FedPHE: A Secure and Efficient Federated Learning via Packed Homomorphic Encryption

Yuqing Li, *Member, IEEE*, Nan Yan, Jing Chen, *Senior Member, IEEE*, Xiong Wang, *Member, IEEE*, Jianan Hong, *Member, IEEE*, Kun He, *Member, IEEE*, Wei Wang, *Member, IEEE*, and Bo Li, *Fellow, IEEE*

***Abstract*—Cross-silo federated learning (FL) enables multiple institutions (clients) to collaboratively build a global model without sharing private data. To prevent privacy leakage during aggregation, homomorphic encryption (HE) is widely used to encrypt model updates, yet incurs high computation and communication overheads. To reduce these overheads, *packed* HE (PHE) has been proposed to encrypt multiple plaintexts into a single ciphertext. However, the original design of PHE assumes all clients share a single private key, making the system vulnerable to security threats of ciphertexts being intercepted and decrypted by *honest-but-curious clients*. Also, it does not consider the *heterogeneity* among different clients, resulting in undermined training efficiency with slow convergence and stragglers. To address these challenges, we propose FedPHE, a secure and efficient FL framework with PHE by jointly exploiting contribution-aware secure aggregation and straggler-resistant client selection. Using CKKS with sparsification and obfuscating, FedPHE achieves efficient secure aggregation that allows clients to only provide *obscured* encrypted updates while the server can perform aggregation by accounting for *contributions* of local updates. To mitigate the straggler effect, we devise a *perturbed sketch*-based selection to cherry-pick representative clients with *heterogeneous models and computing capabilities* in a communication-efficient and privacy-preserving manner. We show, through rigorous security analysis and extensive experiments, that FedPHE can efficiently safeguard clients' privacy, achieve $2.45-6.56\times$ training speedup, cut the communication overhead by $1.32-24.85\times$, and reduce straggler effects by $1.89-2.78\times$.**

***Index Terms*—Federated learning (FL), packed homomorphic encryption (HE), secure aggregation, client heterogeneity, client selection.**

I. Introduction

Cross-silo federated learning (FL) [2], [3] is an emerging distributed learning paradigm that enables multiple institutions (e.g., banks, companies), referred to as clients, to collaboratively train a global model without sharing their private data [4], [5]. In a typical cross-silo FL system, a central parameter server (PS) orchestrates many clients to aggregate local updates (e.g., gradients, model parameters) in multiple rounds of *synchronization*. Although this system does not reveal the clients' raw data in the clear, it has been shown that adversaries can still infer a client's private information from its updates [6], [7], [8].

To avoid *privacy leakage* during aggregation, many privacy-preserving techniques have been employed for FL [9], [10]. Among them, *homomorphic encryption* (HE) is particularly attractive to cross-silo FL, as it provides stronger privacy guarantees without compromising the learning accuracy [11], [12], [13]. With HE, aggregation can be performed directly on ciphertexts, without decrypting them first. However, HE incurs significant computation and communication overheads as it performs computationally intensive cryptographic operations (e.g., modular multiplications and polynomial reductions) and generates ciphertexts that are much larger to transfer than the input plaintexts [14], [15]. A promising approach to address this problem is *packed* HE (PHE), which packs and encrypts multiple plaintext values into a single ciphertext [16]. By facilitating parallel encryption/decryption operations on multiple plaintexts, PHE dramatically reduces the encryption and communication overheads.

Though effective, existing PHE solutions [1], [17] commonly adopt *single-key HE* like Paillier and CKKS, with the assumption that all clients share the *same encryption and decryption keys*.

Received 9 April 2024; revised 24 March 2025; accepted 27 April 2025. Date of publication 6 May 2025; date of current version 4 September 2025. This work was supported in part by the National Key R&D Program of China under Grant 2022YFB3102100, in part by the National Natural Science Foundation of China under Grant 62302343, Grant 62202185, Grant 62472323, and Grant 62202290, in part by the Key R&D Program of Hubei Province under Grant 2024BAB018, in part by Wuhan Scientific and Technical Achievements Project under Grant 2024030803010172, in part by the Wuhan City Joint Innovation Laboratory for Next-Generation Wireless Communication Industry Featuring Satellite-Terrestrial Integration under Grant 4050902040448, in part by the Hubei Province international Technology Science and Collaboration Program under Grant 2023EHA044, in part by the Key R&D Program of Shandong Province under Grant 2022CXPT055, in part by RGC RIF Grant under Contract R6021-20, in part by RGC TRS Grant under Contract T43-513/23N-2, and in part by RGC GRF Grants under Contract 16211123, Contract 16200221, Contract 16207922, and Contract 16207423. An earlier version of this paper was presented in the proceedings of INFOCOM 2024 [DOI: 10.1109/INFOCOM52122.2024.10621440]. *(Corresponding author: Jing Chen.)*

Yuqing Li, Nan Yan, and Kun He are with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: li.yuqing@whu.edu.cn; nanyan@whu.edu.cn; hekun@whu.edu.cn).

Jing Chen is with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the RiZhao Information Technology Institute, Wuhan University, Rizhao 276800, China (e-mail: chenjing@whu.edu.cn).

Xiong Wang is with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab/Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: xiongwang@hust.edu.cn).

Jianan Hong is with the School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: hongjn@sjtu.edu.cn).

Wei Wang and Bo Li are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, Hong Kong (e-mail: weiwa@cse.ust.hk; bli@cse.ust.hk).

Digital Object Identifier 10.1109/TDSC.2025.3567301

Typically, each client encrypts local updates using the shared encryption key before submitting them to PS, ensuring that neither the PS nor external adversaries can learn any information. Nevertheless, since all clients apply the same decryption key, an *honest-but-curious client* has the ability to intercept and decrypt the data from others. Hence, PHE can be secure when all clients fully trust each other, yet it is hard to guarantee in practice, particularly in cross-silo FL scenarios. Without addressing these security challenges, the potential of PHE cannot be fully unleashed.

Also, the original design of PHE largely ignores *client heterogeneity* [18], [19], an intrinsic problem of cross-silo FL, making them hard to deploy in practice. On one hand, data are distributed in an unbalanced fashion across clients (i.e., statistical heterogeneity), which often leads to discrepancies in local models and adversely impacts convergence behavior. This way, additional encrypted communications will be implemented, undermining the training efficiency of FL. On the other hand, clients may have varying computing capacities and communication bandwidth (i.e., system heterogeneity). This results in a prominent straggler problem, which can be further exacerbated by computationally intensive encryption/decryption operations, significantly slowing down the training progress. A large body of works have been proposed to address the heterogeneity issues [20], [21]. One common approach is *weighted aggregation* [22]. As datasets held by clients may have different contributions to model performance, vanilla aggregation often causes serious bias to the global model that hinders convergence. It is thus desirable to differentiate between the contributions of local updates during aggregation. Another popular approach is to judiciously *select a subset of clients* to participate into training, as not all clients are equally important [18], [23]. By identifying fast clients with quality data and involving them in the training process, the straggler issue can be effectively addressed without compromising model accuracy. Although many efforts have been devoted to weighted aggregation and client selection, they were largely explored separately and designed for plaintext data without any encryption protection. In general, weighted aggregation is performed based on client selection to collect clients' contributions, meanwhile, the aggregated global model also affects local model training and, in turn, determines the client selection. It is hence imperative to handle these problems to achieve efficient PHE for heterogeneous FL.

To build a secure and efficient cross-silo FL system, we have to cope with three major challenges. First, existing privacy-preserving weighted aggregation studies primarily rely on single-key HE [24], where all clients share the same encryption key, making the system vulnerable to eavesdropping or collusion attacks by malicious clients. Although multi-key HE [25], [26] has been proposed to address this issue by allowing clients to encrypt their local updates with different encryption keys, it incurs excessive computation overheads, especially when deploying large-scale FL. Consequently, training efficiency will be severely impeded by cryptographic computations, underscoring the necessity for achieving *secure and lightweight aggregation* for FL. Second, existing HE solutions [17] often rely on homomorphic addition for secure weighted aggregation, where clients directly weigh their local updates based on data size and encrypt the weighted updates for aggregation. However, this approach becomes infeasible under client heterogeneity, as a client's data quantity does not necessarily reflect its overall contribution to the global model. Additionally, since the aggregation weights are calculated by clients themselves, there is a risk that a client could misreport its weight to skew the global model towards its local training. As such, it is desired to employ *homomorphic multiplication* on ciphertext for weighted aggregation on the PS side. This introduces potential *communication bottlenecks* as homomorphic multiplication typically requires a large ciphertext space for encryption [27]. Third, existing client selection approaches [18] are mainly carried out in plaintext and require direct access to local model updates, raising *privacy concerns* in FL. Accurately measuring clients' contributions to the global model is challenging due to client heterogeneity, which is further exacerbated by privacy protection requirements. Even if achieved, the selection efficiency will be significantly compromised as data encryption demands many extra operations (e.g., communications and computations), which can be overly expensive. Thus, we have to carefully navigate the tradeoff between *security* and *efficiency* in client selection.

In this paper, we propose FedPHE, a secure and efficient PHE-based FL framework to tackle challenges associated with security threats and client heterogeneity. FedPHE develops a contribution-aware secure aggregation scheme using the packed CKKS techniques, which supports homomorphic multiplication. In a nutshell, the PS aggregates the encrypted local updates from the selected clients with encrypted weights accounting for their contributions to the global model, facilitating the model to quickly incorporate new knowledge and thus accelerating training convergence. Given that vanilla CKKS often generates substantially enlarged ciphertexts, we use a *pack-based* sparsification approach to optimize data transfer efficiency during periodical encrypted FL synchronizations. To enhance privacy against honest-but-curious adversaries, we employ a secret sharing-based obfuscating technique to obscure the individual updates with the random factor, later to be recovered and removed to obtain the correct aggregated result. This way, the encrypted local updates are transmitted and aggregated in the obfuscated form. To mitigate the straggler effect, FedPHE devises a *sketch-based* client selection scheme to judiciously select clients that host diverse models with fast training capability. The key insight is that different clients might send *similar or redundant* model updates to the PS, incurring unnecessary communication costs. We propose to measure the similarity of local updates using the sketching technique which maps high-dimensional model updates to a lower dimension through entry hashing. To further avoid potential privacy leakage from such sketching, clients perturb the computed sketches of their model updates before sending to PS. The PS then removes these perturbations in the received sketches, clusters clients with similar sketches together and only selects the fastest client from each cluster. This ensures that clients are selected in a communication-efficient and privacy-preserving manner. We provide rigorous security analysis for FedPHE and also validate its efficiency through empirical studies.

