distribution R_q. This is because the adversary cannot access the final values η_0 and η_1. Therefore, in this case, it is not necessary to ensure that the sum of all simulated $\eta_{0,j}$ and $\eta_{1,j}$ is equal to η_0 and η_1. In addition, the same indistinguishability is achieved as described above. (ii) When $i \in \mathcal{A}$, it means that η_0 and η_1 are submitted to $\mathcal{A}$. Hence, $(\eta_{\tilde{0},j}, \eta_{\tilde{1},j})$ can be constructed as $([\eta_0 - \sum_{j\in\mathcal{A}\cup\mathcal{H}} \eta_{\tilde{0},j}]_q, [\eta_1 - \sum_{j\in\mathcal{A}\cup\mathcal{H}} \eta_{\tilde{1},j}]_q)$. As a result, the sum of all simulated $\eta_{0,j}$ and $\eta_{1,j}$ is equal to η_0 and η_1. This guarantees the indistinguishability between the simulated view and the real view. □

VI. Performance Evaluation

We evaluate the performance of **D^2-MHE** in terms of classification accuracy, computation and communication overheads. Specifically, we simulate a decentralized system with varied numbers of users using Pytorch, where we make use of Onet[3] to build the decentralized communication protocol. The average connection rate is $A_{\mathcal{N}} = 0.2$ (i.e., each user is randomly connected to 20% of all users in the system). Our multiparty version of BFV is modified based on the standard BFV in the SEAL library [54], where the security parameters are taken as 2048 and 4096, respectively, to test the performance. The Smart-Vercauteren ciphertext packing technique [55] is used to accelerate the efficiency of encryption and ciphertext computation: we set the plaintext slot to 1024, which can pack 1024 plaintexts into one ciphertext at a time, and support Single-Instruction-Multiple-Data (SIMD) operations. We consider three image classification tasks trained in the decentralized learning system: (1) a MLP model with two fully connected layers (100 and 10 neurons, respectively) for MNIST; (2) a CNN model with two convolutional layers (kernel size of 3×1 per layer) and three fully connected layers (384 neurons per hidden layer and 10 neurons in the output layer) for CIFAR-10, and (3) ResNet-18 and EfficientNet-B0 to train ImageNet, respectively. All experiments are carried out on a server running Centos7.4 OS, equipped with 256G-B RAM, 64 CPUs (Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHZ), and 8 GPUs (Tesla V100 32G).

We select the following baselines for comparison. (1) D-PSGD [56] is the original D-PSGD algorithm without any privacy protection. (2) LEASGD [30], $A(DP)^2SGD$ [31] and DLDP [8] are the three most advanced decentralized learning algorithms with differential privacy. We reproduce these algorithms using exactly the same parameter configuration as the authors in their papers. (3) Threshold Paillier-HE [57], [46] is a classic homomorphic encryption algorithm that supports distributed key encryption and decryption operations. We extend the threshold Paillier-HE to the decentralized mode for comparison. (4) COPML [58] is a distributed learning framework based on Shamir's secret sharing protocol, which can be considered a special kind of decentralized learning with a connection rate of $A_{\mathcal{N}} = 1$. It is also feasible to adapt this framework to a generalized decentralized network.

[3]https://github.com/dedis/cothority

A. Classification Accuracy

We first discuss the performance of **D^2-MHE** on the model classification accuracy. Table I shows the performance comparisons of **D^2-MHE** with existing approaches in the decentralized settings of 50 and 100 users, where we use ImageNet-Res and ImageNet-Eff to denote the ImageNet classification task over ResNet-18 and EfficientNet-B0, repectively. Compared with D-PSGD, we observe that the accuracy drop of HE-based solutions (including **D^2-MHE** and Paillier-HE) is negligible, which is mainly attributed to the losslessness of the HE encryption algorithm. Although HE can only handle integers in ciphertext, existing optimization methods (e.g., conversion of fixed point arithmetic circuits [59]) ensure that the error of ciphertext evaluation for any floating point number is maintained within 2^{-d} (usually $d \geq 13$).

