Dual-Server Privacy-Preserving Collaborative Deep Learning: A Round-Efficient, Dynamic and Lossless Approach

Lulu Wang, Lei Zhang, *Member, IEEE*, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, Josep Domingo-Ferrer, *Fellow, IEEE*, Mauro Conti, *Fellow, IEEE*, and Yuanyuan Gao

***Abstract*—To address limitations in existing privacy-preserving collaborative deep learning (CDL) schemes, we propose a dual-server privacy-preserving CDL scheme based on homomorphic encryption and a *masking technique*. Specifically, in our scheme a random seed is used to initialize a pseudorandom generator that produces multiple pseudorandom numbers. These pseudorandom numbers, along with a random noise, are utilized to generate masks that are added to all parameters of a participant's locally trained model. By using homomorphic encryption, the random noise can be encrypted and eventually used to remove the masks with low message expansion. This also ensures that the global model is lossless in accuracy. Furthermore, if participants join or leave the system, only the time required to complete both model update aggregation and encrypted masks aggregation is affected. We demonstrate that our scheme is round-efficient, dynamic and lossless. We also show that it is secure against inference attacks and can resist collusion attacks of up to $t-2$ participants and one of the two servers, where t is a security parameter indicating the minimum number of participants that participate in an aggregation round.**

***Index Terms*—Deep learning, collaborative deep learning, federated learning, privacy, homomorphic encryption.**

I. Introduction

DEEP learning (DL) has traditionally relied on centralized datasets for model training. However, with the increasing digitization of society, valuable data are often siloed within separate organizations (also known as *data islands*). These entities are frequently unwilling or unable to share their data, for example due to commercial interests, competitive concerns, or stringent legal and regulatory frameworks such as the EU General Data Protection Regulation (GDPR) [1] and the California Consumer Privacy Act (CCPA) [2].

These requirements compound the challenges of utilizing distributed data for collaborative model training. Consequently, a number of collaborative DL (CDL) approaches have been proposed in the literature and examples include those presented in [3], [4], [5], [6]. Federated learning (FL) [7], [8], [9], [10], [11], [12] is a representative paradigm in this context. In FL, a central server (known as the model aggregator), coordinates multiple participants who each hold local datasets. When the system is initialized, the server assigns an initial global model to all participants. Each participant then uses its local dataset to update the model parameters and sends the corresponding model update to the server. Then in the aggregation stage, the latter uses the model averaging algorithm [13] to merge model updates from all participants into an updated global model, prior to sending the model to all other participants. This is an iterative process that will conclude once a termination condition is reached.

Data privacy is a fundamental challenge in CDL. The main operation in CDL is to calculate the weighted average of all participants' local model updates during the aggregation stage, in which the weight of a participant's update is determined by the size of the particular participant's dataset vs the size of the overall dataset of all participants [14]. Without effective privacy mechanisms, an attacker may observe and use the model updates computed by a participant to *indirectly* infer that participant's local data or local model, even if the attacker cannot directly access the participant's dataset [15], [16]. The attacker can be a curious central server, another participant, or an external entity. Furthermore, two or more attackers may even launch collusion attacks to target specific honest participant(s). If the model update's generalization ability is weak [17], then an attacker can use the update to infer sensitive attributes in the local dataset of the corresponding participant [18]. Such a risk is real, as partly evidenced by the number of concrete inference attacks, i.e., membership inference attacks [19] and model inversion attacks [20] described in the literature.

A. Related Work

Privacy in CDL can be realized by using some cryptographic primitives, typically secure multi-party computation

Received 14 April 2023; revised 11 August 2025; accepted 12 August 2025. Date of publication 19 August 2025; date of current version 4 November 2025. This work was supported by the NSF of China under Grant 62372177. The work of Kim-Kwang Raymond Choo was supported by the Cloud Technology Endowed Professorship. *(Corresponding author: Lei Zhang.)*

Lulu Wang, Lei Zhang, and Yuanyuan Gao are with the Engineering Research Center of Software/Hardware Co-design Technology and Application, Ministry of Education, East China Normal University, Shanghai 200062, China (e-mail: leizhang@sei.ecnu.edu.cn).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249 USA.

Josep Domingo-Ferrer is with the Department of Computer Engineering and Mathematics, CYBERCAT-Center for Cybersecurity Research of Catalonia, Universitat Rovira I Virgili, E-43007 Tarragona, Spain.

Mauro Conti is with the Department of Mathematics and HIT Research Center, University of Padua, 35121 Padua, Italy, and also with the Örebro University, 70182 Örebro, Sweden.

Digital Object Identifier 10.1109/TDSC.2025.3599911

1545-5971 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.

(MPC) [21], [22], [23], or by using differential privacy (DP) [4], [24], [25], or by resorting to a mixture of both.

MPC approaches can guarantee data confidentiality and computational correctness throughout the collaborative training process. However, they typically incur high computational and communication overheads due to reliance on resource-intensive cryptographic techniques like homomorphic encryption and secret sharing [26], [27], [28], [29]. For instance, homomorphic encryption-based solutions [26] result in significant ciphertext expansion and roughly triple the communication costs of non-private CDL. Secret sharing-based schemes [28] similarly require multiple communication rounds for aggregation, substantially increasing protocol complexity. Additionally, partial homomorphic encryption methods (such as Paillier encryption) face challenges in directly computing weighted averages since they cannot multiply ciphertexts by plaintext weights [26], [30]. Recently, MPC techniques have also been extended to privacy-preserving inference scenarios to protect both model and data privacy. For example, Wang et al. [31] propose a multi-key dual-server architecture enabling secure outsourced inference. However, inference-phase privacy protection methods are orthogonal to the collaborative training process addressed in our work. These inference techniques may nonetheless serve as complementary mechanisms to enhance overall privacy, providing valuable directions for future integration. In summary, while MPC-based methods provide strong privacy guarantees, their substantial computational complexity and communication overhead limit their practicality in large-scale deployments.

DP approaches achieve privacy by injecting calibrated noise into data or model parameters, offering computational simplicity and minimal overhead. However, DP inherently entails a privacy-utility tradeoff, often leading to accuracy degradation of the learned models. For instance, local DP (LDP) methods perturb model updates before sending them to aggregators, significantly impacting accuracy if overly strong privacy guarantees are enforced [32]. Empirical evaluations show notable accuracy losses on standard datasets such as MNIST and CIFAR-10 when DP is employed [33]. Moreover, sequential composition in DP exacerbates privacy degradation across multiple training rounds [34]. Thus, DP-based techniques, while computationally efficient, struggle to balance strong privacy guarantees against maintaining model accuracy.

Recently, hybrid schemes combining cryptographic primitives and DP have emerged to leverage advantages of both techniques. Nonetheless, these approaches often introduce practical challenges, including reliance on fully trusted third parties, susceptibility to collusion attacks, significant accuracy degradation, and lack of support for dynamic participant scenarios. For example, the hybrid federated learning framework proposed in [35] combines both MPC and DP to provide privacy guarantees to the participants. However, this scheme is not secure against collusion attacks carried out by the (model) aggregator and the participants. At the same time, this scheme also introduces some noise, which affects the accuracy of the learned model. Further, it does not support dynamic participant joining or leaving. Another federated learning framework proposed in [36] is designed to mitigate the risk of collusion attacks among participants and the (model) aggregator. However, a fully trusted third party knowing each participant's private key is required to ensure resistance to collusion attacks and assist in the aggregation of participants' model updates. Besides, the encryption of the entire model generates significant communication overhead. In summary, existing hybrid schemes fail to simultaneously achieve robust privacy protection, accuracy preservation, efficient communication, and support for dynamic participant environments.

B. Our Contributions

To overcome the limitations of high communication cost, lack of support for dynamic participants, inability to perform secure weighted averaging, and accuracy degradation, we design a dual-server privacy-preserving CDL framework. In this architecture, one server functions as the model aggregator (MA), and a dedicated mask aggregator (NA) server is introduced to decouple the masking process from model aggregation. Each participant masks all parameters of its local model update using pseudorandom values derived from a private random seed and a random noise value, submitting the masked update to the MA and the encrypted mask information to the NA. The NA securely reconstructs and aggregates masks from all participants and provides the aggregated mask to the MA, enabling accurate model reconstruction without exposing sensitive information.

A summary of our main contributions is described below:

- The proposed dual-server architecture that decouples masking from aggregation by leveraging a dedicated mask aggregator. This design achieves strong privacy guarantees, even when the MA or NA colludes with up to $t-2$ participants, and significantly reduces the trust requirement for any single server.
- Our protocol enables model aggregation and unmasking to be completed within a single communication round, thereby ensuring round efficiency. Each participant transmits only the masked model and two compact ciphertexts, resulting in minimal communication overhead. Moreover, it supports secure and accurate weighted averaging through homomorphic operations, and guarantees lossless model accuracy without any residual noise or degradation.
- We empirically validate the proposed framework on standard benchmarks and demonstrate its resilience to various inference and inversion attacks, confirming both its privacy guarantees and practical performance.

The rest of this paper is organized as follows. Section II reviews the relevant background information. In Section III, we present our proposed privacy-preserving scheme. In Sections IV and V, we analyze the security and performance of our scheme, respectively. Finally, Section VI concludes this work.