We summarize our main contributions as follows:

- We propose FedPHE, a secure and efficient cross-silo FL framework with PHE. To our knowledge, this is the first attempt that enables both *contribution-aware secure* aggregation and *sketch-based straggler-resistant* client selection to effectively address the *security threats* and *heterogeneity* challenges, thus closing the gap between privacy-preserving FL and its practical implementation.
- Building upon CKKS's homomorphic encryption, FedPHE achieves efficient encrypted weighted aggregation that accounts for *contributions* of local updates to the global model. To enhance both *efficiency* and *security* of data transfer, the pack-level sparsification, and secret sharing-based obfuscating schemes are jointly implemented, addressing the issues of increased ciphertext size using vanilla CKKS and ciphertext being decrypted by honest-but-curious clients.
- FedPHE leverages *perturbed* sketches of local updates to facilitate a communication-efficient client selection in a privacy-preserving manner. By jointly considering *data distributions* and *resource availability*, FedPHE clusters similar clients together and then cherry-picks the *fastest* client from each cluster, effectively mitigating the straggler problem without compromising model accuracy.
- Rigorous security analysis demonstrates that FedPHE is secure against honest-but-curious adversaries. Extensive experiments on real-world datasets show that compared to the state-of-the-art approaches, FedPHE accelerates the training speed by 2.45-6.56×, cuts the communication overhead by 1.32-24.85×, and mitigates the straggler effect by 1.89-2.78×, with a slight degradation of model accuracy (1.45% only).

II. Preliminaries and Motivation

We start by introducing the basics of cross-silo FL and the HE technique. We then motivate the design of FedPHE.

A. Cross-Silo Federated Learning

Consider a cross-silo FL system consisting of a central PS and a set of N clients $\mathcal{N} = \{1, \ldots, N\}$ that collaboratively train a machine learning model without sharing their raw data. Each client i holds a local dataset $\mathcal{D}_i$ containing $D_i = |\mathcal{D}_i|$ data samples. Let $f_i(\boldsymbol{w}, \xi_i)$ be the loss value computed from the training sample $\xi_i \in \mathcal{D}_i$ with parameters $\boldsymbol{w}$. The local loss function of client i is computed as

$$f_i(\boldsymbol{w}) \triangleq \frac{1}{D_i} \sum_{\xi_i \in \mathcal{D}_i} f_i(\boldsymbol{w}, \xi_i). \tag{1}$$

The goal of clients is to jointly solve the following optimization problem, under the coordination of the PS:

$$\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \triangleq \min_{\boldsymbol{w}} \sum_{i \in \mathcal{N}} p_i f_i(\boldsymbol{w}), \tag{2}$$

where $\mathcal{L}(\boldsymbol{w})$ is the global loss function, and p_i is client i's aggregation weight, where $p_i \geq 0$ and $\sum_{i \in \mathcal{N}} p_i = 1$. To solve (2), clients perform a synchronous update ($g_i = \nabla f_i(\boldsymbol{w})$) that proceeds in rounds of communication.

A key requirement for cross-silo FL is to provide a strong *privacy* guarantee, as there is increasing evidence that adversaries can extract private information from client updates even when training data are kept locally [6], [8].

B. Homomorphic Encryption

HE is a powerful cryptographic primitive that enables computations to be performed directly on the encrypted data without decrypting them in advance [28], [29]. HE ensures that the calculations performed on ciphertexts, when decrypted, give identical results of that obtained by directly operating on the plaintexts. More formally, an encryption scheme $\boldsymbol{E}(\cdot)$ is said to be an *additive* HE scheme if $\boldsymbol{E}(m_1) \oplus \boldsymbol{E}(m_2) = \boldsymbol{E}(m_1 + m_2)$ for any plaintext messages m_1 and m_2, where $\oplus$ is an addition operation. Similarly, a scheme is a *multiplicative* HE scheme if $\boldsymbol{E}(m_1) \odot \boldsymbol{E}(m_2) = \boldsymbol{E}(m_1 \cdot m_2)$, where $\odot$ is a multiplication operation. Popular HE schemes include Paillier [30], BFV [31], and CKKS [32], where Paillier only allows the addition operation to be performed on ciphertexts, whereas BFV and CKKS support both additions and multiplications.

While HE allows the computation to be securely delegated to an untrusted third party, it suffers from critical inefficiency associated with encryption operations and ciphertext transmissions. Many efforts have been devoted to improving HE efficiency [14], [15], [33]. Among them, a promising approach is PHE which packs and encrypts multiple plaintext values into a single ciphertext, allowing for parallel encryption/decryption operations [17]. However, current (packed) homomorphically encrypted FL solutions either often assume all clients use a single encryption key, or largely ignore the intrinsic *client heterogeneity* in a cross-silo FL, substantially limiting their practical applications.

C. Motivation

The need for weighted aggregation on ciphertexts: In real-world FL systems, the training datasets owned by clients often have different contributions to the global updates, a phenomenon known as the *statistical heterogeneity*. In this case, simply performing unweighted aggregation, i.e., $\boldsymbol{g}^{t+1} = \sum_{i \in \mathcal{N}} \frac{1}{N} \boldsymbol{g}_i^t$ where $p_i = \frac{1}{N}$, results in an undesirable *bias* to the global updates that hinder the training convergence. A more appropriate approach is to perform weighted aggregation, in which clients are assigned different weights for aggregation based on their contributions (e.g., data size or quality). Taking FedAvg [34] as an example, the weighted aggregation based on data size is performed to minimize the loss in (2), i.e.,

$$\boldsymbol{g}^{t+1} = \sum_{i \in \mathcal{N}} p_i \boldsymbol{g}_i^t = \sum_{i \in \mathcal{N}} \frac{D_i}{\sum_{i \in \mathcal{N}} D_i} \boldsymbol{g}_i^t, \tag{3}$$

where $\boldsymbol{g}_i^t$ and $\boldsymbol{g}^t$ are the local and global updates. To illustrate the significance of weighted aggregation, we refer to Fig. 1. Compared to the unweighted approach, weighted aggregation (weight set based on the data size) results in much-improved accuracy loss and faster convergence. However, existing HE

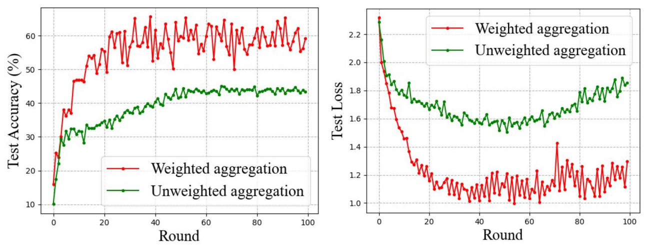


Fig. 1. Comparisons of weighted and unweighted aggregation.

TABLE I
COMPARISONS OF DIFFERENT (PACKED) HE SCHEMES

(Packed) HE scheme	Ciphertext size	Encryption time (s)	Decryption time (s)
Paillier	21.97 MB	63.46	39.63
PackedPaillier	264.96 KB	3.18	2.60
BFV		**Memory out**	
PackedBFV	22.68 MB	0.04	0.02
CKKS		**Memory out**	
PackedCKKS	4.54 MB	0.06	0.04

solutions [17] employ either unweighted aggregation or homomorphic addition-based aggregation, making them *infeasible* to support general weighted aggregation. It is hence desired to employ *homomorphic multiplication* on ciphertext for secure weighted aggregation.

The need for efficient secure aggregation: HE-based FL methods offer strong privacy guarantees, albeit at the expense of efficiency. Table I shows the comparison results of Paillier, BFV, and CKKS as well as their packed implementation with plaintext size 109.89 KB. Specifically, Paillier generates a ciphertext close to $205\times$ larger than the plaintext, while consuming considerable computation time. BFV and CKKS produce larger ciphertexts than Paillier and even lead to *memory overflow*. Though the PHE technique can address these issues, the yielded communication overheads remain too high, resulting in an *inflation* of $2.4\times$ for Paillier, $211.3\times$ for BFV, and $42.3\times$ for CKKS. Moreover, existing PHE solutions commonly adopt *single-key HE* like Paillier and CKKS, where all clients are assumed to share a single private key, making the system vulnerable to security threats of ciphertexts being intercepted and decrypted by *honest-but-curious clients*. Thus, it is imperative to jointly address these two issues for achieving efficient secure aggregation for FL.

The need for straggler resistance: In synchronous cross-silo FL systems, client heterogeneity inevitably results in stragglers, i.e., slow clients. This problem is further exacerbated with computationally intensive HE operations. Waiting for these stragglers significantly prolongs FL training. Although it seems feasible to set a staleness bound by directly ignoring stragglers, deriving the optimal bound is challenging. Table II illustrates the straggler effect, which extends the training time by 91.0% and the encryption/decryption time by 93.6%. Moreover, normal clients, except for stragglers, experience an extra 36.2% waiting time. Therefore, stragglers cause high *latency* and hinder

TABLE II
BREAKDOWN OF TRAINING ITERATION TIME FOR NORMAL CLIENTS AND STRAGGLERS

Clients \ Time (s)	Training	Encryption	Idle	Decryption
Normal clients	3.24	6.68	8.25	4.65
Stragglers	6.19 ↑	12.24 ↑	2.00 ↓	9.69 ↑

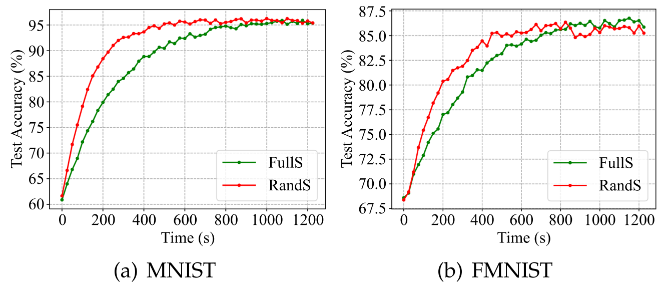


Fig. 2. Comparisons of **FullS**election and **Rand**om**S**election.

synchronization efficiency, necessitating the straggler-resistant solutions for FL.

The need for client selection. One possible remedy for stragglers is to select a subset of clients to participate in FL. As shown in Fig. 2, a simple approach that *randomly* chooses 80% of clients can significantly decrease the delay without sacrificing model accuracy. Although random selection expedites convergence by reducing the selection probability of stragglers, it fails to tackle the straggler issue at its core. Moreover, existing client selection methods are carried out in plaintext, which is susceptible to privacy leakage [35]. Hence, how to achieve efficient and privacy-preserving client selection remains challenging for mitigating the straggler effect.

III. DESIGN OF FEDPHE

In this section, we describe FedPHE, a secure and efficient PHE-based FL framework designed to address challenges associated with *security threats* and *client heterogeneity*. We begin with a system overview and then elaborate on how FedPHE co-designs contribution-aware secure aggregation and sketch-based straggler-resistant client selection, followed by its security analysis.

A. Overview

Our proposed system consists of three types of entities: a key distribution center (KDC), a central PS and a set of N clients. The KDC is widely adopted in current research for key distribution and management [24], [36], [37]. Its primary role is to generate and dispatch keys, vectors, and local obfuscating factors to clients and the PS via secure channels. These operations are computationally lightweight and can be pre-computed, making them well-suited for the KDC infrastructure. During collaborative model training, data exchanges are carried out between clients and PS through insecure channels.