In contrast, other three works based on differential privacy inevitably result in a large accuracy drop even if the privacy budget $\epsilon > 8$, which is already vulnerable to various types of privacy inference attacks[4]. Here we take the membership attack [60] under $\epsilon = 8$ as an example. Based on the definition of differential privacy [61], the condition $Pr[F(D) \in \mathbf{S}] \leq e^8 \times Pr[F(D') \in \mathbf{S}]$ should be guaranteed for any two neighboring sets D and D'. In other words, even if the target record detected in the dataset D has a probability of 0.0001, it can be detected with a probability of up to 0.9999 in D' that contains the record. This allows the adversary to infer the presence or absence of the target record from the training data with very high confidence.

Remark 3: Note that HE-based schemes always exhibit superiority in accuracy over DP-based schemes, as the latter obtains a proper trade-off between accuracy and privacy by introducing noise. However, the comparison with the DP-based solution is not only to illustrate the advantages of our method in accuracy; a more noteworthy conclusion is that it is still unclear whether DP-based algorithms can provide satisfactory accuracy and privacy trade-offs in practical applications. Our experimental results are consistent with the results in work [32], i.e., current mechanisms for differentially private deep learning may rarely offer acceptable accuracy-privacy trade-offs for complex learning tasks. Therefore, one of the main motivations to compare with DP is to explain the choice to use HE primitives, which may provide better accuracy and privacy performance.

B. Computation Overhead

We further analyze the computation cost of **D^2-MHE**. In summary, the computational load of each user depends

[4]According to [32], differential privacy with $\epsilon > 1$ will loss its effectiveness for deep learning training

TABLE I: Classification accuracy of different privacy-preserving approaches

# of users	Dataset	D-PSGD	LEASGD	$A(DP)^2SGD$	DLDP	**D^2-MHE**	Paillier-HE
50	MNIST	91.23%	87.67% ($\epsilon = 8.71$)	84.68% ($\epsilon = 9.43$)	86.7% ($\epsilon = 9.21$)	91.17%	91.19%
	CIFAR-10	65.21%	57.61% ($\epsilon = 11.23$)	53.44% ($\epsilon = 12.13$)	50.7% ($\epsilon = 9.81$)	65.13%	65.10%
	ImageNet-Res	69.75%	59.61% ($\epsilon = 14.23$)	56.47% ($\epsilon = 13.17$)	56.7% ($\epsilon = 9.88$)	69.74%	69.23%
	ImageNet-Eff	77.37%	59.64% ($\epsilon = 16.23$)	57.44% ($\epsilon = 13.15$)	56.3% ($\epsilon = 9.69$)	77.32%	77.20%
100	MNIST	92.43%	88.89% ($\epsilon = 8.91$)	85.82% ($\epsilon = 9.79$)	87.15% ($\epsilon = 9.98$)	91.18%	91.14%
	CIFAR-10	68.32%	59.69% ($\epsilon = 12.49$)	54.14% ($\epsilon = 13.58$)	52.9% ($\epsilon = 11.31$)	68.25%	68.26%
	ImageNet-Res	74.75%	59.41% ($\epsilon = 13.53$)	56.47% ($\epsilon = 13.36$)	56.72% ($\epsilon = 9.71$)	74.74%	74.60%
	ImageNet-Eff	80.41%	60.61% ($\epsilon = 13.23$)	59.44% ($\epsilon = 12.53$)	59.7% ($\epsilon = 9.61$)	80.41%	80.37%

TABLE II: Computation overhead of different privacy-preserving approaches for each user (unit: seconds)