II. Background

A. System Architecture

Fig. 1 illustrates the architecture of our system, which consists of four types of entities, namely: certificate authority (CA), MA, NA and participants. All entities are assumed to be semi-trusted (i.e., honest-but-curious), unless otherwise stated.

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.

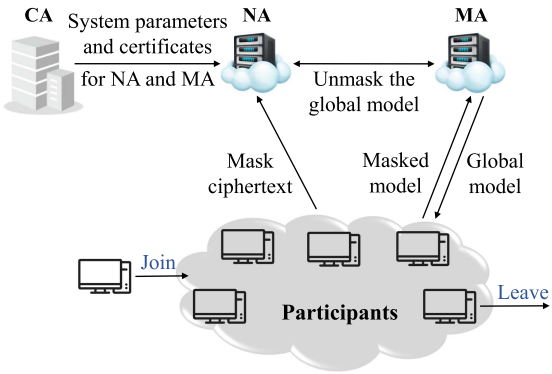


Fig. 1. System architecture.

- *CA:* CA is a trusted third party that is only responsible for generating system parameters and issuing certificates for the servers MA and NA. It does not participate in other operations. It is thus different from the trusted third party that performs model aggregation in other schemes.
- *MA:* This server is responsible for parameter initialization. In addition, MA initiates model aggregation by collecting local model updates sent by participants and aggregating them into an updated global model.
- *Participants:* Each participant (e.g., an organization or an individual) holds its own dataset. A group of participants may decide to launch a CDL. In each training round, each participant trains a model on his/her own local dataset, masks all parameters of the model, sends the masked model update to MA, encrypts the random noise and random seed, and sends the encrypted results to NA. In our approach, we assume the group to be dynamic, that is, a participant may join or leave the group.
- *NA:* This server is responsible for creating encrypted masks for each participant based on the encrypted random noise and random seed from each participant, aggregating all encrypted masks and assisting MA in completing the decryption of the global model.

B. Threat Model and Goals

As previously discussed, common threats faced by CDL models originate from attackers seeking to obtain local model updates or private information of the participants, for example via membership inference attacks or model inversion attacks. The attacker can be a malicious insider (a participant, MA or NA) or an outsider. A summary of threats from different entities is presented below.

- *Threats from participants:* The semi-trusted participants correctly train their local model, but may attempt to obtain the local model updates of other participants. Once they get the model updates, curious participants can launch inference attacks (e.g., membership inference attacks or model inversion attacks) to infer the private information or raw data of target participants.
- *Threats from MA:* The semi-trusted MA legitimately collects the protected local model updates sent by all participants and aggregates them into a global model. However, the MA may try to infer private information from the participants' model updates.
- *Threats from NA:* In our scheme, the encrypted random noise and random seed are sent to NA and are used to generate aggregated encrypted masks. NA may try to obtain the plain masks of a single participant or the plain aggregated masks, with the aim of obtaining the models of some participants.
- *Threats from the outsiders:* An external attacker can obtain local model updates of a specific participant by monitoring the communication between the participant and the MA, in order to launch an inference attack.
- *Threats from (colluding) attackers:* In our approach, *we assume that NA and MA do not collude with each other.* Multiple participants may collude, or one among MA and NA may collude with some curious participants to attack specific participants in order to obtain their local model updates. Then, multiple colluding attackers can launch inference attacks on the model.

Our proposed approach is designed to mitigate the threats discussed above. It has the properties listed in the preceding section, namely: round efficiency, dynamic participation, lossless accuracy and protection against privacy attacks.

C. CKKS Homomorphic Encryption

We leverage the scheme proposed by Cheon, Kim, Kim, and Song (hereafter referred to as CKKS) [37], which is a homomorphic encryption (HE) scheme that is additive and permits computing the product between a ciphertext and a plaintext. A basic CKKS scheme consists of the following algorithms:

- HE.KeyGen(λ): On input a security parameter λ, generate public key pk_{ckks} and private key sk_{ckks}.
- HE.Enc(pk_{ckks}, M): Given a polynomial $M \in \mathcal{R}$, output the ciphertext C.
- HE.Dec(sk_{ckks}, C): For a ciphertext C, output a plaintext polynomial $M' \approx M$.

Being homomorphic, the CKKS scheme also supports the following two algorithms:

- HE.Add(C_1, C_2): For the given ciphertexts C_1 and C_2, output the encryption of $M_1 + M_2$, which is identical to the output of HE.Enc(pk_{ckks}, $M_1 + M_2$).
- HE.Mul(C_1, M_2): For a ciphertext C_1 and a message M_2, output the encryption of $M_1 * M_2$, which is identical to the output of HE.Enc(pk_{ckks}, $M_1 * M_2$).

We use (HE.KeyGen, HE.Enc, HE.Dec, HE.Add, HE.Mul) to define a CKKS scheme used in this paper.

D. Paillier Homomorphic Encryption

Our protocol relies on the additive Paillier homomorphic encryption (PHE) scheme [38]. It consists of five algorithms PHE = (PHE.KeyGen, PHE.Enc, PHE.Dec, PHE.Add, PHE.Mul), as follows:

- PHE.KeyGen(ℓ): Define $L(x) = \frac{(x-1)}{n}$. Choose two large prime numbers p and q s.t. $\gcd(pq, (p-1)(q-1)) = 1$. Calculate $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$. Randomly

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.

generate $g \in \mathbb{Z}_{n^2}^*$ s.t. $\gcd(L(g^\lambda \mod n^2), n) = 1$. Output the public key $pk_{\text{PHE}} = (n, g)$ and private key $sk_{\text{PHE}} = \lambda$.

- PHE.Enc(pk_{PHE}, m): given $m \in \mathbb{Z}n$, sample $r \leftarrow \mathbb{Z}_n^*$ and output

$$c = g^m r^n \mod n^2.$$

- PHE.Dec(sk_{PHE}, c) : for a ciphertext c, calculate

$$m = \frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n.$$

- PHE.Add(c_1, c_2): for ciphertexts c_1, c_2 of plaintexts $m_1, m_2 \in \mathbb{Z}_n$, return

$$c_1 c_2 \mod n^2 = \text{PHE.Enc}(pk_{\text{PHE}}, (m_1 + m_2) \mod n).$$

- PHE.Mul(c, k): for a ciphertext c of m and a scalar $k \in \mathbb{Z}_n$, output

$$c^k \mod n^2 = \text{PHE.Enc}(pk_{\text{PHE}}, km \mod n).$$

E. Pseudorandom Generator

A secure pseudorandom generator $\mathcal{PRG}$ is a function that takes as input a uniformly random seed of fixed length and produces an output that appears to be uniformly random. Specifically, $\mathcal{PRG}$ should be computationally secure, meaning that no efficient algorithm can distinguish its outputs from truly uniformly random elements of the output space, as long as the seed is kept secret. In this paper, we will select a $\mathcal{PRG}$ that not only generates pseudorandom output but also ensures that for a given fixed seed, the output is deterministic and remains the same every time the function is executed.

F. Model Average Algorithm

When multiple participants complete local model training, the federated average algorithm [14] can merge multiple models into a global model. According to the difference of fusion parameters, the algorithm can be divided into model average algorithm [14] and gradient average algorithm [39].

Since both averaging algorithms finally achieve the same function, we adopt the commonly used model average algorithm proposed in [14]. It is summarized in Algorithm 1, which mainly consists of the following steps:

1) MA initializes the system, that is, distributes the global model m_0 and related training parameters to N participants in the system.
2) Each participant P_k updates global model m_{t-1} locally for E epochs with mini-batch size b and learning rate α to obtain the updated local model m_t^k, which P_k sends to MA.
3) MA calculates the weighted average of all models according to the size n_k of each participant's dataset D_k to obtain an updated global model m_t.

Steps (2) and (3) are repeated a total of T iterations to obtain the final global model m_T.

Algorithm 1: Model Average Algorithm.

Require: Participants $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$, dataset $\mathcal{D} = \{D_1, D_2, .., D_N\}$, where $P_k \in \mathcal{P}$ holds his/her own dataset D_k and the size of dataset D_k is n_k; n: sum of participants' dataset sizes; $\mathcal{L}$: DL algorithm; α: learning rate.

Ensure: Global model m_T

function Average:
  Initialize m_0
  for $t = 1$ to T **do**
    for each participant $P_k \in \mathcal{P}$ **do**
      $m_t^k \leftarrow \text{LocalUpdate}(m_{t-1})$
    end for
    $m_t \leftarrow \sum_{k=1}^{N} \frac{n_k}{n} m_t^k$
  end for
end Function
function LocalUpdate$m$:
  for iteration=1 to E **do**
    for each batch b in participant's split **do**
      $m \leftarrow m - \alpha \cdot \bigtriangledown \mathcal{L}(m, b)$
    end for
  end for
  return m
end Function

G. Floating Point Modulo Operation

We present a definition of the modulo operation on floating-point numbers. Let x be a floating-point number and Q be a positive floating-point number. Then, the value of $x \mod Q$ is given by:

$$x - Q * \left\lfloor \frac{x}{Q} \right\rfloor.$$

In the above equation, $\lfloor x/Q \rfloor$ is the largest integer less than or equal to x/Q.