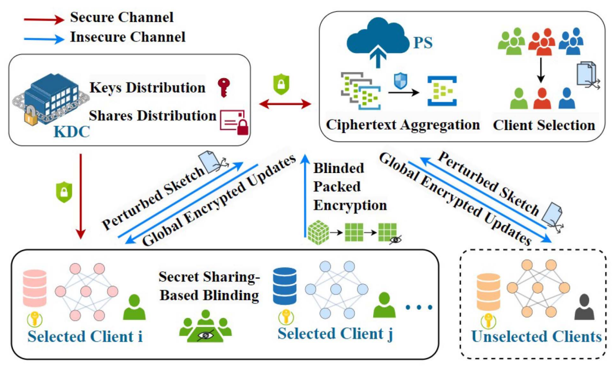


Fig. 3. A snapshot of FedPHE architecture.

Threat model: In our system, the KDC is regarded as fully trusted. The PS and clients are assumed to be *honest-but-curious*, which is commonly adopted in existing privacy-preserving FL works [26], [38]. That is, they will honestly follow the protocol design, yet are curious about the private data of each client. In our threat model settings, honest-but-curious clients may try to *eavesdrop* and decrypt encrypted local updates or *collude* to obtain a certain client's private data, but they do not launch poisoning attacks. We consider a more practical threat model with insecure communication channels. While TLS/SSL protocols establish secure channels, they remain vulnerable to internal adversaries, compromised participants, or potential key leakage.

Design goals: Our design goal is to develop a secure and efficient FL framework with PHE. Specifically, the following desirable objectives should be achieved.

- *Privacy:* It should protect the privacy of local datasets against honest-but-curious PS and clients. Specifically, the PS cannot access the private training data of any particular client, while clients cannot infer any private information about other clients' datasets, even when eavesdropping attacks are launched.
- *Efficiency:* It is expected to be efficient for encrypting local updates and transferring ciphertexts since high computation and communication overheads make PHE-based FL difficult to implement in practice. Furthermore, it should effectively mitigate the negative impact of stragglers under client heterogeneity, accelerating the training process without sacrificing model accuracy.

Architecture: We propose FedPHE to attain the above two objectives by jointly designing *contribution-aware secure* aggregation and *sketch-based straggler-resistant* client selection. Fig. 3 shows the main architecture, where FedPHE proceeds in rounds of communication as described below.

① *Key Distribution:* KDC generates and dispatches keys and vectors to clients in secure channels every τ rounds.

② *Local Training:* At the beginning of each round t, every client $i \in \mathcal{N}$ runs E steps of local stochastic gradient descent in (2) to compute local update $\boldsymbol{g}_i^t$;

③ *Sketching Local Updates with Perturbation:* Using vectors $\mathcal{V}_c, \mathcal{V}_{s,i}$ received from KDC, client $i \in \mathcal{N}$ computes and sends the perturbed sketch $\widetilde{h}_i^t$ of local update $\boldsymbol{g}_i^t$ to PS;

④ *Client Selection via Clustering Sketches:* The PS recovers the original sketches by removing personalized perturbations and clusters them to select a subset of clients $\mathcal{S}^t$ as participants, where the aggregation weight p_i^t for each selected client is derived based on its contribution;

⑤ *Packed Encryption with Sparsification and Obfuscating:* KDC computes and dispatches local obfuscating factors $\{\Delta_i\}_{i\in\mathcal{S}^t}$ to selected clients. Each selected client $i \in \mathcal{S}^t$ performs sparsification by *packing* local updates $\boldsymbol{g}_i^t$ into $\{\mathcal{P}_i^1, \ldots, \mathcal{P}_i^K\}$, and *obfuscates* the sparsely packed local updates with Δ_i; After that, it encrypts the obfuscated packed local updates and sends the ciphertext $\mathcal{C}_i^t$ along with mask M_i^t to PS for aggregation; Finally, it collaborates with other selected clients to retrieve obfuscating factor R and sends it to unselected clients within the cluster.

⑥ *Encrypted Weighted Aggregation:* The PS aggregates the received encrypted local updates along with encrypted weights, and then computes and dispatches the encrypted global updates $\boldsymbol{C}^t$ and mask $\boldsymbol{M}^t$ to all clients;

⑦ *Decryption and Model Update:* Every client decrypts, deobfuscates and unpacks the encrypted global updates, and then performs the local model updates.

The details of FedPHE are shown in Algorithm 1. KDC first distributes the updated keys and perturbation vectors to clients every τ rounds (lines 3–6). During each global round t, clients execute local training and send perturbed sketches of local updates to PS (lines 18–22). The PS then selects a subset of clients $\mathcal{S}^t$ as participants by clustering sketches (lines 11–12). Given local obfuscating factors received from KDC, the selected clients perform PHE with sparsification and obfuscating, and transmit obfuscated encrypted model updates and masks to PS (line 25). Taking into account clients' contributions, PS conducts secure aggregation and sends the results to all clients (lines 13–14). Finally, clients decrypt encrypted global updates and update their local models (lines 29–31). Notice that the update frequency τ is critical for balancing security and efficiency. A lower τ triggers more frequent key and vector updates by the KDC, narrowing the window for adversaries to exploit but increasing communication and computation overhead, while a higher τ cuts these overheads yet may compromise security.

B. Contribution-Aware Efficient Secure Aggregation

Building upon PHE with *sparsification* and *obfuscating*, FedPHE conducts an efficient secure aggregation on the ciphertexts received from selected clients as illustrated in Algorithm 2. In particular, the aggregation weights are adjusted based on the contributions of local updates to global updates so as to accommodate client heterogeneity.

CKKS-based PHE: To improve the efficiency of general HE schemes, PHE [16] is proposed by *packing* and *encrypting* multiple plaintext values $\{g_1, g_2, \ldots, g_B\}$ into a single ciphertext, where B is the packing size. By facilitating parallel encryption/decryption operations on multiple plaintexts, PHE can greatly reduce computation and communication overheads. In this work, we choose CKKS as the basis of PHE, providing

Algorithm 1: FedPHE.

```
   Input: Clients $\mathcal{N}$, global round $T$, local steps $E$,
          learning rate $\eta$, update frequency $\tau$
   Output: Global model $\boldsymbol{w}^T$
1  Initialize models $\{\boldsymbol{w}_i\}_{i\in\mathcal{N}}$;
   // KDC
2  for each round $t \in \{0, \cdots, T-1\}$ do
3      if $t$ mod $\tau = 0$ then
4          Generate keys $\{sk, pk\}$, vectors $\mathcal{V}_c$, $\{\mathcal{V}_{s,i}\}_{i\in\mathcal{N}}$;
5          Dispatch $\{sk, pk, \mathcal{V}_c, \mathcal{V}_{s,i}\}$ to client $i \in \mathcal{N}$;
6          Dispatch $\{\mathcal{V}_{s,i}\}_{i\in\mathcal{N}}$ to the PS;
7      Compute and send $\{r_d, r_i, \Delta_i\}$ to client $i \in \mathcal{S}^t$;
   // PS
8  for each round $t \in \{0, \cdots, T-1\}$ do
9      if $t$ mod $\tau = 0$ then
10         Receive $\{\mathcal{V}_{s,i}\}_{i\in\mathcal{N}}$ from KDC;
11     Receive perturbed sketches $\{\widetilde{\boldsymbol{h}}_i^t\}_{i\in\mathcal{N}}$ from clients;
12     Run client selection by Alg. 3, send $\mathcal{S}^t$ to clients;
13     Receive $\{\mathcal{C}_i^t, M_i^t\}$ from selected clients $i \in \mathcal{S}^t$;
14     Run secure aggregation by Alg. 2, send $\boldsymbol{C}^t$, $\boldsymbol{M}^t$
        to selected clients;
   // Client $i \in \mathcal{N}$
15 for each round $t \in \{0, \cdots, T-1\}$ do
16     if $t$ mod $\tau = 0$ then
17         Receive $\{sk, pk, \mathcal{V}_c, \mathcal{V}_{s,i}\}$ from KDC;
18     for each step $j \in \{0, \cdots, E-1\}$ do
19         $g_i(\boldsymbol{w}_{i,j}^t) \leftarrow \nabla f_i(\boldsymbol{w}_{i,j}^t)$;
20         $\boldsymbol{w}_{i,j+1}^t \leftarrow \boldsymbol{w}_{i,j}^t - \eta \cdot g_i(\boldsymbol{w}_{i,j}^t)$;
21     $\boldsymbol{g}_i^t \leftarrow \boldsymbol{g}_i(\boldsymbol{w}_{\boldsymbol{i},\boldsymbol{E}}^t)$;
22     Compute and send perturbed sketch $\widetilde{\boldsymbol{h}}_i^t$ to the PS;
23     Receive selection set $\mathcal{S}^t$ from the PS;
24     if $i \in \mathcal{S}^t$ then
25         Run secure aggregation by Alg. 2, send
            $\mathcal{C}_i^t, M_i^t$ to the PS and $R$ to unselected clients;
26     else
27         Receive $R$ from the selected client;
28     Receive encrypted global updates $\boldsymbol{C}^t$, mask $\boldsymbol{M}^t$;
29     Decrypt and unblind to get $\mathcal{P} \leftarrow \boldsymbol{D}(\boldsymbol{C}^t, sk) - R$;
30     $\boldsymbol{g}^t \leftarrow$ Unpack $\mathcal{P}$ to obtain global updates;
31     $\boldsymbol{w}_i^{t+1} \leftarrow \boldsymbol{w}_i^t - \eta \cdot \boldsymbol{g}^t$;
```

several advantages over Paillier and BFV. On one hand, CKKS allows direct encryption of *real numbers* on *vectors*, while Paillier and BFV are limited to encrypting *integer* plaintexts, which requires *quantizing* floating-point numbers and introduces the risk of overflow or loss of precision. In contrast, CKKS improves the efficiency of encryption and decryption operations by packing multiple vector elements into a single polynomial directly, thus enhancing the overall effectiveness of cryptographic procedures. Fig. 4 illustrates this packing process concisely. On the other hand, CKKS supports homomorphic multiplication, making it suitable for achieving secure weighted aggregation under heterogeneous cross-silo FL. On the contrary, Paillier only enables homomorphic addition, and BFV may encounter overflow issues when multiplying the quantized integers after encryption. Hence, CKKS is deemed more viable to directly multiply the encrypted parameters with encrypted weights on the PS side during aggregation. This way, it is equivalent to performing weighted aggregation first, followed by encryption of the aggregated result.

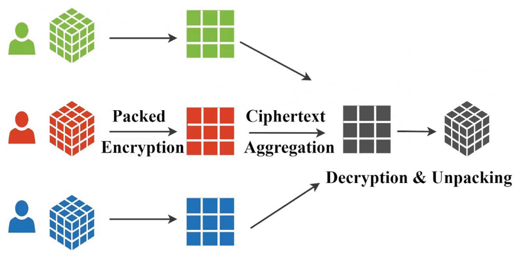


Fig. 4. Illustration of CKKS packing mode.