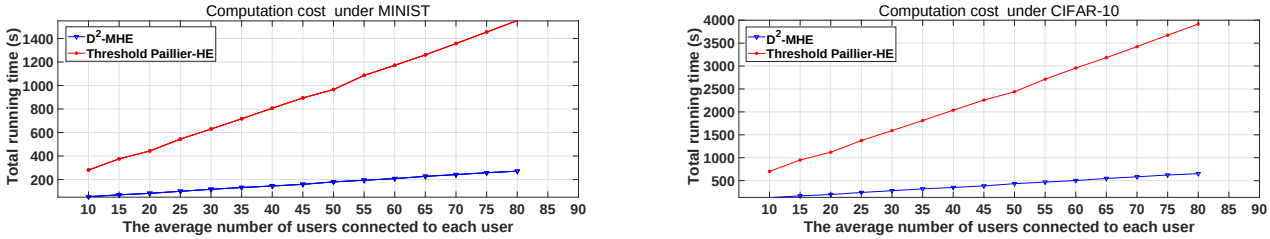
Key Size	Dataset	Method	Initialization	Encryption	Ciphertext Evaluation	Decryption	Total time
2048	MNIST	D-PSGD	-	-	-	-	2.47
		D^2-MHE	0.66	9.96	0.54	15.04	26.2
		Paillier-HE	2.93	21.93	1.27	33.41	59.54
	CIFAR-10	D-PSGD	-	-	-	-	17.18
		D^2-MHE	0.67	23.98	1.31	36.23	62.19
		Paillier-HE	3.01	43.69	2.48	57.24	106.42
	ImageNet-Eff	D-PSGD	-	-	-	-	49.37
		D^2-MHE	0.69	207.98	14.31	257.23	480.21
		Paillier-HE	3.42	230.69	39.48	402.24	675.83
4096	MNIST	D-PSGD	-	-	-	-	2.30
		D^2-MHE	1.42	21.41	1.24	32.62	56.69
		Paillier-HE	18.98	139.25	2.38	148.98	309.59
	CIFAR-10	D-PSGD	-	-	-	-	17.23
		D^2-MHE	1.41	51.56	2.99	78.55	134.51
		Paillier-HE	19.24	335.28	5.75	358.68	718.95
	ImageNet-Res	D-PSGD	-	-	-	-	58.14
		D^2-MHE	1.47	241.23	17.31	266.23	526.24
		Paillier-HE	21.31	992.68	93.45	1277.34	2384.78

mainly on the key size used for encryption and the average number of users connected to it. Intuitively, a user needs more computing resources to handle operations with a larger key size and interact with more users during the training process. To demonstrate this, we first fix the number of users as 100 in the system and record the running time of each user in a single iteration (i.e., a gradient update with a mini-batch of 256). Table II shows the experimental results compared to some baseline methods. To facilitate analysis, we divide the total computation cost into four components: (1) *Initialization* is to prepare the public and secret keys of the system. This only needs to be executed once for both **D^2-MHE** and Threshold Paillier-HE. (2) *Encryption* is used to encrypt the gradients of each user. (3) *Ciphertext Evaluation* is used to perform the cipertext computation. (4) *Decryption* is carried out to decrypt the final results.

From Table II, we observe that the overhead of **D^2-MHE** and Threshold Paillier-HE is larger than D-PSGD, because all gradients are encrypted and processed under ciphertext. However, the overhead of **D^2-MHE** is significantly lower than that of Threshold Paillier-HE, especially for large key sizes. This is mainly due to the following two reasons: (i) The key-sharing and reconstruction processes in Threshold Paillier-HE (including the Initialization and Decryption phases) are highly affected by the key size. A large key size makes it inevitable to perform modular exponential calculations in a large ciphertext space, thereby completing key distribution and distributed decryption. On the contrary, **D^2-MHE** only involves vector operations in the traditional sense, which is much less affected by the key size. (ii) Compared to the threshold Paillier-HE, the BFV cryptosystem is more suitable for SIMD technology, which can process multiple ciphertexts in parallel more efficiently.

We also evaluate the impact of connection rates on the computation overhead of **D^2-MHE**. We fix the number of users in the system to 100, and change the connection rate from 0.1 to 0.8. The key size of both **D^2-MHE** and Threshold Paillier-HE is 4096 bit. Smart-Vercauteren ciphertext packing techniques [55] are used to accelerate the efficiency of encryption and ciphertext computation, where we set the plaintext slot as 1024 to pack 1024 plaintexts into one ciphertext at a time to support Single-Instruction-Multiple-Data (SIMD) operations. Figure 3 shows the running time of each user in a single iteration (i.e., a gradient update with a mini-batch of 256), where we do not experiment on ImageNet since the results are similar to other datasets. We can observe that as the connection rate increases, **D^2-MHE** has more significant advantages over Threshold Paillier-HE in terms of computation overhead. This is mainly due to the inefficiency of Threshold Paillier-HE distributed decryption. As the average number of users connected to each user increases, the number of modular exponential operations performed by Threshold Paillier-HE increases linearly. As a result, it is quite time-consuming to recover the secret key of the system under the ciphertext through exponential operations, thereby decrypting the target ciphertext. On the contrary, **D^2-MHE** only involves vector operations in the traditional sense, which is much less affected by changes in the connection rate compared to Threshold

This article has been accepted for publication in IEEE Transactions on Circuits and Systems for Video Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2023.3234278



(a) MNIST (b) CIFAR10

Fig. 3: Total running time of each user for different connection rates.