This modulo operation exhibits the following characteristics:

1) *Non-negativity:* The output of x mod Q is always a non-negative value.
2) *Distributivity:* For any floating-point numbers x, y, and Q, the expression $(x + y) \mod Q$ is equivalent to $((x \mod Q) + (y \mod Q)) \mod Q$.

III. Proposed Framework

A. High-Level Description

Our approach comprises the system setup, dual server initialization, and training and aggregation stages.

- *System setup:* The CA generates system parameters and releases them to both MA and NA.
- *Dual server initialization:* Both MA and NA generate their public-private key pairs.
- *Training and aggregation:* This stage can be divided into three sub-stages, namely: parameter initialization, local model training and release, and model aggregation. In *parameter initialization*, MA and NA broadcast basic

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.

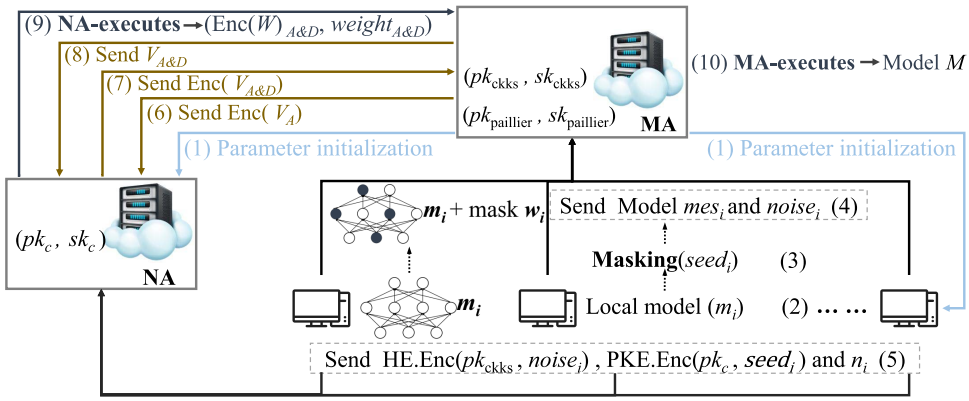


Fig. 2. Training and aggregation stage, which comprises three sub-stages, namely: parameter initialization (step (1)), local model training and release (steps (2) to (5)), and model aggregation (steps (6) to (10)).

parameters to the participants. In *local model training and release*, each participant trains the local model, and sends the masked model update to the MA after masking all model parameters. In addition, each participant sends the encrypted random noise and random seed to NA. Finally, in *model aggregation*, NA creates encrypted masks for each participant based on the encrypted random noise and random seed, aggregates the encrypted masks, and sends the aggregated encrypted masks to MA. Then, MA aggregates the masked model updates of all participants into a masked global model, and uses NA's aggregated encrypted masks to unmask the global model. In this way, MA obtains an updated global model without noise.

B. System Setup

CA is a trusted third-party responsible for generating system parameters. In addition, CA chooses a signature scheme [40] $\text{Sig} = (\text{KeyGen}(\tau), \text{Sig}(msk, M), \text{Ver}(mpk, \sigma))$, where (msk, mpk) is a master private-public key pair, M is the message to be signed, and σ is the signature. The signature scheme is employed by CA to issue digital certificates to MA and NA. Here, MA uses the CKKS homomorphic encryption scheme $\text{HE} = (\text{HE.KeyGen}, \text{HE.Enc}, \text{HE.Dec}, \text{HE.Add}, \text{HE.Mul})$ (see Section II-C) to establish a secure channel with participants. Furthermore, the MA employs the Paillier homomorphic encryption scheme $\text{PHE} = (\text{PHE.KeyGen}, \text{PHE.Enc}, \text{PHE.Dec}, \text{PHE.Add}, \text{PHE.Mul})$ to encrypt selected components of its internal private data. NA uses any public-key encryption scheme $\text{PKE} = (\text{PKE.Gen}(\tau), \text{PKE.Enc}(pk_c, M), \text{PKE.Dec}(sk_c, C))$ to establish a secure channel with participants, where (pk_c, sk_c) is a public-private key pair, M is the plaintext and the C is the ciphertext.

C. Dual Server Initialization

Both MA and NA generate their respective public-private key pairs. Specifically, MA uses the CKKS public key encryption algorithm [37] to generate a public-private key pair $(pk_{\text{ckks}}, sk_{\text{ckks}})$, and uses the Paillier public key encryption algorithm [38] to generate a public-private key pair $(pk_{\text{PHE}}, sk_{\text{PHE}})$. NA uses a secure public key encryption algorithm [41] to generate a public-private key pair (pk_c, sk_c). Each participant needs to obtain the public keys of MA and NA before joining the training. In addition, NA also obtains the public key of MA.

D. Training and Aggregation

As shown in Fig. 2, this stage can be divided into three sub-stages, namely: parameter initialization, local model training and release, and model aggregation. In ***parameter initialization***, the initialization of the participants should be completed before each round of training. MA sends the initial neural network model to the participants, and specifies the corresponding loss function, optimization function, learning rate, a positive integer Q and other necessary information for the training. In addition, MA sends the initial model to NA.

In ***local model training and release***, all participants execute the `Participants-train` function in Algorithm 2 (see lines 1 to 10), where $m_i \leftarrow \mathcal{L}(D_i)$ denotes that participant P_i trains his/her local model m_i on local dataset D_i over many epochs, and the function $\texttt{Masking}(m_i)$ is used for each participant to generate a masked model mes_i and is defined as follows:

- Suppose the total number of parameters of P_i's model is S_a. Participant P_i selects a random seed $seed_i$ as the seed of $\mathcal{PRG}$ to generate pseudorandom numbers $R_i = [r_1^i, r_2^i, \ldots, r_{S_a}^i]$.
- Assume the range of model parameters is $[-max, max]$. P_i generates a random floating-point number $noise_i$, which we call *random noise*. For each parameter par_j^i $(1 \leqslant j \leqslant S_a)$ in model m_i, P_i calculates $(r_j^i * noise_i + par_j^i) \bmod Q$ to obtain the masked model mes_i, where the operation $\bmod Q$ (whose remainder range is $[0, Q)$) is defined in Section II-G.

At this point, the above `Masking` function responsible for masking the model has been executed, and the participant P_i has obtained the masked model mes_i. Then, participant P_i encrypts the random noise $noise_i$ by computing $\text{HE.Enc}(pk_{\text{ckks}}, noise_i)$

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2: Training and Aggregation.

Require: Participants $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$, dataset $\mathcal{D} = \{D_1, D_2, .., D_N\}$, where $P_i \in \mathcal{P}$ holds his/her own dataset D_i and the size of dataset D_i is n_i; learning algorithm $\mathcal{L}$; CKKS key pair $(pk_{\text{ckks}}, sk_{\text{ckks}})$ for MA; Paillier key pair $(pk_{\text{PHE}}, sk_{\text{PHE}})$ for MA; PKE key pair (pk_c, sk_c) for NA; secure pseudorandom generator $\mathcal{PRG}$; threshold t.

Ensure: Global model $\mathcal{M}$.

1: **function** Participants-train:
2: **for** $P_i \in \mathcal{P}$ **do**
3: $m_i \leftarrow \mathcal{L}(D_i)$ ▷Train local model
4: $(mes_i, noise_i, seed_i) \leftarrow$ `Masking`(m_i)
5: **send** mes_i to MA
6: Enc$(noise_i) \leftarrow$ HE.Enc$(pk_{\text{ckks}}, noise_i)$
7: Enc$(seed_i) \leftarrow$ PKE.Enc$(pk_c, seed_i)$
8: **send** (Enc$(noise_i)$, Enc$(seed_i)$, n_i) to NA
9: **end for**
10: **end Function**
11: **function** MA-Prepare Vector:
12: Receive $\{mes_i\}_{1 \le i \le N}$
13: Generate V_A indicating received models
14: Enc$(V_A) \leftarrow \{$PHE.Enc$(pk_{\text{PHE}}, v_i \in V_A)\}$
15: **send** Enc(V_A) to NA
16: **end Function**
17: **function** NA-Prepare VectorEnc(V_A):
18: Receive {Enc $(noise_i)$, Enc$(seed_i), n_i\}_{1 \le i \le N}$
19: Generate vector V_D
20: Enc$(V_{A\&D}) \leftarrow$ Enc$(V_A) \cap V_D$
21: **send** Enc$(V_{A\&D})$ to MA
22: **end Function**
23: **function** MA-executesEnc$(V_{A\&D})$:
24: $V_{A\&D} \leftarrow$ PHE.Dec$(sk_{\text{PHE}}$,Enc $(v_{A\&D}) \in$ Enc$(V_{A\&D}))$
25: Enc$(W)_{A\&D}, weight_{A\&D} \leftarrow$ `NA-Executes`$(V_{A\&D})$
26: **if** Enc$(W)_{A\&D} = \emptyset$ **then**
27: Abort aggregation round
28: **end if**
29: $W \leftarrow$ HE.Dec$(sk_{\text{ckks}}$,Enc$(W)_{A\&D})$
30: $\hat{M} \leftarrow$ WeightedAverage$(\{mes_i\}, weight_{A\&D})$
31: $\mathcal{M} \leftarrow (\hat{M} - W)$
32: return $\mathcal{M}$
33: **end Function**
34: **function** NA-executes$V_{A\&D}$:
35: **if** $V_{A\&D}$ contains fewer than t ones, for a security threshold t, or there has been a previous call to this function in the current aggregation round **then**
36: return $\emptyset$ to MA
37: **else**
38: $n \leftarrow \sum n_i$ where $v_i \in V_{A\&D} = 1$
39: $weight_{A\&D} \leftarrow (\frac{n_1}{n}, \ldots, \frac{n_N}{n})$
40: **for** each P_i where $v_i \in V_{A\&D} = 1$ **do**
41: Enc $(noise_i)_w \leftarrow$ HE.Mul(Enc$(noise_i), \frac{n_i}{n})$
42: $seed_i \leftarrow$ PKE.Dec$(sk_c,$ Enc$(seed_i))$
43: $R_i = [r_1^i, r_2^i, \ldots, r_{S_a}^i] = \mathcal{PRG}(seed_i)$
44: Enc $(w_i)_w \leftarrow$ HE.Mul(Enc$(noise_i)_w, R_i)$
45: **end for**
46: Enc $(W)_{A\&D} \leftarrow \sum$ Enc$(w_i)_w$
47: **send** (Enc$(W)_{A\&D}$, $weight_{A\&D}$) to MA
48: **end if**
49: **end Function**