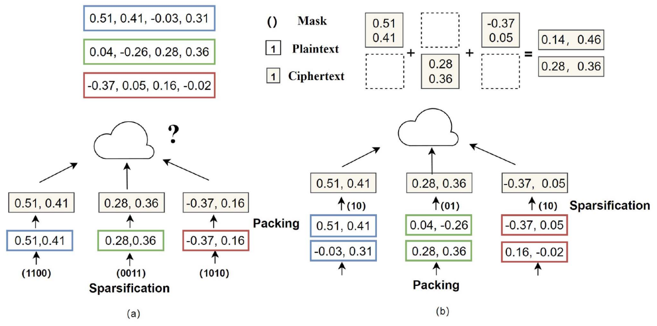


Fig. 5. Top-k sparsification. (a) shows the conventional packing method for HE, which is incompatible with sparsification when applied to ciphertexts; (b) presents an alternative approach to resolving this issue by packing prior to sparsification.

Despite the potential advantages, CKKS-based PHE still incurs high communication overheads and also suffers from the risk of ciphertexts being decrypted by honest-but-curious clients. To further enhance its *efficiency* and *security*, the pack-level sparsification and secret sharing-based obfuscating schemes are jointly implemented.

Pack-level sparsification: Sparsification is a promising approach for reducing the communication traffic in FL training [39]. In top-k sparsification, each client can sparsify its model updates by only selecting the top-k model updates to send to the PS. However, sparsification is mainly implemented at the *scalar* level, and becomes *infeasible* once the data is packed and encrypted. The reason is that if sparsification is conducted before encryption, the PS cannot perform alignment on ciphertexts due to *inconsistent* coordinate masks [33], Fig. 5 provides a clear visualization of this process. An alternative method is packing first and then sparsifying the packs. In this case, the PS can align the ciphertexts based on the packs' masks, i.e., the sparsification granularity is at the packing level.

Each selected client starts by flattening and packing the local model, and then determines the mask given the sparsification ratio ζ. The mask will be set to 1 for those packs with the top ζ ratio largest $L2$-norm values, and the masked packs are obfuscated and encrypted accordingly. The ciphertext $\mathcal{C}_i^t$ along with the corresponding mask M_i^t are sent to the PS. Notice that such sparsification techniques for CKKS can be further applied to enhance the efficiency of Paillier and BFV in heterogeneous scenarios, where clients' model updates are *weighted locally*.

Secret sharing-based obfuscating: While HE allows computing on encrypted data, it still suffers from security threats of ciphertexts being intercepted and decrypted by honest-but-curious clients. To address these security issues, we propose a *Shamir secret sharing-based obfuscating* mechanism that splits a secret obfuscating number into multiple shares and distributes these shares to clients, thereby securely obfuscating the sparsely packed local updates before sending to PS. When enough clients with obfuscating shares work together, the secret can be correctly reconstructed.

To realize this, a *Shamir (th,* $|\mathcal{S}^t|$*) threshold secret sharing* protocol is employed using polynomial interpolation. Specifically, each selected client i receives a obfuscating number r_i (secret) and constructs a polynomial $\boldsymbol{F}_i(x)$ of degree $th-1$ whose constant term is r_i (i.e. $\boldsymbol{F}_i(0)=r_i$) and whose remaining coefficients are randomly chosen from the finite field p, denoted as $a_1, a_2, \dots, a_{th}$. That is,

$$\boldsymbol{F}_i(x) = r_i + a_1 x + a_2 x^2 + \cdots + a_{th} x^{th-1} \pmod{p}, \quad (4)$$

where $th < |\mathcal{S}^t|$ is called the threshold. Then the client splits r_i into $|\mathcal{S}^t|$ shares $\{r_{i,j}\}_{j\in\mathcal{S}^t}$ according to $\boldsymbol{F}_i(x)$, and exchanges them with others. Upon decrypting global updates, each client performs an deobfuscating operation to compute $\widetilde{r}_i = \sum_{j=1}^{|\mathcal{S}^t|} r_{j,i}$, and then collaboratively recovering $\sum_{i\in\mathcal{S}^t} r_i$ as r through Lagrange interpolation, i.e.,

$$r = \sum_{i\in\mathcal{S}^t} r_i = \sum_{i=1}^{th} \widetilde{r}_i \cdot \prod_{j=1, j\neq i}^{th} \frac{x - x_j}{x_j - x_i} \pmod{p}. \quad (5)$$

This allows clients to construct the original polynomial and cumulative obfuscating values r provided at least th shares are known, all without disclosing their individual obfuscating numbers to each other.

To mitigate the risk of privacy leakage when exchanging secret shares among clients, KDC further generates a unique obfuscating number r_d and employs a lightweight *additive secret sharing* protocol to securely dispatch local obfuscating factors $\{\Delta_i\}_{i\in\mathcal{S}^t}$ to clients. When preparing to obfuscate data, client i combines its unique obfuscating factor Δ_i provided by KDC. After decrypting the global updates, it subtracts the global obfuscating factor $R = r + r_d$ to deobfuscate the result. This *dual-layer* obfuscating scheme provides a strong safeguard against unauthorized data access and enhances the privacy of the distributed secret sharing system.

Contribution-aware weighted aggregation: To accommodate client heterogeneity, the PS aggregates encrypted local updates from selected clients with encrypted weights accounting for their contributions to the global model. Here the contribution of client i is quantified based on the *similarity* of its sketches in the last round and current round, $\{h_i^{t-1}, h_i^t\}$. *Locality-Sensitive Hashing* (LSH) [40] has been widely employed in many applications to approximate *Jaccard Similarity*, i.e.,

$$\Pr_{\mathcal{H}}(h_i^{t-1} = h_i^t) = JS(g_i^{t-1}, g_i^t). \quad (6)$$

Algorithm 2: Efficient Secure Aggregation.

```
Input: Selected clients $\mathcal{S}^t$, round $t$, number of packs
       $K$, sparsification ratio $\zeta$, public key $pk$
Output: Encrypted global updates $\boldsymbol{C}^t$ and mask $\boldsymbol{M}^t$
// KDC
1  Generate $r_d$, $\{r_i\}_{i\in\mathcal{S}^t}$ and $\{r_{d,i}\}_{i\in\{1,\cdots,|\mathcal{S}^t|-1\}}$;
2  $r_{d,|\mathcal{S}^t|} \leftarrow (r_d - \sum_{i=1}^{|\mathcal{S}^t|-1} r_{d,i}) \bmod p$;
3  Receive $\{p_i^t\}_{i\in\mathcal{S}^t}$ from PS;
4  Compute $\{\Delta_i \leftarrow (r_i + r_{d,i})/p_i^t\}_{i\in\mathcal{S}^t}$;
5  Dispatch $\{r_d, r_i, \Delta_i\}$ to selected client $i \in \mathcal{S}^t$;
// PS
6  Receive $\{\mathcal{C}_i^t, M_i^t\}_{i\in\mathcal{S}^t}$ from selected clients;
7  $\boldsymbol{C}^t \leftarrow \sum_{i\in\mathcal{S}^t} \boldsymbol{E}(p_i^t, pk)\cdot\mathcal{C}_i^t$, $\boldsymbol{M}^t \leftarrow \sum_{i\in\mathcal{S}^t} p_i^t \cdot M_i^t$;
8  Dispatch $\boldsymbol{C}^t$ and $\boldsymbol{M}^t$ to all clients;
// Client $i \in \mathcal{S}^t$
9  Receive $\{r_d, r_i, \Delta_i\}$ from KDC;
10 $\mathcal{P}_i := \{\mathcal{P}_i^1 \cdots \mathcal{P}_i^K\} \leftarrow$ Flatten and pack with $\boldsymbol{g}_i^t$;
11 Decide masks $M_i^t$ for threshold $\zeta$ largest in $\mathcal{P}_i$;
12 for $M_i^t[l] \in M_i^t$ do
13     $\overline{\mathcal{P}}_i[l] \leftarrow \mathcal{P}_i[l] + \Delta_i$;
14     $\mathcal{C}_i^t \leftarrow$ Append $\boldsymbol{E}(\overline{\mathcal{P}}_i[l], pk)$;
15 Send $\mathcal{C}_i^t, M_i^t$ to the PS;
16 $\{r_{ij}\}_{j\in\mathcal{S}^t} \leftarrow$ Split $r_i$ into $|\mathcal{S}^t|$ shares by Eq. (4);
17 Exchange $r_{i,j}$ with client $j \in \mathcal{S}^t\backslash i$;
18 Exchange $\widetilde{r}_i = \sum_{j=1}^{|\mathcal{S}^t|} r_{j,i}$ with clients $j \in \mathcal{S}^t\backslash i$;
19 Recover $r$ from $\{\widetilde{r}_j\}_{j\in\mathcal{S}^t}$ by Eq. (5);
20 Compute blinding factor $R \leftarrow r + r_d$, and send it to
     unselected clients;
```

The PS calculates the probability of sketch *collision* to estimate the Jaccard similarity of two local updates denoted by $JS(g_i^{t-1}, g_i^t)$, i.e., $\Pr_{\mathcal{H}}(h_i^{t-1} = h_i^t) = JS(g_i^{t-1}, g_i^t)$. According to [41], lower similarity implying higher inference loss is likely to achieve better performance and thus should be assigned a larger aggregation weight p_i^t. That is,

$$p_i^t = \frac{\exp(-\beta \cdot JS(g_i^{t-1}, g_i^t))}{\sum_{j\in\mathcal{S}^t} \exp(-\beta \cdot JS(g_j^{t-1}, g_j^t))}, \quad (7)$$

where β is a positive number used to modify the exponential function's curve.