Paillier-HE.

C. Communication Overhead

TABLE III: Theoretical communication overhead of the user i for different approaches

Method	Gradients-sharing	Aggregation	Total
D-PSGD	$O(\lambda_3)$	$O(\lvert\mathcal{N}_i\rvert\lambda_3)$	$O((\lvert\mathcal{N}_i\rvert+1)\lambda_3)$
COPML	$O(\lvert\mathcal{N}_j\rvert\lambda_1)$	$O(\lvert\mathcal{N}_i\rvert\lambda_1)$	$O(\sum_{j\in N,E_{i,j}\neq 0}\lvert\mathcal{N}_j\rvert\lambda_1+\lvert\mathcal{N}_i\rvert\lambda_1)$
D²-MHE	$O(\lambda_2)$	$O(2\lvert\mathcal{N}_i\rvert\lambda_2)$	$O((2\lvert\mathcal{N}_i\rvert+1)\lambda_2)$

We finally analyze the performance of **D²-MHE** in terms of communication overhead, We theoretically compare the communication complexity of **D²-MHE** with D-PSGD and COPML. The results are shown in Table III, where λ_i $(i = 1, 2, 3)$ denotes the size of a single message. The total computation costs are divided into 2 components (i.e., Gradients-sharing and Aggregation) to facilitate our analysis. Specifically, in the Gradients-sharing phase, each user i in COPML is required to share its every gradient to each user set $(\mathcal{N}_j | j \in N, E_{i,j} \neq 0)$, which results in the communication complexity of $O(\sum_{j\in N,E_{i,j}\neq 0}|\mathcal{N}_j|\lambda_1)$. In contrast, in **D²-MHE**, the gradients of all users are encrypted with the same public key. As a result, the user i only needs to transmit a single gradient to other users. In the Aggregation phase (lines 6-8 in **Algorithm 2**), the complexity of COPML is consistent with that of D-PSGD, that is, the information returned by each user to generate the aggregated gradient. In general, compared to D-PSGD, the communication complexity of **D²-MHE** is only increased by a constant multiple, while the complexity of COPML can reach $O(\sum_{j\in N,E_{i,j}\neq 0}|\mathcal{N}_j|\lambda_1+|\mathcal{N}_i|\lambda_1)$.

TABLE IV: Experimental communication overhead of user i for different approaches and datasets (unit: MB)

Dataset	Model	Gradients-sharing	Aggregation	Total
MNIST	COPML	674	33.7	707.7
	D²-MHE	18.9	37.8	56.7
CIFAR-10	COPML	3235	161.76	3396.76
	D²-MHE	90.72	181.44	272.16
ImageNet-Eff	COPML	10237	279.43	10516.43
	D²-MHE	363.38	246.35	609.73
ImageNet-Res	COPML	31726	549.51	32275.51
	D²-MHE	484.52	329.17	813.69

It should be noted that COPML uses a packed secret sharing method to reduce the complexity of communication from $O(\sum_{j\in N,E_{i,j}\neq 0}|\mathcal{N}_j|\lambda_1+|\mathcal{N}_i|\lambda_1)$ to $O(\frac{\sum_{j\in N,E_{i,j}\neq 0}|\mathcal{N}_j|\lambda_1+|\mathcal{N}_i|\lambda_1)}{K})$, where K is the number of secrets packed each time. However, packed secret sharing [62] is restricted to $K < \min(\mathcal{N}_j | j \in N, E_{i,j} \neq 0)$ and only tolerates the collusion of $\min|(\mathcal{N}_j | j \in N, E_{i,j} \neq 0)| - K$ users at most. On the contrary, we use the Smart-Vercauteren ciphertext packing technique [55] to pack multiple plaintexts into one ciphertext, where the number of plaintext slots is independent of the number of users in the system. As a result, compared with existing works, **D²-MHE** has a significant advantage in communication overhead.