to obtain the *encrypted random noise* Enc$(noise_i)$. At the same time, the participant also needs to encrypt the random seed $seed_i$ by computing PKE.Enc$(pk_c, seed_i)$ to obtain the encrypted random seed Enc$(seed_i)$. Finally, each participant sends the local masked model update mes_i to MA. He/she also sends the encrypted random noise Enc$(noise_i)$, the encrypted random seed Enc$(seed_i)$, and the local dataset size n_i to NA.

In ***model aggregation***, either MA receives model updates $\{mes_i\}_{1 \le i \le N}$ from all participants, or the maximum waiting time of MA is reached, MA executes the `MA-Prepare Vector` function (see lines 11 to 16). MA needs to request from NA the mask added by each participant in the model update. MA will generate a vector V_A to indicate which model updates of participants it has received, and it can be set as $V_A = (1, 1, 0, 1, 0, \ldots, 1)$ s.t. $|V_A| = N$, where N is the total number of participants. MA may not receive some participants' local model updates due to participants dropping out, network delay, etc. If MA has received the local model updates submitted by a participant, the corresponding location in the V_A is 1, otherwise it is 0. Therefore, the impact of some participants leaving the system is small, which enables dynamic participation. Then, MA encrypts the Boolean vector V_A element-wise under Paillier, obtaining Enc$(V_A) =$ [PHE.Enc$(pk_{\text{PHE}}, v_i), \ldots,$ PHE.Enc$(pk_{\text{PHE}}, v_N)]$, and sends vector Enc(V_A) to NA to request the aggregated encrypted masks.

When NA receives a request from MA, it will run the `NA-Prepare Vector` function (see lines 17 to 22). NA generates a vector V_D to describe which participants' encrypted masks matrix it has received, in the same way as MA generates vector V_A. Then, it will generate vector Enc $(V_{A\&D})$ based on the intersection of vectors Enc(V_A) and V_D. Specifically, for the i th element v_i in Enc(V_A) and element d_i in V_D, NA performs PHE.Mul(PHE.Enc$(v_i), d_i)$ and finally obtains Enc$(V_{A\&D}) = \{$PHE.Mul(PHE.Enc$(v_i), d_i\}_{1 \le i \le N}$. Then NA sends Enc$(V_{A\&D})$ to MA for decryption and obtains $V_{A\&D}$.

When NA obtains the decrypted $V_{A\&D}$ from MA, it starts executing the `NA-Executes` function (see lines 34 to 49). The participants included in vector $V_{A\&D}$ (for which the corresponding location in the vector is 1) are those who have successfully sent the local masked model updates to MA and encrypted random noise to NA. If there are fewer than t such successful participants (where t is a security parameter related to collusions, see Section IV-C), NA returns nothing to MA. Otherwise, NA calculates the sum of the dataset sizes of the participants included in vector $V_{A\&D}$ and denotes it by n. NA needs to calculate the weight of each participant based on his/her dataset size, and sets $weight_{A\&D} = (\frac{n_1}{n}, \ldots, \frac{n_N}{n})$, where $n_i > 0$ if participant P_i's component in $V_{A\&D}$ is 1 and $n_i = 0$ otherwise. Now NA needs to aggregate the encrypted masks of the participants included in $V_{A\&D}$, which is done as follows:

- NA generates a ciphertext Enc(0.0) by computing HE.Enc$(pk_{\text{ckks}}, 0.0)$. Then, it creates a new model W with the same network structure as the model sent by MA in the

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.

parameter initialization stage, and sets all the elements in W to be Enc(0.0).

- NA multiplies the encrypted random noise Enc $(noise_i)$ of each participant P_i included in vector $V_{A\&D}$ times the corresponding weight $\frac{n_i}{n}$ by computing HE.Mul(Enc$(noise_i)$, $\frac{n_i}{n}$). In this way, NA obtains the *weighted* encrypted random noise Enc$(noise_i)_w$.
- NA decrypts participant P_i's encrypted random seed Enc $(seed_i)$ with PKE.Dec$(sk_c,$ Enc$(seed_i))$ to obtain random seed $seed_i$. According to $seed_i$, NA generates P_i's pseudorandom numbers $R_i = [r_1^i, r_2^i, \ldots, r_{S_a}^i]$.
- NA obtains participant P_i's weighted encrypted masks Enc $(w_i)_w$ by calculating HE.Mul(Enc$(noise_i)_w, r_j^i)_{1\leqslant j\leqslant S_a}$.
- NA adds each element Enc $(noise_j^i)_w$ $(1 \le j \le S_a)$ in Enc$(w_i)_w$ to the encrypted parameter e_j in model W by computing HE.Add$(e_j,$ Enc$(noise_j^i)_w)$. At this point, the weighted encrypted masks of all participants included in vector $V_{A\&D}$ have been aggregated into Enc$(W)_{A\&D}$.

Finally, NA sends (Enc$(W)_{A\&D}$, $weight_{A\&D}$) to MA.

Upon receiving a response from NA, MA performs the remaining procedures in the `MA-Executes` function (see lines 23 to 33). MA decrypts each element e_j $(1 \le j \le S_a)$ in Enc$(W)_{A\&D}$ by computing HE.Dec(sk_{ckks}, e_j), and performs the modulo operation on each decrypted element by computing $e_j \bmod Q$ $(1 \le j \le S_a)$. At this point, MA obtains a set of weighted aggregated masks W, which can be represented as:

$$W = \left\{ \left(\sum_{i=1}^{N} r_j^i * \frac{n_i}{n} * noise_i \right) \bmod Q \right\}_{1\leqslant j\leqslant S_a}.$$

Then, MA uses the model average algorithm mentioned in Section II-F to aggregate the local masked model updates of the participants with weight $\frac{n_i}{n} \neq 0$ in $weight_{A\&D}$. In this way, MA obtains a weighted masked aggregated model $\hat{M} = \sum_{i=1}^{N} \frac{n_i}{n} mes_i$. After that, MA performs the modulo operation on each element $\hat{m_j}$ in $\hat{M}$ by calculating $\hat{m_j} \bmod Q$ $(1 \le j \le S_a)$. Specifically, $\hat{M}$ can be represented as:

$$\begin{aligned}\hat{M} &= \left\{ \left[\sum_{i=1}^{N} \frac{n_i}{n} * ((r_j^i * noise_i + par_j^i) \bmod Q) \right] \right. \\ &\quad \left. \times \bmod Q \right\}_{1\leqslant j\leqslant S_a} \\ &= \left\{ \left[\sum_{i=1}^{N} \left(\frac{n_i}{n} * r_j^i * noise_i + \frac{n_i}{n} * par_j^i \right) \right] \bmod Q \right\}_{1\leqslant j\leqslant S_a}.\end{aligned}$$

Finally, MA subtracts the aggregated masks W from the aggregated model $\hat{M}$, and subsequently obtains an updated global model $\mathcal{M}$.

We note here that the case $|m_j| > max$ needs a correction by MA. As an example, take $Q = 100$ and $max = 1.0$. If $\hat{m_j} \in \hat{M}$ is 0.3 and $w_j \in W$ is 99.7, then $m_j \in M = \hat{m_j} - w_j = 0.3 - 99.7 = -99.4$. Obviously this m_j calculated by MA is wrong. Therefore, for all $m_j \in M$, MA needs to correct $|m_j| > max$ as:

$$m_j = \begin{cases} m_j + Q, & \text{if } m_j < 0; \\ m_j - Q, & \text{if } m_j > 0. \end{cases}$$

At this point, MA recovers the global model $\mathcal{M}$ without noise.

E. Noise-Bounded Precision in CKKS

In our proposed scheme, each participant masks their model parameters using random noise values. To ensure accurate and error-free recovery of the aggregated model parameters after homomorphic operations, the precision of these random noise values must be carefully controlled. Typically, our scheme requires maintaining up to eight decimal digits of precision for each parameter, and hence, the random noise values must also retain this precision.