After receiving ciphertexts $\{\mathcal{C}_1^t, \dots, \mathcal{C}_{|\mathcal{S}^t|}^t\}$ and masks $\{M_1^t, \dots, M_{|\mathcal{S}^t|}^t\}$ from selected clients $\mathcal{S}^t$, the PS performs encrypted weighted aggregation to get global updates, i.e.,

$$\begin{aligned} \boldsymbol{E}(\boldsymbol{g}^{t+1}) &= \sum_{i\in\mathcal{S}^t} \boldsymbol{E}\left(p_i^t\right) \times \boldsymbol{E}\left(\boldsymbol{g}_i^t + \Delta_i\right) \\ &= \sum_{i\in\mathcal{S}^t} \boldsymbol{E}\left(p_i^t\right) \times \boldsymbol{E}\left(\boldsymbol{g}_i^t + (r_i + r_{d,i})/p_i^t\right) \end{aligned} \quad (8)$$

which $\boldsymbol{E}(p_i^t)$ is the encrypted weight assigned to client i. Building upon CKKS's homomorphic multiplication, the aggregation in (8) is equivalent to performing weighted aggregation on

Algorithm 3: Sketch-Based Client Selection.

```
Input: Clients $\mathcal{N}$, round $t$, cluster threshold $\gamma$, seed $s$,
       dimension $k$, random vectors $\mathcal{V}_c, \mathcal{V}_{s,i}$
Output: Selected clients $\mathcal{S}^t$
// PS
1:  Receive perturbed sketches $\{\widetilde{\boldsymbol{h}}_i^t\}_{i\in\mathcal{N}}$ from clients;
2:  $\overline{\boldsymbol{h}}_i^t \leftarrow \widetilde{\boldsymbol{h}}_i^t - \mathcal{V}_{s,i}, \forall i \in \mathcal{N}$;
3:  C $\leftarrow$ min($\mathbb{G}(\{\overline{\boldsymbol{h}}_1^t, \ldots, \overline{\boldsymbol{h}}_N^t\}), \gamma N$);
4:  Cluster $\{\overline{\boldsymbol{h}}_i^t\}_{i\in\mathcal{N}}$ into classes $\mathcal{A}^t := \{\mathcal{A}_1^t \cdots \mathcal{A}_C^t\}$;
5:  $\mathcal{S}^t \leftarrow$ Select clients from $\mathcal{A}^t$ by Eq. (12);
6:  Send $\mathcal{S}^t$ to all clients;
7:  Send weights $\{p_i^t\}_{i\in\mathcal{S}^t}$ to KDC based on Eq. (7);
    // Client $i \in \mathcal{N}$
8:  $\overline{\boldsymbol{g}_i^t} \leftarrow$ Flatten local updates $\boldsymbol{g}_i^t$;
9:  $\mathcal{M} \leftarrow$ Generate mapping matrix $\mathcal{G}(s, k)$;
10: $\boldsymbol{h}_i^t \leftarrow$ Sketching $\mathcal{H}(\overline{\boldsymbol{g}_i^t}, \mathcal{M})$;
11: $\widetilde{\boldsymbol{h}}_i^t \leftarrow \boldsymbol{h}_i^t + \mathcal{V}_c + \mathcal{V}_{s,i}$;
12: Send perturbed sketch $\widetilde{\boldsymbol{h}}_i^t$ to the PS;
```

plaintexts and encrypting the result, i.e.,

$$\begin{aligned}\boldsymbol{E}(\boldsymbol{g}^{t+1}) &= \boldsymbol{E}\left(\sum_{i\in\mathcal{S}^t} p_i^t \times \boldsymbol{g}_i^t + \sum_{i\in\mathcal{S}^t} r_i + \sum_{i\in\mathcal{S}^t} r_{d,i}\right)\\ &= \boldsymbol{E}\left(\sum\nolimits_{i\in\mathcal{S}^t} p_i^t \times \boldsymbol{g}_i^t + r + r_d\right)\\ &= \boldsymbol{E}\left(\sum\nolimits_{i\in\mathcal{S}^t} p_i^t \times \boldsymbol{g}_i^t + R\right) \qquad (9)\end{aligned}$$

where $R = r + r_d$ is referred to as global *obfuscating factor*. The encrypted global updates $\boldsymbol{C}^t = \sum_{i\in\mathcal{S}^t} \boldsymbol{E}(p_i^t) \cdot \mathcal{C}_i^t$ and mask $\boldsymbol{M}^t = \sum_{i\in\mathcal{S}^t} p_i^t \cdot M_i^t$ are subsequently dispatched to all clients for decryption and model update. We summarize how FedPHE conducts contribution-aware efficient secure aggregation in Algorithm 2.

C. Sketch-Based Straggler-Resistant Client Selection

We employ client selection to address the straggler issue arising from client heterogeneity. In practice, different clients might have similar or redundant model updates, causing *unnecessary* communication costs. Existing client selection approaches are largely conducted on plaintext, which contradicts the principles of privacy-preserving FL. To this end, we leverage the similarity of local updates to facilitate a sketch-based client selection in a privacy-preserving manner. Specifically, in each round, clients compute and send perturbed sketches of their model updates to the PS. After removing the personalized perturbations from sketches, the PS clusters similar sketches together and selects the fastest client from each cluster. The main steps are summarized in Algorithm 3.

Sketching local updates with perturbation: After decryption and model updates, each client i first flattens the local model updates g_i^t to a d element tensor $\overline{g_i^t}$ and then generates a $k \times d$ matrix called $\mathcal{M}$, which is made up of k d-dimensional vectors, using the shared seed s. By leveraging the LSH technique, the client achieves *dimensionality reduction* on $\overline{g_i^t}$ and obtains a sketch $\boldsymbol{h}_i^t$. In particular, LSH is a family $\mathcal{F}$ of functions $\mathcal{H}$: $\mathbb{R}^d \rightarrow \mathbb{S}$, with the property that if two inputs are similar in the original data space, they will also have a high similarity after being converted by the hash [42], [43]. For any two clients' model updates, g_m and g_n, any hash function h chosen uniformly at random from $\mathcal{F}$ should satisfy

- If $d(g_m, g_n) < \mathcal{R}$, then $\Pr_{\mathcal{H}}(h(g_m) = h(g_n)) \geq p_1$;
- If $d(g_m, g_n) \geq c\mathcal{R}$, then $\Pr_{\mathcal{H}}(h(g_m) = h(g_n)) \leq p_2$.

Here, c is an approximation ratio for the nearest neighbor search, $\mathcal{R}$ is the distance from the nearest neighbor, and p_1, p_2 denote the probabilities such that $p_1 > p_2$. This definition ensures that if g_m and g_n are close, they are hashed to the same bucket (collision) with a high probability ($\geq p_1$), whereas they are hashed to the same bucket with a low probability ($\leq p_2$). We exploit the properties of LSH functions to signify higher similarity during collisions when two inputs' hash codes are highly similar.

Accordingly, the flattened local update $\overline{g}_i^t$ are quantized into a binary matrix $\mathcal{M}_{\{0,1\}}$, where each element is assigned a value of 1 if it exceeds a threshold ϵ, and 0 otherwise. To compute the m-th element of sketch h_i^t, we iterate through each element in m-th row of $\mathcal{M}_{k\times d}$ and use each element's value as an index in $\mathcal{M}_{\{0,1\}}$ to locate the first occurrence of 1.If a 1 is found, the corresponding value is recorded in h_i^t. Despite the limited information involved in sketches, there is a risk of privacy leakage when sketches are transferred to the PS or intercepted by honest-but-curious clients. To prevent the PS from deducing sketches, each client adds a shared vector $\mathcal{V}_c$ to its sketch h_i^t, yielding a perturbed sketch $\overline{h}_i^t$. The addition of $\mathcal{V}_c$ is carefully calibrated to be minimal, ensuring that it does not interfere with the accuracy of operations such as Jaccard Similarity or clustering. Moreover, the security of h_i^t is maintained since the PS is not privy to $\mathcal{V}_c$. To further shield against potential interception of sketches, a personalized vector $\mathcal{V}_{s,i}$, exclusively known to the PS and client i, is also added into the sketch before transmission. The addition of perturbation terms is conducted within the finite cyclic additive group $\mathbb{Z}_d$. Consequently, even if the perturbed sketch $\overline{h}_i^t + \mathcal{V}_{s,i}$ is intercepted by honest-but-curious clients, they won't be able to learn any meaningful information in the absence of $\mathcal{V}_{s,i}$. An illustrative example of this secure perturbed sketching process is depicted in Fig. 6 and in practice $d \gg k$.

Clustering sketches: After receiving the sketches of local updates from clients, the PS removes personalized perturbations $\{\mathcal{V}_{s,i}\}_{i\in\mathcal{S}^t}$ and computes the number of clusters $C = \min(\mathbb{G}(\{\overline{h}_1^t, \ldots, \overline{h}_N^t\}), \gamma N)$, where $\mathbb{G}(\cdot)$ denotes the *gap statistic*, a standard technique to determine the optimal cluster number [44]. Gap statistic compares the actual intra-cluster variation to the expected values based on a null reference distribution, which is generated using *Monte Carlo* simulations. Here, γ is the threshold that limits the maximum number of clusters. For any given $k = 1, \ldots, k_{\max}$, the gap statistic is defined as

$$\mathbb{G}_n(k) = \mathbb{E}_n^*(\log W_k) - \log W_k, \qquad (10)$$

where W_k denotes the dispersion within the cluster, by comparing to its expectation $\mathbb{E}_n^*$ under a sample size n from the reference distribution. To correct the error in Monte

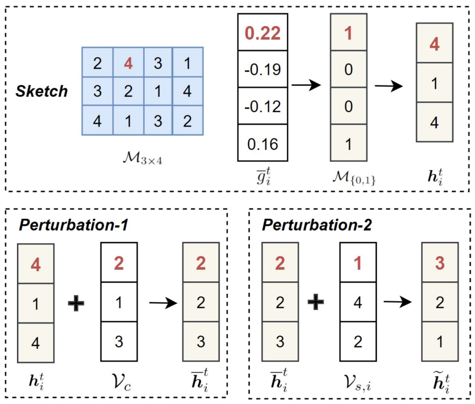


Fig. 6. A toy example of perturbed sketching with $\epsilon = 0.1$.

Carlo sampling, the correction factor s_k can be calculated from B copies of the reference datasets. Let $\overline{w} = \frac{1}{B}\sum_{b=1}^{B}\log\left(W_{kb*}\right)$. The standard deviation denoted by $sd(k)$ can be derived, i.e., $sd(k) = \sqrt{\frac{1}{B}\sum_{b=1}^{B}(\log W_{kb*} - \overline{w})^2}$. Define $s_k = sd(k) \times \sqrt{(1+B)/B}$. We finally choose the smallest k as the number of clusters such that

$$\mathbb{G}_k \geq \mathbb{G}_{k+1} - s_{k+1}. \tag{11}$$

Following that, clients can be clustered into C classes $\{\mathcal{A}_1^t \cdots \mathcal{A}_C^t\}$ by *K-means*, where those clients in the same class share similar sketches. Based on the assumption that LSH hashes similar input items into the same *buckets* with a high likelihood, similar sketches mean similar local updates.

Selecting clients: Instead of randomly selecting a client from a cluster, the PS prioritizes selecting one representative client having the ability to train *quickly*. Denote T_i^t as the order of client i's local update received by the PS in round t. Given the participation history $T_i^0, \ldots, T_i^{j-1}$, the priority $\mathbb{F}_i^t$ of being selected can be determined by

$$\mathbb{F}_i^t = \frac{1}{\alpha\delta_i^{t-1} + (1-\alpha)\times T_i^t}, \tag{12}$$

where $\alpha \in (0,1)$ is the influencing factor, and $\delta_i^{t-1} = \frac{1}{t}\sum_{j=0}^{t} T_i^j$ represents the *historical* engagement performance of client i. If the cluster consists of only one client, we directly add this client to the selection set $\mathcal{S}^t$.

Compared to traditional similarity determination techniques like cosine similarity, adopting the perturbed sketches of local updates as a cluster feature is not only communication-efficient but also privacy-preserving. This advantage becomes particularly pronounced when applied to more sophisticated machine learning models.