We also present the experimental results in terms of communication overhead. We define the sizes of a single message in COPML and **D²-MHE** as 64 bit and 4096 bit, respectively. Such parameters are commonly used to ensure the security of the MPC protocol and the HE. Additionally, the system has 100 users with a connection rate of 0.2. Note that, for communication overhead, the connection rate exhibits a linear relationship with each user in COPML, but has no effect on our method. To be precise, the increase of the connection rate makes the number of adjacent nodes of each user increase linearly. Since the communication cost of each user has a positive linear relationship with the number of adjacent nodes, this implies a linear relationship between the connection rate and the communication cost of each user. However, our method requires each user only to transmit a ciphertext to all users, regardless of the value of the connection rate. We iteratively execute the above two schemes 500 times and 1000 times under the MNIST, CIFAR-10, and ImageNet datasets, respectively. Then we record the total communication overhead in Table IV. For simplicity, we assume $\min(\mathcal{N}_j | j \in N, E_{i,j} \neq 0) = 10$, so the maximum number of secrets shared by the package sharing protocol in COPML is $K < 10 = 9$. Furthermore, the average number of $(\sum_{j\in N,E_{i,j}\neq 0}|\mathcal{N}_j|)$ is set to 20. In our **D²-MHE**, the plaintext slot of the Smart-Vercauteren ciphertext packing technique [55] is 1024, which can pack 1024 plaintexts into one ciphertext at a time. We can observe that compared with COPML, **D²-MHE** has a significant advantage in the communication overhead. This is mainly due to the large number of interactions in the gradient sharing process of COPML. Moreover, we use Smart-Vercauteren ciphertext packing techniques [55], [63] to pack

multiple plaintexts into one ciphertext, where the number of plaintext slots is independent of the number of users in the system.

VII. Conclusion

In this work, we propose **D^2-MHE**, a practical, privacy-preserving, and high-fidelity decentralized deep learning framework. To the best of our knowledge, **D^2-MHE** is the first work to protect the privacy and accelerate the performance of decentralized learning systems using cryptographic primitives. Experimental results show that **D^2-MHE** can provide the optimal accuracy-performance trade-off compared to other state-of-the-art works. In the future, we will focus on improving the computation overhead of **D^2-MHE**, since this is the main bottleneck of the current homomorphic encryption applied to real-world applications.

ACKNOWLEDGMENT

This work was supported in part by Singapore Ministry of Education (MOE) AcRF Tier 1 RG108/19 (S),Nanyang Technological University (NTU)-DESAY SV Research Program under Grant 2018-0980, NTU Start-Up Grant, Singapore Ministry of Education (MOE) AcRF Tier 2 MOE-T2EP20121-0006, Singapore National Research Foundation (NRF) under its National Cybersecurity R&D Program (NRF2018NCR-NCR005-0001 and NRF2018NCR-NSOE003-0001), and NRF Investigatorship (NRFNRFI06-2020-0001).

References

[1] S. Guo, T. Zhang, G. Xu, H. Yu, T. Xiang, and Y. Liu, "Topology-aware differential privacy for decentralized image classification," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[2] P. Sun, T. Liu, X. Chen, S. Zhang, Y. Zhao, and S. Wei, "Multi-source aggregation transformer for concealed object detection in millimeter-wave images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 9, pp. 6148–6159, 2022.

[3] S. Guo, T. Zhang, H. Yu, X. Xie, L. Ma, T. Xiang, and Y. Liu, "Byzantine-resilient decentralized stochastic gradient descent," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 6, pp. 4096–4106, 2022.

[4] X. Lou, S. Guo, J. Li, and T. Zhang, "Ownership verification of dnn architectures via hardware cache side channels," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[5] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J.-P. Bossuat, J. S. Sousa, and J.-P. Hubaux, "Poseidon: Privacy-preserving federated neural network learning," in *Proceedings of NDSS*, 2021.

[6] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proceedings of NeurIPS*, vol. 32, 2019.