The CKKS encryption scheme inherently introduces small computational errors due to encoding, encryption, and homomorphic operations. To guarantee that these errors do not affect the final aggregated results, we leverage Lemma 1 from the original CKKS paper [37], which provides a formal bound on the noise introduced by the CKKS encryption.

Specifically, Lemma 1 states the following sufficient condition for exact integer recovery:

Lemma 1: [CKKS Encoding and Encryption Lemma [37]] Let $\mathbf{z} \in \mathbb{Z}[i]^{N/2}$ and $m = \text{Ecd}(z; \Delta)$ be its plaintext encoding with scale Δ. If ciphertext $c \leftarrow \text{Enc}_{pk}(m)$ is encrypted under scale Δ, then after decryption and decoding we have exactly:

$$\text{Dcd}(\text{Dec}_{sk}(c)) = \mathbf{z},$$

provided that the scaling factor satisfies:

$$\Delta > N + 2B_{\text{clean}},$$

where

$$B_{\text{clean}} = 8\sqrt{2}\sigma N + 6\sigma\sqrt{N} + 16\sigma\sqrt{hN}.$$

We extend this lemma to fixed-point real numbers. Specifically, for a real-valued input $x \in \mathbb{R}$ which is with n-decimal-digit precision, that means the inequality $\Delta^{-1} \cdot (N/2 + B_{\text{clean}}) < 1/(2 \times 10^n)$ must hold. Additionally, our scenario involves aggregation of up to k (e.g. $k = 10^3$) ciphertexts, which results in the encryption noise term B_{clean} being increasing linearly with the homomorphic additions. Therefore, the inequality becomes $\Delta^{-1} \cdot (N/2 + k \cdot B_{\text{clean}}) < 1/(2 \times 10^n)$. Then we have the scaling factor satisfying

$$\boxed{\Delta > 10^n \cdot (N + 2 \cdot k \cdot B_{\text{clean}}).}$$

This ensures n-decimal-digit precision of the output result after the aggregation of up to k ciphertexts.

With typical CKKS parameters ($N = 8192, \sigma = 3.2,$ and $h = 64$), and set $n = 8, k = 10^3$, we estimate $B_{\text{clean}} \approx 3.35 \times 10^5$ and the required scaling factor satisfies:

$$\Delta > 10^8 \cdot (8192 + 2 \times 10^3 \times 3.35 \times 10^5) \approx 2^{55.9}.$$

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.

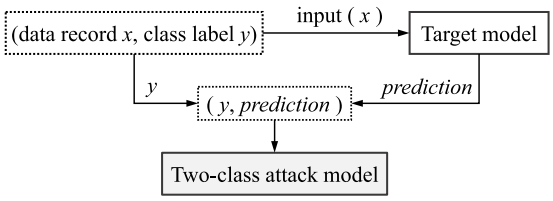


Fig. 3. Membership inference attacks in the black-box setting.

To ensure a conservative margin of error, we select

$$\Delta = 2^{56},$$

which gives a safe margin for exact recovery of the encoded integers after aggregation, thus reliably preserving full eight-digit decimal precision throughout the encrypted computations.

IV. Security and Privacy Evaluation

The proposed privacy-preserving scheme is meant to resist attacks by masking all parameters of the model. In this section, we will analyze the privacy and security of the scheme.

The privacy analysis aims to verify whether the masked model may still disclose the participants' local training data. We use membership inference and model inversion attacks against the model protected by our scheme to evaluate privacy (see Sections IV-A and IV-B). Our attacks were performed on a PC with Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz, 64G RAM and NVIDIA TITAN V graphics card.

The security analysis first considers the collusion attacks that the system may actually encounter and the corresponding countermeasures (see Section IV-C), and then further analyzes the security of the scheme theoretically (see Section IV-D).

A. Resistance to Membership Inference Attacks

Given a data record and black-box access to a model, membership inference attacks [19] attempt to determine whether the record was in the model's training set. Take as an example training data consisting of patient disease information. If an attacker can infer that a specific person's record was in the training set, he can guess that the person has a certain disease. This seriously violates personal privacy.

We mount membership inference attacks as described in [19] and depicted in Fig. 3. Assume that the training set of the target model is known. Take a piece of data (x, y) as the input of the target model to which the attacker has black-box access. Let the model output $prediction$. According to whether this piece of data is in the training set of the target model, $(y, prediction, in)$ or $(y, prediction, out)$ can be constructed. Use $(y, prediction)$ as the training set and in or out as the label to train a two-class model. That is, the attack model is a two-class model, which takes the output of the target model as input to determine whether a piece of data is in the training set of the target model. The rationale of this attack method is that the performance of the target model on its training set and on the data encountered for the first time is often different. Based on this, it can be inferred whether a piece of data was in the training set.

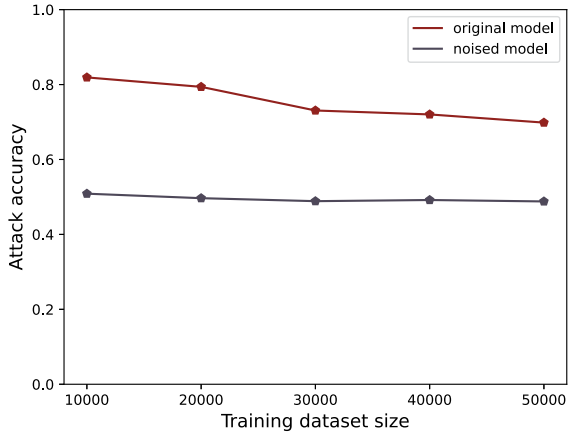


Fig. 4. Accuracy of the membership inference attacks against ResNet-18 trained on CIFAR-100, as a function of the training data set size.

In the real world, the attacker cannot directly obtain the training set of the target model. In order to train the attack model, the authors of [19] constructed multiple shadow models and used both their training and test datasets to train the attack model. Constructing shadow models is not the focus of our experiments. We directly use the training set and test set of the target model as the training set for the attack model, so as to maximize the attacker's background knowledge. If the attacker fails in this setting, then our system is guaranteed to be resistant in the setting proposed by [19].

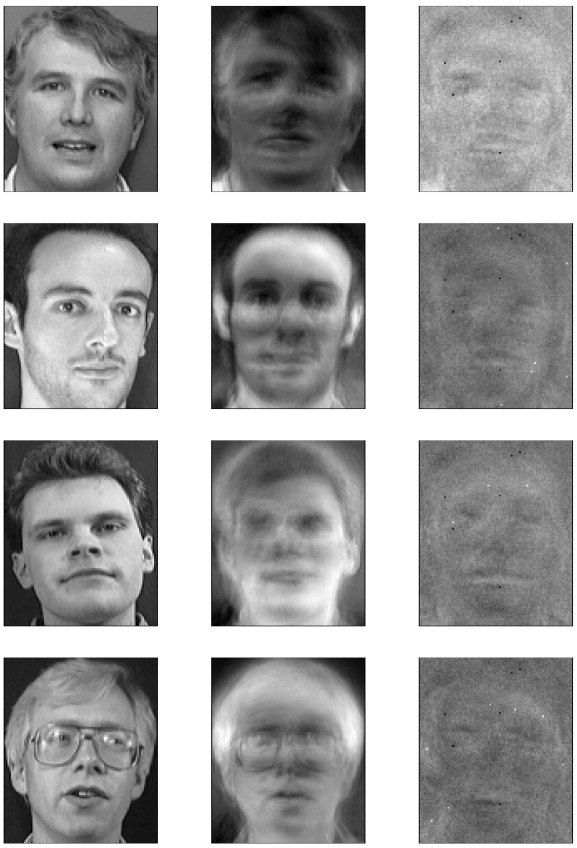
We chose PyTorch [42] as an experiment platform, and we used the convolutional neural network architecture ResNet-18 [43] to train the classification model on CIFAR100 [44]. This dataset is composed of 100 classes of 32×32 color images, and each class contains 600 images, divided into 500 training sets and 100 test sets. In addition, we set the learning rate to 0.01 and the maximum training epochs to 40. We selected 10000, 20000, 30000, 40000, and 50000 CIFAR100 training sets to train the models respectively, and we masked all models. Fig. 4 shows that the attack on all original models was effective but, as the number of model training sets increased, the attack accuracy decreased. The accuracy of attacks on masked models was always at the baseline 0.5, which proves that our model protection method can resist membership inference attacks.

B. Resistance to Model Inversion Attacks

Model inversion attacks mainly refer to the attacker extracting information related to training data from the model prediction results [20]. For example, a neural network model for face recognition provides a face recognition classification service in the form of API. For each face picture, the predicted name and corresponding confidence level can be output. The attacker can reconstruct the facial image of a person from the person's name included in the training set or a label output by the model. This process indirectly filches the private picture information about Alice in the training set.

We use the model inversion attack proposed in [20] to reconstruct a face from a name included in the model training set.

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.



(a) training set (b) original model (c) masked model

Fig. 5. Images of the training set (left), images reconstructed based on the original model (middle), and images reconstructed based on the masked model (right). The attacker is only given the victim's name and access to the facial recognition system, which returns a confidence score for a class.