D. Security Analysis

In this section, we analyze the security of FedPHE. Our analysis is based on the assumption that KDC is completely trustworthy and dispatches keys/obfuscating factors to clients via secure channels. Generally, there are two main threats to clients' data. First, data leakage can occur on honest-but-curious PS during aggregation and client selection. Second, an honest-but-curious client may launch eavesdropping or collusion attacks. We next show that FedPHE is secure against both honest-but-curious PS and clients.

Theorem 1: Security against the **honest-but-curious PS**: In FedPHE, the honest-but-curious PS cannot infer any private information about clients' datasets.

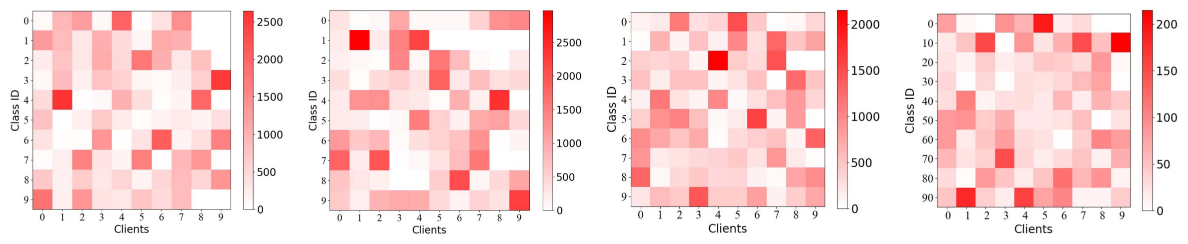
Proof: In FedPHE, the PS is responsible for *aggregation* and *client selection* in FedPHE. In the aggregation phase, the PS can only have access to ciphertexts since the local updates received from selected clients are encrypted via PHE before transmission. This prevents the PS from accessing any individual client's update during aggregation.

While in client selection phase, the perturbed sketch technique is employed for communication efficiency and privacy preservation. Specifically, the sketches are dependent on a seed s or mapping matrix $\mathcal{M}$, both of which are opaque to PS. Without the seed s, PS can only infer the matrix $\mathcal{M}$ through guesswork, which yet is ineffectual for launching model inversion attacks or deducing updates since guessing cannot reliably create a collision with a client's sketch. Additionally, PS cannot obtain any client's random vector $\mathcal{V}_c$, which is transmitted from KDC via a secure channel. Hence, the low-information nature of the sketches and the perturbation of shared vectors impede the reconstruction of local updates, thereby establishing a dual-layer protection mechanism. Although PS can learn the similarity of client sketches, it cannot recover any individual client's specific sketch or data distribution. Consequently, any reverse engineering attempts to extract information from sketches are destined to fail.

Therefore, the honest-but-curious PS cannot infer any particular client's dataset information. □

Theorem 2. Security against **honest-but-curious clients**: In FedPHE, an honest-but-curious client capable of launching eavesdropping attacks or colluding with up to $|\mathcal{S}^t| - 2$ other clients cannot infer any private information about other clients' datasets.

Proof. An honest-but-curious client j may attempt to eavesdrop and decrypt the ciphertexts of any other client i's updates, denoted as $\mathcal{C}_i^t = \boldsymbol{E}(\mathcal{P}_i[l] + \Delta_i, pk) = \boldsymbol{E}(\mathcal{P}_i[l] + (r_i + r_{d,i})/p_i^t, pk)$. However, the adversary cannot have access to client i's local updates due to the unawareness of the personalized obfuscating number r_i, obfuscating share $r_{d,i}$ and aggregation weight p_i^t. In particular, the aggregation weights $p_i^t, i \in \mathcal{S}^t$ are dynamically updated by PS in each round and remain undisclosed to the clients. As a result, even with access to the global model, the adversary cannot infer individual contributions or local updates. Moreover, even if the adversary can recover the obfuscating number r_i by intercepting all shares $r_{i,k}, k \in \mathcal{S}^t$ distributed to selected clients, the critical obfuscating share $r_{d,i}$ is securely stored in the KDC and is kept inaccessible. This ensures that clients cannot extract any other client's private data through eavesdropping on ciphertexts, thereby significantly reducing the risk of membership inference attacks. Similarly, if an honest-but-curious client attempts to eavesdrop on the sketch $\widetilde{\boldsymbol{h}}_i^t = \overline{\boldsymbol{h}}_i^t + \mathcal{V}_c + \mathcal{V}_{s,i}$ during the client selection phase, although the client possesses $\mathcal{V}_c$, he is unaware of the personalized perturbation vector $\mathcal{V}_{s,i}$, which is securely



(a) MNIST (b) FMNIST (c) CIFAR-10 (d) CIFAR-100

Fig. 7. Client data distributions under the non-IID setting ($\alpha = 1$), where color intensity indicates the data volume of each class.

transmitted via the KDC. Thus, the client is unable to obtain any private information by intercepting others' sketches.

Furthermore, consider a collusion attack scenario involving up to $|\mathcal{S}^t| - 2$ clients and the PS. Without loss of generality, assume that two selected clients i and j do not collude, while the remaining $|\mathcal{S}^t| - 2$ clients collude to try to deduce the local updates of i and j. During aggregation, these colluding clients can only obtain the sum of obfuscating numbers, i.e., $r_i + r_j = r - \sum_{k \in \mathcal{S}^t \setminus \{i,j\}} r_k$, and the sum of shares, i.e., $r_d - \sum_{k \in \mathcal{S}^t \setminus \{i,j\}} r_{d,k}$, pertaining to the two non-colluding clients. Even if these adversaries further gain access to the secret key sk and collude with the PS to learn the aggregation weights p_i^t and p_j^t, they still cannot determine the individual obfuscating numbers r_i or r_j and the corresponding shares $r_{d,i}$ or $r_{d,j}$. Without these specifics, the local updates for clients i and j remain securely protected. Consequently, the collusion attacks between the PS and up to $|\mathcal{S}^t| - 2$ clients are ineffective in FedPHE. Therefore, an honest-but-curious client cannot infer any private training data of other clients. □

IV. Performance Evaluation

In this section, we evaluate the performance of FedPHE, including contribution-aware efficient secure aggregation and sketch-based straggler-resistant client selection.

A. Evaluation Setup

Platform and parameters. Evaluations are conducted with 3 NVIDIA GeForce RTX 4090 GPUs using Pytorch. Consider a cross-silo FL scenario where $N = 10$ clients collaboratively train a model. We implement BFV and CKKS with *TenSEAL* [45], and their poly modulus degrees are set to 8192. To mimic the presence of stragglers, we randomly select 25% of clients as stragglers and introduce an artificial delay of $3 - 5$ rounds' training time. The batch size is $B = 64$ and the learning rate is $\eta = 1e - 3$ ($1e - 2$ for CIFAR-100). The packing size and the number of LSH hash functions are set to 4096 and 200, respectively.

Datasets and models: We evaluate the results on four real datasets: MNIST [46], FMNIST [47], CIFAR-10 and CIFAR-100 [48]. In particular, we partition MNIST and FMNIST into 60,000 training data and 10,000 test data. For CIFAR-10 and CIFAR-100, we use 50,000 and 10,000 images as the training data and test samples, respectively. We study the client heterogeneity of the Dirichlet Non-IID data setting (α=1), similar to [49]. The client data distributions under such non-iid setting are visualized in Fig. 7. We see that datasets are highly *imbalanced* with different clients holding unequal amounts of samples, which closely mirrors real-world data distributions. A straightforward LeNet-5 neural network architecture [46] is employed for MNIST. For FMNIST, a CNN model with 2 convolutional layers and 1 fully connected layer is utilized. The ResNet-20 and ResNet-32 models [50] are applied respectively to conduct experiments on the CIFAR-10 and CIFAR-100 datasets.

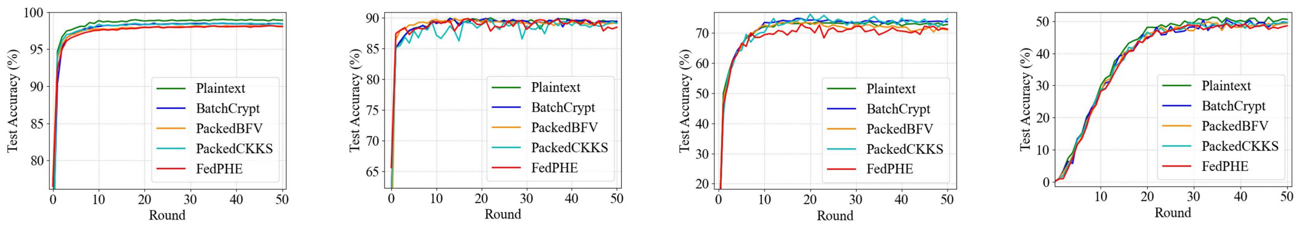
Baselines: To validate the proposed FedPHE, we introduce the following FL algorithms for comparison.

- *Plaintext:* an ideal upper bound for computation and communication overheads, where parameter transmission and aggregation are conducted in plaintext.
- *BatchCrypt:* Paillier-based PHE [17], where clients quantize first, then pack and encrypt the model updates, while the PS performs aggregation on the ciphertext.
- *PackedBFV:* BFV-based PHE [31], where model updates are quantized and weighted on the client side before encryption, as BFV only supports integer operations.
- *PackedCKKS:* CKKS-based PHE [32], which leverages the ciphertext multiplication of CKKS to facilitate encrypted weighted aggregation on the PS side.
- *FedAvg:* federated averaging [34], where the PS randomly selects the subset of clients for aggregation.
- *FLANP:* straggler-resilient FL with adaptive client selection [51], which starts the training process with faster clients and gradually involves slower clients once the accuracy of current participants' data is reached.

B. Evaluations on Efficient Secure Aggregation

We evaluate the efficiency of the proposed FedPHE by examining the test accuracy, network traffic and training time under different datasets, compared to the baselines, including plaintext training (no encryption), BatchCrypt, PackedBFV, and PackedCKKS. The experiments were conducted until reaching convergence.

Accuracy: Fig. 8 illustrates the training process of the global model on different datasets, i.e., MNIST, FMNIST, CIFAR-10 and CIFAR-100. Basically, the accuracy curves of the plaintext and other baselines almost overlap with each other, signifying that the PHE technique does not lead to a reduction in accuracy. While for FedPHE, the accuracy *fluctuates* within an acceptable



(a) MNIST (b) FashionMNIST (c) CIFAR-10 (d) CIFAR-100

Fig. 8. Accuracy versus global rounds of FedPHE and the baselines on different datasets.