[7] Y. Zhang and W. Luo, "Vector-based efficient data hiding in encrypted images via multi-msb replacement," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2022.

[8] H.-P. Cheng, P. Yu, H. Hu, S. Zawad, F. Yan, S. Li, H. Li, and Y. Chen, "Towards decentralized deep learning with differential privacy," in *International Conference on Cloud Computing*. Springer, 2019, pp. 130–145.

[9] H. Xiao, Y. Ye, and S. Devadas, "Local differential privacy in decentralized optimization," *arXiv preprint arXiv:1902.06101*, 2019.

[10] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 2090–2103, 2020.

[11] W. Zheng, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Helen: Maliciously secure coopetitive learning for linear models," in *IEEE Symposium on S&P*, 2019, pp. 724–738.

[12] D. Froelicher, J. R. Troncoso-Pastoriza, A. Pyrgelis, S. Sav, J. S. Sousa, J.-P. Bossuat, and J.-P. Hubaux, "Scalable privacy-preserving distributed learning," in *Proceedings of PETs*, 2021.

[13] J. Li, M. Khodak, S. Caldas, and A. Talwalkar, "Differentially private meta-learning," in *Proceedings of ICLR*, 2020.

[14] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow: Secure tensorflow inference," in *IEEE Symposium on Security and Privacy*. IEEE, 2020, pp. 336–353.

[15] H. Chaudhari, R. Rachuri, and A. Suresh, "Trident: Efficient 4pc framework for privacy preserving machine learning," in *Proceedings of NDSS*, 2020.

[16] A. Patra and A. Suresh, "Blaze: blazing fast privacy-preserving machine learning," *Proceedings of NDSS*, pp. 1–18, 2020.

[17] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow2: Practical 2-party secure inference," in *Proceedings of ACM CCS*, 2020, pp. 325–342.

[18] Q. Zhang, C. Xin, and H. Wu, "Gala: Greedy computation for linear algebra in privacy-preserved neural networks," in *Proceedings of NDSS*, 2021, pp. 1–18.

[19] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference," in *Proceedings of ACM CCS*, 2019, pp. 395–412.

[20] Y. Xian and X. Wang, "Fractal sorting matrix and its application on chaotic image encryption," *Information Sciences*, vol. 547, pp. 1154–1169, 2021.

[21] Y. Xian, X. Wang, and L. Teng, "Double parameters fractal sorting matrix and its application in image encryption," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[22] Y. Xian, X. Wang, X. Wang, Q. Li, and X. Yan, "Spiral-transform-based fractal sorting matrix for chaotic image encryption," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2022.

[23] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption." *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 144, 2012.

[24] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proceedings of ASIACRYPT*. Springer, 2017, pp. 409–437.

[25] L. Ducas and D. Micciancio, "Fhew: bootstrapping homomorphic encryption in less than a second," in *Proceedings of EUROCRYPT*. Springer, 2015, pp. 617–640.

[26] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *IEEE Symposium on Security and Privacy*. IEEE, 2019, pp. 332–349.

[27] R. McKenna, D. Sheldon, and G. Miklau, "Graphical-model based estimation and inference for differential privacy," in *Proceedings of ICML*. PMLR, 2019, pp. 4435–4444.

[28] D. Bernau, G. Eibl, P. Grassal, H. Keller, and F. Kerschbaum, "Quantifying identifiability to choose and audit epsilon in differentially private deep learning," *Proc. VLDB Endow.*, vol. 14, no. 13, pp. 3335–3347, 2021.

[29] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2018, pp. 473–481.

[30] H.-P. Cheng, P. Yu, H. Hu, F. Yan, H. Li, Y. Chen *et al.*, "Leasgd: an efficient and privacy-preserving decentralized algorithm for distributed learning," in *Proceedings of PPML*, 2018.

[31] J. Xu, W. Zhang, and F. Wang, "A (dp) ^{2} sgd: Asynchronous decentralized parallel stochastic gradient descent with differential privacy," *arXiv preprint arXiv:2008.09246*, 2020.

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

[33] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of ACM CCS*, 2017, pp. 603–618.