We first introduce the background knowledge and main idea of the attack. The neural network backpropagation algorithm calculates the loss value under the conditions of given training data x, labels y, and initialization model, and then backpropagates the parameter w of the optimized model to get the correct output Y corresponding to x. The attack calculates the loss value given the trained model, output Y and randomly initialized input data x, and reversely optimizes the input data x corresponding to Y.

We chose PyTorch [42] as the experimental platform, and we used a two-layer neural network to train the classification model on the AT&T Laboratories Cambridge database of faces[1], which consists of 40 classes of 92×112 black-and-white images. Each class contains 10 images, divided into 7 training sets and 3 test sets. In addition, we set the learning rate to 0.0003 and the maximum training epochs to 400.

We implemented the reconstruction attack in [20]. As shown in Fig. 5, images in the first column are those in the model training set. After obtaining the victim's name and black-box access to the facial recognition system, the images in the second column were reconstructed. Then, we masked all parameters of the model and performed the same attack. As shown in the third column of images, the reconstructed faces hardly reveal anything on the victim. This confirms that our privacy protection scheme can effectively resist the attempted model inversion.

[1]http://cam-orl.co.UK/facedatabase.html

C. Resistance to Collusion Attacks

We consider how to resist collusion attacks by the one of the semi-trusted MA / NA and curious participants. Their main purpose is to capture the original data of some honest participants by obtaining their model updates. Here, we will prove the security of our scheme in an honest-but-curious setting.

Theorem 1: Any collusion between one of the semi-trusted servers and no more than $t-2$ curious participants fail to obtain the private information of other honest participants except the aggregated model updates.

Proof: The participants mask all parameters of the model updates to protect model privacy. If MA wants to obtain a certain participant's model updates, it needs to obtain the masks of this participant from NA. MA has three ways to obtain the masks of a specific participant: (a) to deliberately send a specific V_A to NA, (b) to collude with curious participants, or (c) to request the masks from NA twice and subtract the two values to get the difference which is the masks of the specific participant.

For attack method (a), MA can set vector V_A as $V_A = (0,\ldots,0,1,0,\ldots,0)$ where the single 1 is placed at position k. If NA accepted this $V(A)$, it would return the masks of participant P_k to MA. To prevent this attack, NA checks in function NA-EXECUTES that $V_A \cap V_D$ contains at least t 1s, which means that $p \geq t$ participants take part in the aggregation.

For attack method (b), MA has to collude with at least $p-1$ curious participants to get the masks of a specific participant, which is difficult to achieve. Finally, attack method (c) can be thwarted by allowing MA only one masks request during one round of aggregation, as implemented in function NA-EXECUTES.

It can also happen that NA wants to obtain the plain masks of some participants to unmask their masked model updates. This would give NA the noise-free model updates of these participants. To that end, NA needs to decrypt the encrypted random noise, which in turn requires breaking the CKKS homomorphic encryption scheme. Since this scheme is secure, this attack is not practical.

NA can also collude with curious participants to: (1) obtain the plaintext masks of a specific participant's model, or (2) decrypt the encrypted aggregated random noise. To follow approach (1), NA needs to collude with at least $p-1$ curious participants, which is difficult to achieve. To follow approach (2), NA needs to collude with all the participants, which is even more difficult. This proves that our scheme can resist collusion attacks of up to $t-2$ participants.

D. Security Analysis

In our security argument, we will analyze whether an attacker can launch a successful attack on our scheme, that is, whether s/he can obtain an unmasked model by deciphering a masked model. We will prove that the attacker cannot distinguish the masked model from a randomly generated model, and hence

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.

s/he cannot launch an effective attack based on the masked model update.

Theorem 2: Fix N, Q, $max > 0$, S_a, $noise_i \in (0, max]$, $\{\{par_j^i\}_{1\leqslant j\leqslant S_a}\}_{1\leqslant i\leqslant N}$, $\{\{r_j^i\}_{1\leqslant j\leqslant S_a}\}_{1\leqslant i\leqslant N}$ where $par_j^i \in [-max, max]$, $r_j^i \in \mathbb{Z}$, $Q \gg max$. Then,

$$\{(r_j^i * noise_i + par_j^i) \bmod Q\}_{1\leqslant j\leqslant S_a}$$
$$\equiv$$
$$\{y_j \xleftarrow{\$} \mathbb{Z}_R \bmod Q\}_{1\leqslant j\leqslant S_a},$$

where $\equiv$ denotes that the distributions are identical.

Proof: To prove that $(r_j^i * noise_i + par_j^i) \bmod Q$ is uniformly random on [0, Q), we need to show that for any subinterval [a, b) of [0, Q), the probability that $(r_j^i * noise_i + par_j^i) \bmod Q$ falls within [a, b) is equal to the length of the subinterval divided by Q.

Let $\mathcal{A}$ be the event that $(r_j^i * noise_i + par_j^i) \bmod Q$ falls in the interval $[a, b)$. Then, we can express this event as:

$$\mathcal{A} = \{(r_j^i * noise_i + par_j^i) \bmod Q \in [a, b)\}$$
$$= \left\{(r_j^i * noise_i + par_j^i) \in \bigcup_k [k * Q + a, k * Q + b)\right\}.$$

In the above expression, k takes all possible values such that $k * Q + a \leqslant r_j^i * noise_i + par_j^i < k * Q + b$. When $a = 0$, $b = Q$, $\bigcup_k [k * Q + a, k * Q + b)$ represents the complete value range of $r_j^i * noise_i + par_j^i$. That is, the value range of $r_j^i * noise_i + par_j^i$ can be regarded as $|\{k\}|$ subintervals of length Q.

We note that since $noise_i$ is a uniformly random decimal on [0, ς), where ς is a large floating point number, the value of $r_j^i * noise_i + par_j^i$ is uniformly random on the interval [par_j^i, $r_j^i * noise_i + par_j^i$). Therefore, we can express the probability of the event $\mathcal{A}$ as:

$$\Pr[\mathcal{A}] = \frac{|\{k\}| * ((k * Q + b) - (k * Q + a))}{r_j^i * noise_i + par_j^i - par_j^i}$$
$$= \frac{|\{k\}| * (b - a)}{|\{k\}| * Q} = \frac{(b - a)}{Q}.$$

Hence, $(r_j^i * noise_i + par_j^i) \bmod Q$ is uniformly random on [0, Q). Thus an attacker cannot distinguish between a model protected by our scheme and a randomly generated model whose parameter values are distributed on [0,Q].

V. Performance Evaluation

In this section we systematically analyze the properties of our scheme other than privacy, and we compare with previous proposals.

A. Satisfied Properties

We show that our scheme achieves round efficiency and supports dynamic participation, while enabling efficient weighted average computation without loss of accuracy.

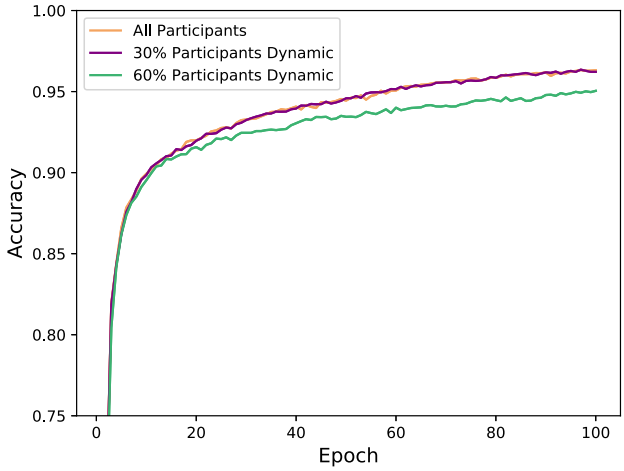


Fig. 6. Impact of dynamic participants on global model accuracy.

Round efficiency: Our scheme achieves round efficiency primarily by minimizing encryption operations and reducing communication complexity. Specifically, instead of encrypting the entire locally trained model, each participant applies lightweight pseudorandom masking techniques to obscure model parameters, encrypting only the small random noise and seed values required for mask reconstruction. Consequently, our architecture enables secure model aggregation and unmasking within a single communication round between participants and aggregators. Because each participant's update comprises merely the masked model and two compact ciphertexts, message expansion is minimal, substantially lowering communication overhead. Compared to conventional MPC-based protocols that require multiple interactive communication rounds among participants or between participants and servers, our design attains comparable security guarantees with significantly improved round efficiency and reduced bandwidth usage.

Dynamic participation: In practical CDL deployments, participants may join or leave the system at any time due to resource constraints, connectivity issues, or operational requirements. Our scheme is explicitly designed to accommodate such dynamic participation. For newly joining participants, the initialization phase described in Section III-D ensures seamless onboarding, requiring only the transmission of a small number of parameters from the MA and NA. Similarly, participants who leave the system have minimal impact on the overall performance. The protocol maintains correct operation as long as the number of active participants remains above a security threshold t ($t \leq N$).

We empirically evaluate the robustness of our scheme to dynamic participant behavior using the MNIST dataset, with an average of 60 online participants per epoch over 100 training epochs. In our experiments, we simulate environments where up to 30% and 60% of participants may join or leave the task at each epoch. As illustrated in Fig. 6, the global model accuracy remains virtually unaffected when 30% of participants are dynamic, closely matching the performance of a fully stable system. Even in the more challenging scenario with 60% dynamic

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.