TABLE III
NETWORK TRAFFIC, TEST ACCURACY, AND TRAINING TIME OF FEDPHE AND THE BASELINES ON DIFFERENT DATASETS

Dataset	Metric	Plaintext	BatchCrypt	PackedBFV	PackedCKKS	FedPHE
MNIST	Traffic (MB)	55	133	1996	1763	150
	Accuracy	96.64%	95.14%	95.13%	95.10%	95.15%
	Time (s)	622.84	3314.79	1371.73	1235.21	623.03
FMNIST	Traffic (MB)	53	103	1938	1371	78
	Accuracy	89.31%	87.72%	87.81%	87.26%	87.11%
	Time (s)	636.64	1690.23	750.68	923.88	690.65
CIFAR-10	Traffic (MB)	347	619	11330	9776	3815
	Accuracy	72.46%	71.19%	70.08%	72.64%	71.17%
	Time (s)	770.97	6845.01	1622.70	2863.77	1044.04
CIFAR-100	Traffic (MB)	906	1450	24128	18650	9057
	Accuracy	50.32%	49.55%	49.22%	49.50%	48.53%
	Time (s)	981.56	11006.16	2648.26	3594.59	2410.07

range of 0.26%–1.45%, which arises from pack-level sparsification and client selection.

Network traffic and training time: We present the network traffic and training time of FedPHE and the baselines on different datasets in Table III. We observe that FedPHE reduces the network footprint for MNIST, FMNIST, CIFAR-10 and CIFAR-100 by up to $11.75\times$, $17.58\times$, $2.56\times$, and $2.06\times$, respectively, compared to PackedCKKS. Moreover, it outperforms PackedBFV for $2.66-24.85\times$ across four datasets. It is worth noting that the ciphertext size is only $1.13\times$, $0.76\times$, $6.16\times$, and $6.25\times$ compared to the BatchCrypt. This indicates the efficiency of FedPHE in reducing the ciphertext generated by CKKS to the level of BatchCrypt encryption with Paillier. This achievement is truly remarkable. Additionally, the ciphertext size, which was previously in a "memory out" state as shown in Table I, has been reduced to only $1.47-10.99\times$ larger than the plaintext baseline, making FedPHE applicable to FL in practice. In conclusion, FedPHE achieves communication overhead *reduction* ranging from $1.32\times$ to $24.85\times$ compared to these baselines.

As shown in Table III, BatchCrypt requires $2.65-11.21\times$ more training time compared to plaintext. In contrast, FedPHE incurs only $1.00-2.46\times$ training time of the plaintext baseline, greatly enhancing the efficiency of model training. Furthermore, leveraging sparsification and client selection, FedPHE achieves a training *acceleration* of $2.45-6.56\times$. With an apt sparsification ratio, FedPHE does not adversely affect the trained model quality. Instead, it achieves significant compression while maintaining high performance.

TABLE IV
CONVERGENCE ROUNDS, ACCURACY, TRAINING TIME, AND NETWORK TRAFFIC OF DIFFERENT *SPARSIFICATION POLICIES* UNDER VARYING LEVELS OF NON-IIDNESS

α	Sparsification	Rounds	Accuracy	Time(s)	Traffic(MB)
1	✗	45	72.64%	2763.77	9756
	Random	50	65.99%	2595.50	7492
	Average	45	70.25%	1527.68	6744
	L2 norm	31	71.19%	1044.14	4645
5	✗	44	72.78%	2714.49	9538
	Random	49	67.05%	2537.20	7341
	Average	46	70.51%	1605.95	6894
	L2 norm	30	71.45%	1162.54	4495
10	✗	42	72.88%	2581.40	9110
	Random	45	66.34%	2355.27	6743
	Average	42	70.47%	1422.46	6292
	L2 norm	28	71.89%	938.14	4194

Sparsification policies: Table IV presents a comparison of three commonly used sparsification policies under varying non-IIDness levels for the MNIST dataset, provided sparsification ratio $\zeta = 70\%$. Obviously, FedPHE without sparsification

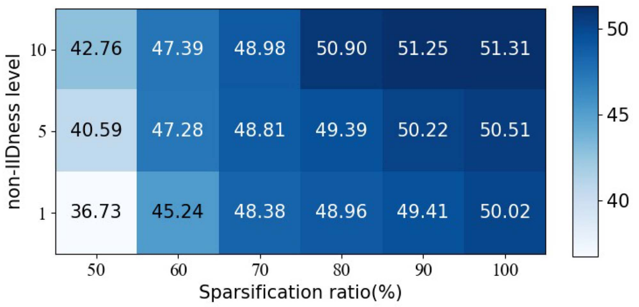


Fig. 9. Impact of sparsification ratios and non-IIDness levels on test accuracy across the CIFAR-100 dataset.

achieves the slowest convergence and highest network traffic. By randomly selecting ζ of packed updates, *random sparsification* reduces convergence time and communication volume by $1.06 - 1.10\times$ and $1.30 - 1.35\times$. This, however, results in a drastic accuracy drop of 6.65%, since essential data may be discarded indiscriminately. *Average-based sparsification*, which chooses the top-ζ packed updates by mean value, outperforms random sparsification in terms of reducing convergence time and network traffic, yet it still fails to precisely select crucial data, resulting in a 2.41% accuracy decline. In contrast, *L2 norm-based sparsification*, targeting the top-ζ packed updates by L2 norm magnitude, cuts down convergence time by $2.33 - 2.75\times$ and network traffic by $2.10 - 2.17\times$, while keeping the accuracy fluctuation within a modest 1.45%. By considering the cumulative impact of packed updates, especially the smaller ones, on model performance, this approach offers a more promising strategy to navigate the trade-off between convergence and accuracy, compared to solely relying on average values.

Impact of sparsification ratios and non-IIDness levels: Fig. 9 provides a visual analysis of how sparsification ratios and non-IIDness levels impact the test accuracy. For a given sparsification ratio ζ, the accuracy tends to increase as the sample distribution becomes more uniform (i.e., larger α). As expected, under a specific non-IID setting, an increase in sparsification ratio is consistently associated with improved accuracy. It is worth noting that setting ζ too low may lead to a drastic reduction in model accuracy, particularly when the sample distribution is highly imbalanced. This highlights the necessity of carefully calibrating the sparsification ratio to strike a balance between enhancing communication efficiency and maintaining satisfactory accuracy.

C. Evaluations on Sketch-Based Client Selection

We show FedPHE with client selection alone has superior performance over two baselines, i.e., FedAvg and FLANP.

Accuracy: Fig. 10 shows that FedPHE consistently outperforms the baselines in terms of test accuracy. FLANP exhibits faster convergence compared to FedAvg since it selects fewer stragglers to participate. Moreover, FedPHE achieves accelerated convergence compared to FedAvg and FLANP. This is because FedPHE employs sketch-based client selection to cherry-pick representative clients hosting diverse models and having the capability to train quickly.

TABLE V
CONVERGENCE ROUNDS, ACCURACY, TRAINING TIME, AND NETWORK TRAFFIC OF DIFFERENT *SELECTION POLICIES* UNDER VARYING LEVELS OF NON-IIDNESS

α	Selection	Rounds	Accuracy	Time(s)	Traffic(MB)
1	✗	49	95.15%	1737.21	1718
	Random	45	94.58%	1005.37	1197
	Weighted	35	95.14%	823.03	929
	Fastest	37	95.10%	654.49	986
5	✗	44	95.15%	1708.44	1543
	Random	39	95.03%	1182.41	1044
	Weighted	32	95.09%	1050.62	855
	Fastest	35	95.10%	829.77	940
10	✗	49	95.17%	1743.18	1718
	Random	40	94.98%	809.32	1142
	Weighted	34	95.16%	703.33	947
	Fastest	35	95.18%	589.74	978

Selection policies: After clustering similar sketches together, PS selects a representative client from each cluster. We evaluate different selection policies under varying non-IIDness levels for the MNIST dataset, as shown in Table V. Compared to the baseline (no selection), *random selection* reduces convergence time and network traffic by $1.44 - 2.15\times$ and $1.44 - 1.50\times$, respectively. However, randomly choosing clients may still involve some stragglers as participants. *Weighted selection*, which prioritizes clients with higher aggregation weights, achieves superior performance in reducing convergence time and communication overhead. Yet, indiscriminately selecting a client with the maximum weight may potentially select less responsive clients due to a lack of awareness of stragglers. In contrast, *fastest selection* cherry-picks the fastest client, effectively mitigating the straggler effect. This accelerates training by $2.06 - 2.96\times$ and reduces network traffic by $1.64 - 1.76\times$ compared to the baseline. Hence, it is effective for FedPHE to select the *fastest* client from each cluster to participate in the training.

Number of clusters: From Fig. 11(a), we can see that the cluster number fluctuates between 1 and 10, where the cluster threshold is set to $\gamma = 1$. A decrease in the cluster number suggests a higher similarity between local models. This means that similar sketches of local models are clustered together, resulting in a reduction in the cluster number. Throughout the process, the cluster number is dynamically determined based on the similarity of sketches sent by clients. There is no need for the PS to specify the exact number of clusters, making the model more robust.

Client selection efficiency: We record the number of normal clients and stragglers of FedPHE and the baselines. As depicted in Fig. 11(b) with $\gamma = 0.625$, we observed that the proportions of selected stragglers are 25%, 17%, 12% for FedAvg, FLANP and FedPHE, respectively. FedAvg randomly selects a subset of clients to participate, which reduces the overall number of

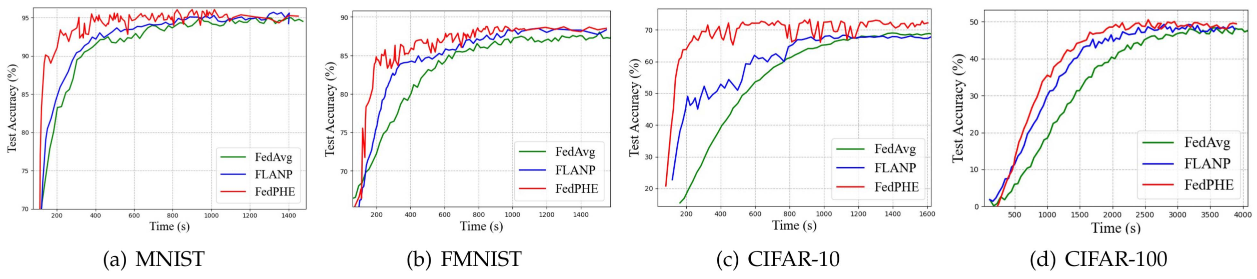


Fig. 10. Accuracy versus clock time of FedPHE and the baselines on different datasets.

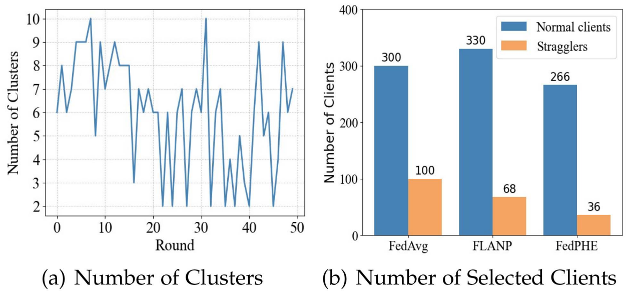


Fig. 11. Number of clusters in each round and number of selected clients of FedPHE and the baselines.

clients. However, many stragglers are still involved in the selection process. FLANP initially selects normal clients, but stragglers will join in the final rounds of training. In contrast, FedPHE can efficiently ensure the minimal inclusion of stragglers and decrease the overall number of clients without accuracy loss. It can reduce the straggler effect by up to $1.89 - 2.78\times$ compared to baselines. Our client selection scheme does not compromise model accuracy. That is, selecting a subset of representative clients to participate during aggregation can dramatically mitigate the straggler effect caused by client heterogeneity.