[34] I. Damgård, D. Escudero, T. Frederiksen, M. Keller, P. Scholl, and N. Volgushev, "New primitives for actively-secure mpc over rings with applications to private machine learning," in *IEEE Symposium on Security and Privacy)*. IEEE, 2019, pp. 1102–1120.

[35] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "Xonn: Xnor-based oblivious deep neural network inference," in *USENIX Security Symposium*, 2019, pp. 1501–1518.

[36] N. Agrawal, A. Shahin Shamsabadi, M. J. Kusner, and A. Gascón, "Quotient: two-party secure neural network training and prediction," in *Proceedings of ACM CCS*, 2019, pp. 1231–1247.

[37] F. Benhamouda, A. Degwekar, Y. Ishai, and T. Rabin, "On the local leakage resilience of linear secret sharing schemes," *Journal of Cryptology*, vol. 34, no. 2, pp. 1–65, 2021.

[38] C. Mouchet, J. Troncoso-Pastoriza, J.-P. Bossuat, and J.-P. Hubaux, "Multiparty homomorphic encryption from ring-learning-with-errors," Cryptology ePrint Archive, Report 2020/304. https://eprint. iacr. org/2020/304, Tech. Rep., 2020.

[39] J. So, B. Guler, and S. Avestimehr, "A scalable approach for privacy-preserving collaborative machine learning," in *Proceedings of NeurIPS*, vol. 33, 2020, pp. 8054–8066.

[40] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.

[41] H. Chen, I. Chillotti, and Y. Song, "Improved bootstrapping for approximate homomorphic encryption," in *Proceedings of EUROCRYPT*. Springer, 2019, pp. 34–54.

[42] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proceedings of EUROCRYPT*. Springer, 2010, pp. 24–43.

[43] C. Gentry, S. Halevi, and N. P. Smart, "Fully homomorphic encryption with polylog overhead," in *Proceedings of EUROCRYPT*. Springer, 2012, pp. 465–482.

[44] H. Chen, I. Chillotti, and Y. Song, "Multi-key homomorphic encryption from tfhe," in *Proceedings of ASIACRYPT*. Springer, 2019, pp. 446–472.

[45] N. Genise, C. Gentry, S. Halevi, B. Li, and D. Micciancio, "Homomorphic encryption for finite automata," in *Proceedings of ASIACRYPT*. Springer, 2019, pp. 473–502.

[46] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," in *Proceedings of EUROCRYPT*. Springer, 2001, pp. 280–300.

[47] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. Rasmussen, and A. Sahai, "Threshold cryptosystems from threshold fully homomorphic encryption," in *Proceedings of CRYPTO*. Springer, 2018, pp. 565–596.

[48] T. Vogels, S. P. Karimireddy, and M. Jaggi, "Practical low-rank communication compression in decentralized deep learning," *Proceedings of NeurIPS*, vol. 33, 2020.

[49] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *Proceedings of ICLR*, 2020.

[50] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter *et al.*, "Homomorphic encryption standard," in *Protecting Privacy through Homomorphic Encryption*. Springer, 2021, pp. 31–62.

[51] E. Boyle, N. Gilboa, and Y. Ishai, "Function secret sharing," in *Proceedings of EUROCRYPT*. Springer, 2015, pp. 337–367.

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

[53] R. Canetti, A. Jain, and A. Scafuro, "Practical uc security with a global random oracle," in *Proceedings of ACM CCS*, 2014, pp. 597–608.

[54] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library-seal v2. 1," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 3–18.

[55] G. Xu, H. Li, H. Ren, J. Sun, S. Xu, J. Ning, H. Yang, K. Yang, and R. H. Deng, "Secure and verifiable inference in deep neural networks," in *Proceedings of ACM ACSAC*, 2020, pp. 784–797.

[56] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Proceedings of neurIPS*, 2017, pp. 5336–5346.

[57] C. Hazay, G. L. Mikkelsen, T. Rabin, T. Toft, and A. A. Nicolosi, "Efficient rsa key generation and threshold paillier in the two-party setting," *Journal of Cryptology*, vol. 32, no. 2, pp. 265–323, 2019.

[58] E. Dawson and D. Donovan, "The breadth of shamir's secret-sharing scheme," *Computers & Security*, vol. 13, no. 1, pp. 69–78, 1994.