TABLE I
A COMPARATIVE SUMMARY OF CDL SCHEMES

Scheme	Dynamic	Round Efficient	Lossless	Universal	Model weighted average
Shokri et al. [45]	Join / Leave	✓	✗	✗	✓
Bonawitz et al. [28]	Leave	✗	✓	✓	✓
Xu et al. [46]	Leave	✗	✓	✓	✓
Phong et al. [26]	Join / Leave	✗	✓	–	✓
Zhang et al. [30]	–	✓	✗	–	✓
Our work	Join / Leave	✓	✓	✓	✓

participation, only a moderate decrease in accuracy is observed, and the model still converges reliably. These results demonstrate that our approach supports resilient and accurate learning under dynamic and realistic participation patterns.

Efficient weighted average computation: Although our design incorporates homomorphic encryption and masking techniques to ensure enhanced privacy protection, it remains efficient in performing weighted averaging of model parameters. This efficiency is primarily attributed to the additive homomorphic properties of the employed encryption schemes (CKKS and Paillier), which allow NA to securely and directly compute weighted sums over encrypted masks. Consequently, MA only needs to execute a single decryption operation to recover the aggregated mask. Moreover, all weight-related computations are performed either in plaintext or through lightweight homomorphic operations, thus preserving computational efficiency.

Lossless accuracy: In CDL, it is essential to protect participant model privacy while simultaneously preserving the accuracy and utility of the aggregated global model. In our scheme, each participant masks all parameters of their local model updates prior to submission, ensuring privacy without compromising data integrity. Our design employs two non-colluding servers to securely perform weighted aggregation and subsequent unmasking operations. Consequently, the final global model contains no added noise and achieves lossless accuracy.

B. Comparison

A comparative summary of our proposed scheme and several typical privacy-preserving CDL schemes [26], [28], [30], [45], [46] is presented in Table I. These state-of-the-art schemes have been selected for the following reasons. The first systematic Paillier homomorphic encryption-based privacy-preserving deep learning scheme was proposed in [26]. The scheme in [30] proposes a batch encryption algorithm based on Paillier homomorphic encryption to preserve the privacy of federated learning, which almost minimizes the computation and communication overhead. As a secure aggregation scheme for privacy-preserving machine learning, the scheme in [28] has the highest comprehensive efficiency that ensures full data privacy. The scheme in [46] extends [28], and it can additionally verify the correctness of the aggregation results. The scheme in [45] is the first to propose collaborative model training by sharing partial parameters.

One can observe that our scheme outperforms the other five schemes. The scheme of [45] allows selectively sharing model parameters. However, in order to protect the privacy of the shared parameters, this scheme introduces noise, which affects the accuracy of the global model. In addition, the parameter selection algorithm for selectively sharing parameters in this scheme needs to be carefully configured to stay applicable to any CDL scenario. The security aggregation protocols proposed in [28], [46] do not support the joining of new participants, and they require four rounds of interactions to aggregate model updates from one iteration of training. The privacy protection scheme (based on Paillier homomorphic encryption) in [26] introduces significant computation and communication overhead. The scheme proposed in [30] reduces the computation and communication overhead of Paillier homomorphic encryption through batch encryption and gradient clipping. However, it leads to a certain loss of accuracy, and the overhead is still unsatisfactory.

C. Computation and Communication Overheads

We evaluate the computation and communication overheads for both participants and servers. All experiments are conducted using PyTorch on a workstation equipped with an 11th Gen Intel Core i7-11700K (3.60GHz), 16GB RAM, and an NVIDIA Quadro P5000 graphics card. Overheads are measured using two representative models: ResNet-18 (see Section IV-A) and a two-layer neural network (see Section IV-B).

1) Computation Overhead: In our architecture, each participant is responsible solely for encrypting a random noise value and a random seed, while NA performs homomorphic weighted aggregation of encrypted masks, and MA carries out the final decryption of the aggregated masks. Table II presents the computational costs for participants, NA, and MA. The additional computational burden on participants to secure their model updates is minimal. The time cost for weighted aggregation on the NA is linearly dependent on the number of participants.

2) Communication Overhead: We systematically analyze the communication overhead introduced by our scheme from three perspectives: protocol structure, message size, and empirical performance validation, complemented by comparisons to baseline and privacy-preserving schemes.

a) Protocol-level communication pattern. Although Fig. 2 depicts ten steps for detailed illustration, only two steps (Steps 4 and 5) involve communication from participants to the server, both occurring within a single logical round. Specifically, each participant uploads a masked model update accompanied by an encrypted random noise scalar, an encrypted seed, and the local dataset size. All other steps involve either local participant

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.

TABLE II
CALCULATION AND COMMUNICATION OVERHEAD BASED ON CKKS ENCRYPTION SCHEME

	Network	Number of Parameters	Plaintext	Communication Overhead	Encryption	Weighted Aggregation	Decryption
CIFAR100	ResNet18	33161024	42.86MB	42.86MB + 116KB	0.01ms	$N * 0.03$ms	7635.55ms
AT&T face Database	2-layer NN	10593320	40.41MB	40.41MB + 116KB	0.01ms	$N * 0.01$ms	2439.18ms

TABLE III
COMMUNICATION OVERHEAD COMPARISON WITH EXISTING WORKS

Scheme	Technique	Rounds	Per-Client Overhead	Model Sensitivity	Comments
FL (no privacy) [13]	–	1	model size	Low	No privacy protection
Bonawitz et al. [28]	Secret Sharing	4	$\approx$ 3–4$\times$model size	Low	Requires multiple interaction rounds
Zhang et al. [30]	Paillier	2	$\approx$ 2–3$\times$model size	High	Batch encryption, quantization
Wang et al. [8]	Paillier	1	≈ 1.75–$2.14\times$model size	Low	Selective bit encryption
Ours	CKKS + Paillier	1	model size + 116KB	Low	Lightweight masking and minimal encryption

computation or server-side internal communication, which imposes no additional communication burden on participants.

b) Message size analysis. Unlike schemes that require encrypting the complete model update, our scheme demands only two small ciphertexts in addition to the masked model update from each participant. The encrypted noise scalar is approximately 115KB using CKKS encryption, while the encrypted seed is roughly 256 bytes via public-key encryption. Therefore, the total additional payload per participant per round is about 116KB. Compared to typical model sizes, such as 42.86MB for ResNet-18, this represents a negligible overhead of less than 0.27% and remains constant regardless of the underlying model complexity.

c) Empirical validation and comparison. Table II provides detailed communication overhead results for two representative deep learning models: ResNet-18 trained on CIFAR-100 and a two-layer neural network on the AT&T face database. The additional communication cost from our privacy-preserving measures remains consistently low at approximately 116KB per participant, independent of the model size. Practically, this corresponds to an additional upload latency of less than 50ms under a standard 20Mbps uplink connection, making it negligible in typical deployment scenarios.

Table III contextualizes our approach by comparing it with a standard FL baseline without privacy protection and various existing privacy-preserving schemes. Relative to standard FL [13], which directly transmits raw model updates, our scheme introduces minimal extra overhead while ensuring robust privacy. Compared to well-known secure aggregation methods such as Bonawitz et al. [28], our approach avoids multiple interactive communication rounds and substantial ciphertext expansions. By shifting mask reconstruction and verification entirely to server-side operations within our dual-server architecture, our method eliminates the need for repeated uploads and peer-to-peer interactions among participants. Consequently, our scheme effectively balances privacy protection, computational simplicity, and communication efficiency, making it highly suitable for large-scale applications.

VI. CONCLUSION

We described our proposed practical privacy-preserving collaborative deep learning scheme, designed to preserve the privacy of both participant models and their local data. We also showed that our proposal can withstand membership inference attacks, model inversion attacks and collusion attacks between one of the servers and up to $t-2$ participants, where t is a security parameter. In addition, we demonstrated that our scheme supports dynamic participation, is round-efficient, and achieves lossless model accuracy. Findings from the performance evaluation also demonstrated that our proposed approach incurs a very small overhead while achieving high model accuracy.

REFERENCES

[1] P. Voigt and A. Von dem Bussche, "The Eu General Data Protection Regulation (GDPR)," in *A Practical Guide*, 1st ed., vol. 10. Berlin, Germany: Springer, 2017, Art. no. 3152676.
[2] L. de laTorre, "A guide to the california consumer privacy act of 2018 (november 2018)," 2018. [Online]. Available: http://dx.doi.org/10.2139/ssrn.3275571
[3] C. Esposito, X. Su, S. A. Aljawarneh, and C. Choi, "Securing collaborative deep learning in industrial applications within adversarial scenarios," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4972–4981, Nov. 2018.
[4] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1486–1500, 2019.
[5] G. Song and W. Chai, "Collaborative learning for deep neural networks," 2018, *arXiv: 1805.11761*.
[6] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 1235–1244.
[7] X. Liu, J. Yang, S. Chen, H. He, and X. Jiang, "PRFL: Achieving efficient robust aggregation in privacy-preserving federated learning," *IEEE Trans. Netw. Service Manag.*, vol. 22, no. 2, pp. 2138–2155, Apr. 2024.
[8] L. Wang, M. Polato, A. Brighente, M. Conti, L. Zhang, and L. Xu, "PriVeriFL: Privacy-preserving and aggregation-verifiable federated learning," *IEEE Trans. Services Comput.*, vol. 18, no. 2, pp. 998–1011, Mar./Apr. 2025.
[9] Y. Gao, L. Zhang, L. Wang, K.-K. R. Choo, and R. Zhang, "Privacy-preserving and reliable decentralized federated learning," *IEEE Trans. Services Comput.*, vol. 16, no. 4, pp. 2879–2891, Jul./Aug. 2023.
[10] L. Zhong et al., "Dual-server-based lightweight privacy-preserving federated learning," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 4, pp. 4787–4800, Aug. 2024.