[59] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *USENIX Security*, 2018, pp. 1651–1669.

[60] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang, "Membership inference attack against differentially private deep learning model." *Trans. Data Priv.*, vol. 11, no. 1, pp. 61–79, 2018.

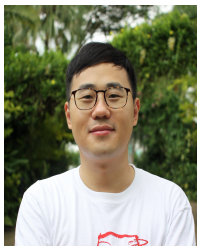
[61] M. Jagielski, J. Ullman, and A. Oprea, "Auditing differentially private machine learning: How private is private sgd?" in *Proceedings of NeurIPS*, vol. 33, 2020, pp. 22 205–22 216.

[62] R. Cramer, I. B. Damgård *et al.*, *Secure multiparty computation*. Cambridge University Press, 2015.

[63] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu, "Efficient and privacy-preserving truth discovery in mobile crowd sensing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3854–3865, 2019.

Guowen Xu is currently a Research Fellow with Nanyang Technological University, Singapore. He received his Ph.D. degree in 2020 from the University of Electronic Science and Technology of China. He has published a wealth of papers in reputable conferences/journals, including ACM CCS, NeurIPS, ECCV, IEEE TIFS, TDSC, ASIACCS, ACSAC, ESORICS, etc. He is the recipient of the Best Paper Award of the 26th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2020), the Best Student Paper Award of Sichuan Province Computer Federation (SCF 2019), the Student Conference Award of IEEE International Conference on Computer Communications (INFOCOM 2020), and the Distinguished Reviewer of ACM Transactions on the Web. His research interests include applied cryptography and privacy-preserving Deep Learning.

Guanlin Li is currently a Ph.D. student in the School of Computer Science and Engineering, Nanyang Technological University. He received his bachelor's degree in information security from the Mathematics School of Shandong University, Shandong, China in 2018. He has published papers in reputable conferences/journals, including CVPR and ECCV. His research interests include deep learning, computer vision, adversarial examples, and neural network security.



Shangwei Guo is an associate professor in College of Computer Science, Chongqing University. He received the Ph.D. degree in computer science from Chongqing University, Chongqing, China at 2017. He worked as a postdoctoral research fellow at Hong Kong Baptist University and Nanyang Technological University from 2018 to 2020. He has published papers in reputable conferences and journals, including IEEE TCSVT, TIFS, CVPR, and ICLR. His research interests include deep learning, cloud / edge computing, and database security.

Tianwei Zhang is an assistant professor in School of Computer Science and Engineering, at Nanyang Technological University. He currently serves as the Associate Editor of IEEE Transactions on Circuits and Systems for Video Technology, the Guest Editor of ACM Transactions on Sensor Networks, and the Senior PC of AAAI 2023. He has published a wealth of papers in reputable conferences/journals, including ACM CCS, NeurIPS, CVPR, ICLR, ECCV, IEEE TCSVT, TIFS, TDSC, etc. His research focuses on computer system security. He is particularly interested in security threats and defenses in machine learning systems, autonomous systems, computer architecture, and distributed systems. He received his Bachelor's degree at Peking University in 2011 and his Ph.D. degree from Princeton University in 2017.

Hongwei Li is currently the Head and a Professor at the Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include network security and applied cryptography. Prof. Li won the Best Paper Award from IEEE MASS 2018, IEEE HEALTHCOM 2015 and IEEE ICPADS 2020. He serves as the Associate Editor for the IEEE INTERNET OF THINGS JOURNAL and Peer-to-Peer Networking and Applications, the Lead Guest Editor for IEEE Network, IEEE Transactions on Vehicular Technology, and IEEE INTERNET OF THINGS JOURNAL. He also serves/served as the technical Symposium Co-Chair of IEEE ICC 2022, ACM TUR-C 2019, IEEE ICCC 2016, IEEE GLOBECOM 2015, and IEEE BigDataService 2015, and many technical program committees for international conferences, such as IEEE INFOCOM, IEEE ICC, IEEE GLOBECOM, IEEE WCNC, IEEE SmartGridComm, BODYNETS, and IEEE DASC. He is the Distinguished Lecturer of IEEE Vehicular Technology Society.