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.

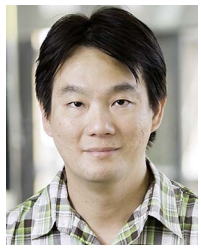
[11] C. Wu, L. Zhang, L. Xu, K.-K. R. Choo, and L. Zhong, "Privacy-preserving serverless federated learning scheme for Internet of Things," *IEEE Internet Things J.*, vol. 11, no. 12, pp. 22429–22438, Jun. 2024.
[12] Y. Dong et al., "Multi-task federated split learning across multi-modal data with privacy preservation," *Sensors*, vol. 25, no. 1, 2025, Art. no. 233.
[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[14] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," 2016, *arXiv:1602.05629*.
[15] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proc. 22nd ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, 2003, pp. 202–210.
[16] R. Wang, Y. F. Li, X. Wang, H. Tang, and X. Zhou, "Learning your identity and disease from research papers: Information leaks in genome wide association study," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 534–544.
[17] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE 31st Comput. Secur. Found. Symp.*, 2018, pp. 268–282.
[18] C. Song and V. Shmatikov, "Overlearning reveals sensitive attributes," 2019, *arXiv: 1905.11742*.
[19] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. 2017 IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.
[20] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1322–1333.
[21] 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 *Proc. 2019 IEEE Symp. Secur. Privacy*, 2019, pp. 1102–1120.
[22] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *Proc. 2013 IEEE Symp. Secur. Privacy*, 2013, pp. 334–348.
[23] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *Proc. 2017 IEEE Symp. Secur. Privacy*, 2017, pp. 19–38.
[24] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory cryptogr. Conf.*, 2006, pp. 265–284.
[25] C. Dwork et al., "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3/4, pp. 211–407, 2014.
[26] Y. Aono et al., "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.
[27] K. Cheng et al., "Secureboost: A lossless federated learning framework," 2019, *arXiv: 1901.08755*.
[28] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.
[29] X. Zhang, S. Ji, H. Wang, and T. Wang, "Private, yet practical, multiparty deep learning," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 1442–1452.
[30] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. 2020 {USENIX} Annu. Tech. Conf.*, 2020, pp. 493–506.
[31] B. Wang et al., "Privacy-preserving convolutional neural network classification scheme with multiple keys," *IEEE Trans. Services Comput.*, vol. 17, no. 1, pp. 322–335, Jan./Feb. 2024.
[32] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proc. 2014 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 1054–1067.
[33] M. Abadi et al., "Deep learning with differential privacy," in *Proc. 2016 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.
[34] J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "The limits of differential privacy (and its misuse in data release and machine learning)," *Commun. ACM*, vol. 64, no. 7, pp. 33–35, 2021.
[35] S. Truex et al., "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 1–11.
[36] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "HybridAlpha: An efficient approach for privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 13–23.
[37] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2017, pp. 409–437.
[38] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.
[39] H. Su and H. Chen, "Experiments on parallel training of deep neural network using model averaging," 2015, *arXiv:1507.01239*.
[40] Q. Wu, L. Zhang, Y. Yang, and K.-K. R. Choo, "Certificateless signature scheme with batch verification for secure and privacy-preserving V2V communications in VANETs," *IEEE Trans. Dependable Secure Comput.*, vol. 22, no. 2, pp. 1448–1459, Mar./Apr. 2025.
[41] B. Kang, Z. Huang, and L. Zhang, "Selective-opening security for public-key encryption in the presence of parameter subversion," *Secur. Commun. Netw.*, vol. 2021, no. 1, 2021, Art. no. 5895809.
[42] A. Paszke et al., "Automatic differentiation in pytorch," in *Proc. Int. Conf. Neural Inf. Process. Syst. Workshop*, 2017. [Online]. Available: https://openreview.net/pdf?id=BJJsrmfCZ
[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[44] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, Canada, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
[45] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1310–1321.
[46] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 911–926, 2019.



Lulu Wang received the PhD degree in software engineering from East China Normal University, Shanghai, China, in 2025. He is currently a postdoctoral researcher with the Hong Kong Polytechnic University. He was a visiting PhD student with the SPRITZ Security and Privacy Research Group, University of Padua, Italy (2023), and a visiting scholar in the Information Systems Technology and Design Pillar at the Singapore University of Technology and Design (2024). His research interests are at the intersection of security, privacy, and machine learning.



Lei Zhang (Member, IEEE) received the PhD degree in computer engineering from Universitat Rovira i Virgili, Tarragona, Spain. Since then, he has been with Universitat Rovira i Virgili, as a postdoctoral researcher. He is currently a full professor with the Software Engineering Institute, East China Normal University, Shanghai, China. He has been a holder/coholder of more than twenty China/Spain-funded (key) projects. His fields of activity are information security, VANET security, cloud security, data privacy, and network security. He has authored more than 100 publications. He has served in the program committee of more than 100 international conferences in information security and privacy.



Kim-Kwang Raymond Choo (Senior Member, IEEE) received the PhD degree in information security from the Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship with the University of Texas at San Antonio. He is the founding co-EiC of ACM Distributed Ledger Technologies: Research and Practice, a Fulbright Visiting Scholar at Rutgers University School of Criminal Justice and Palo Alto Research Center (PARC) in 2009, and a Fulbright Distinguished Scholar with the University of Iceland, in 2025.

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.



Josep Domingo-Ferrer (Fellow, IEEE) received the the BSc, MSc, and PhD degrees in computer science from the Autonomous University of Barcelona, also received the BSc, MSc degrees in mathematics from UNED and the MA degree in philosophy from the University of Paris-Nanterre. He is a distinguished full professor of computer science and an ICREA-academia research professor with Universitat Rovira i Virgili, Tarragona, Catalonia, where he leads CYBERCAT (Center for Cybersecurity Research of Catalonia). He is an invited professor with LAAS-CNRS, France. He is an ACM Distinguished Scientist, an elected member of Academia Europaea, an Elected Member of the International Statistical Institute, a Fellow of Institut d'Estudis Catalans, and a Fellow of AAIA. He has chaired the ERC Panel for Starting Grants in Computer Science, and he has received the Monturiol Medal for merit in science and technology. His research interests are in privacy technologies, data protection, trustworthy machine learning, and ethics-by-design in AI and IT. More information on him can be found at http://crises-deim.urv.cat/jdomingo



Mauro Conti (Fellow, IEEE) received the PhD degree from the Sapienza University of Rome, Italy, in 2009. He is full professor with the University of Padua, Italy. He is also affiliated with University of Washington, Seattle, and is a Wallenberg WASP guest professor with Örebro University, Sweden. After his PhD, he was a post-doc researcher with Vrije Universiteit Amsterdam, The Netherlands. In 2011 he joined as assistant professor with the University of Padua, where he became associate professor, in 2015, and full professor, in 2018. He has been visiting researcher with GMU, UCLA, UCI, TU Darmstadt, UF, FIU, and Affiliate Professor at TU Delft. He has been awarded with a Marie Curie Fellowship (2012) by the European Commission, with a Fellowship by the German DAAD (2013), and with a WASP Fellowship by the Knut and Alice Wallenberg Foundation (2025). His research is also funded by companies, including Cisco, Intel, and Huawei. His main research interest is in the area of Security and Privacy. In this area, he published more than 600 papers in topmost international peer-reviewed journals and conferences. He is area editor-in-chief for *IEEE Communications Surveys & Tutorials*, and has been editor-in-chief for *IEEE Transactions on Information Forensics and Security* (2022-24) and associate editor for several journals, including *IEEE Communications Surveys & Tutorials*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, and *IEEE Transactions on Network and Service Management*. He was program chair for TRUST 2015, ICISS 2016, WiSec 2017, ACNS 2020, CANS 2021, CSS 2021, WiMob 2023 and ESORICS 2023, and general chair for SecureComm 2012, SACMAT 2013, NSS 2021, ACNS 2022, RAID 2024, NDSS 2026 and 2027. He is Fellow of the AAIA, distinguished member of the ACM, and Fellow of the Young Academy of Europe.



Yuanyuan Gao received received the BS degree in computer science and technology from the Zhongnan University of Economics and Law, Wuhan, China and the MS degree in software engineering from East China Normal University, Shanghai, China, in 2022. Her research interests include privacy-preserving machine learning.

Authorized licensed use limited to: UNIVERSITAT POMPEU FABRA. Downloaded on December 22,2025 at 09:57:48 UTC from IEEE Xplore. Restrictions apply.