D. Ablation Study

We continue to validate FedPHE through the ablation study.

Sparsification ratios: Table VI illustrates the test accuracy, convergence time and network traffic with varying sparsification ratios across four different datasets. For MNIST and FMNIST datasets, employing a sparsification ratio of 10% is desirable as it reduces the network footprint by $8.48\times$ and $10.39\times$, respectively, with only negligible impact on model accuracy. While for CIFAR-10 and CIFAR-100 datasets, a 50% sparsification ratio leads to unacceptable accuracy losses of 12.62% and 13.06%, despite achieving high communication efficiency. In contrast, a more moderate 70% sparsification ratio appears to be feasible, reducing network traffic by $1.49\times$ and $1.43\times$, while maintaining a tolerable accuracy reduction of only 1.45%. These suggest that more complex datasets typically require a less aggressive sparsification approach to preserve satisfactory accuracy. Hence, the sparsification ratio should be carefully determined by accounting for dataset characteristics in order to strike a balance between efficiency and model performance.

TABLE VI
ABLATION RESULTS FOR VARYING SPARSIFICATION RATIOS

Dataset	ζ	Accuracy	Time(s)	Traffic(MB)
MNIST	10%	95.10%	1345.56	182
	20%	95.13%	1600.85	402
	100%	95.14%	1730.89	1543
FMNIST	10%	87.5%	903.81	132
	20%	87.53%	1258.42	305
	100%	87.9%	1335.21	1371
CIFAR-10	50%	60.02%	1393.23	4945
	70%	71.19%	2469.59	6556
	100%	72.64%	2763.77	9756
CIFAR-100	50%	36.51%	1907.52	9851
	70%	48.78%	3055.26	13071
	100%	49.57%	3864.72	18651

TABLE VII
ABLATION RESULTS FOR CLIENT SELECTION

Dataset	Client selection	Accuracy	Time(s)	Traffic(MB)
MNIST	✗	95.2%	1937.21	1718
	✓	95.1%	632.03	929
FMNIST	✗	88.2%	1335.21	1371
	✓	87.9%	739.04	861
CIFAR-10	✗	72.6%	3063.77	9756
	✓	72.5%	1244.04	6771
CIFAR-100	✗	49.5%	3864.72	18651
	✓	48.8%	2790.99	14547

Client selection: Table VII presents the performance comparisons with or without client selection across four different datasets. Notably, our client selection scheme enables a $1.38 - 3.07\times$ faster convergence while incurring a negligible maximum accuracy decline of only 0.7%. Also, by excluding stragglers during aggregation, it achieves a remarkable reduction in network traffic by $1.28 - 1.85\times$. These results corroborate the effectiveness of our proposed selection scheme in mitigating the negative effects caused by stragglers, making it a favorable choice for efficient PHE-based FL training.

V. Related Work

We briefly survey the related works below.

HE-based FL: HE is a widely used privacy-preserving technique for FL to avoid privacy leakage during aggregation [10], [52], yet leads to significant inefficiency in computation and communication. To address this, Zhang et al. design BatchCrypt to achieve a substantial decrease in encryption overhead and ciphertext volume by packing multiple plaintexts into a long integer [17]. Jiang et al. propose FLASHE, an additively symmetric HE in double masking to address the compatibility issues with sparsification [33]. Nevertheless, these methods rely on Paillier and fail to support weighted aggregation on ciphertexts. There is still ample scope for improving communication efficiency. In a similar vein, Smart et al. design a ciphertext-packing technique based on polynomial-CRT [15], and Brakerski et al. further extend SIMD to the standard LWE to achieve nearly optimal homomorphic evaluation [14]. However, existing HE solutions either suffer from the risk of ciphertexts being decrypted by honest-but-curious clients, or largely neglect the intrinsic client heterogeneity in a cross-silo FL.

HE-based secure aggregation: Traditional HE schemes mostly assume all clients share a secret key, making the system suffering from the risk of ciphertexts being intercepted and decrypted by honest-but-curious clients. To mitigate these security issues, various approaches have been proposed. Zheng et al. present a lightweight encryption and aggregation protocol that enables clients to submit obfuscated updates [24]. Cai et al. propose SecFed, which adopts multi-key HE to preserve client privacy and delegates some operations to TEE [26]. Shi et al. develop a secure aggregation framework built upon cryptographic schemes, ensuring aggregator obliviousness [25]. Despite these advances, current solutions remain resource-intensive for practical FL applications.

Weighted aggregation: A key challenge posed to FL is heterogeneity in the clients' local datasets and computation speeds, which drastically slows down the training process. Many efforts have been devoted to addressing such challenge. Among them, weighted aggregation is a promising technique to accelerate convergence. Zeng et al. present a contribution-aware aggregation scheme, considering higher loss values are indicative of more substantial performance improvements [41]. Wu et al. adaptively assign aggregation weights based on clients' contributions, measured by the angle between local and global gradient vectors [53]. Deng et al. propose FAIR to quantify each client's learning quality and automatically determine the optimal weights to enhance the global model quality [22]. Nonetheless, existing aggregation solutions are mainly conducted on plaintexts, rendering them inapplicable in HE-based FL scenarios.

Client selection: Client selection is widely adopted to accelerate convergence and mitigate the straggler effect [10]. FedAvg [34] employs random selection, acting as a common and general setting in FL. Reisizadeh et al. propose FLANP to start training by exchanging models with fast clients and gradually include slower clients over time [51]. Fraboni et al. introduce clustered sampling based on similarity for client selection. However, directly transmitting gradients to the PS raises privacy concerns and substantial communication overheads [35]. Kollias et al. utilize the sketches of local models to select clients that are similar to ours at a high level [54]. However, it can not be applied to HE-based FL since after encrypting with HE, the PS cannot calculate the sketch of the global model based on the ciphertext. In contrast, FedPHE conducts sketch-based client selection to cherry-pick representative clients that host diverse models with fast training capability, greatly reducing the communication overheads, without sacrificing model accuracy.

Differences from the prior work: Our prior work [1] primarily addresses client heterogeneity, a challenge largely overlooked in existing PHE-based FL methods. In contrast, FedPHE develops a more secure PHE-based FL framework that mitigates security threats from ciphertexts being intercepted and decrypted by honest-but-curious clients. Specifically, it adopts a more realistic threat model that accounts not only for privacy leakage during aggregation but also for clients eavesdropping or colluding to access specific local updates. A secret sharing–based obfuscating technique is designed to ensure encrypted local updates are transmitted and aggregated in the obfuscated form, thereby enhancing privacy. Additionally, FedPHE adopts a perturbed sketch-based client selection to further mitigate the potential privacy leakage from sketching process. Furthermore, a rigorous security analysis is provided to demonstrate that neither the honest-but-curious PS nor an honest-but-curious client capable of launching eavesdropping or collusion attacks can deduce private information about other clients' datasets.

VI. Conclusion

In this paper, we present FedPHE, a secure and efficient PHE-based FL framework to tackle challenges associated with security threats and client heterogeneity. By adopting CKKS-based PHE with sparsification and obfuscating, FedPHE achieves efficient secure aggregation that allows clients to only provide obscured encrypted updates while PS can still perform the aggregation by accounting for contributions of local updates to the global model. To mitigate the straggler effect posed by client heterogeneity, a perturbed sketch-based client selection is conducted to cluster similar clients together and then cherry-pick the fastest client from each cluster in a communication-efficient and privacy-preserving manner. We provide rigorous security analysis for FedPHE and verify its efficiency through extensive experiments.

Although FedPHE is proven secure against honest-but-curious adversaries, our method relies on a trusted KDC, which is widely adopted but may be difficult to guarantee in practical applications. In future work, we will focus on exploring distributed key generation methods that eliminate the need of KDC while improving communication and computation efficiency.

References

[1] N. Yan et al., "Efficient and straggler-resistant homomorphic encryption for heterogeneous federated learning," in *Proc. IEEE Conf. Comput. Commun.*, 2024, pp. 791–800.

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[3] Y. Huang et al., “Personalized cross-silo federated learning on non-IID data,” in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 7865–7873.
[4] P. Kairouz et al., “Advances and open problems in federated learning,” *Found. Trends Mach. Learn.*, vol. 14, no. 1/2, pp. 1–210, 2021.
[5] Y. Aono et al., “Privacy-preserving deep learning via additively homomorphic encryption,” *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1333–1345, May 2018.
[6] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 14774–14784.
[7] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, “See through gradients: Image batch recovery via gradinversion,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 16337–16346.
[8] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, “Inverting gradients-how easy is it to break privacy in federated learning?,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 16937–16947.
[9] K. Wei et al., “Federated learning with differential privacy: Algorithms and performance analysis,” *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, 2020.
[10] S. Zhang, Z. Li, Q. Chen, W. Zheng, J. Leng, and M. Guo, “Dubhe: Towards data unbiasedness with homomorphic encryption in federated learning client selection,” in *Proc. Int. Conf. Parallel Process.*, 2021, pp. 1–10.
[11] A. Li et al., “FedSDG-FS: Efficient and secure feature selection for vertical federated learning,” in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.
[12] N. Jovanovic, M. Fischer, S. Steffen, and M. Vechev, “Private and reliable neural network inference,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2022, pp. 1663–1677.
[13] X. Yang, J. Chen, K. He, H. Bai, C. Wu, and R. Du, “Efficient privacy-preserving inference outsourcing for convolutional neural networks,” *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 4815–4829, 2023.
[14] Z. Brakerski, C. Gentry, and S. Halevi, “Packed ciphertexts in LWE-based homomorphic encryption,” in *Proc. Int. Workshop Public Key Cryptography*, 2013, pp. 1–13.
[15] N. P. Smart and F. Vercauteren, “Fully homomorphic SIMD operations,” *Des. Codes Cryptography*, vol. 71, pp. 57–81, 2014.
[16] T. Ge and S. Zdonik, “Answering aggregation queries in a secure system model,” in *Proc. Int. Conf. Very Large Data Bases*, 2007, pp. 519–530.
[17] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, “BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning,” in *Proc. USENIX Conf. Usenix Annu. Tech. Conf.*, Jul. 2020, pp. 493–506.
[18] B. Luo, W. Xiao, S. Wang, J. Huang, and L. Tassiulas, “Tackling system and statistical heterogeneity for federated learning with adaptive client sampling,” in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1739–1748.
[19] X. Wang, Y. Chen, Y. Li, X. Liao, H. Jin, and B. Li, “FedMoS: Taming client drift in federated learning with double momentum and adaptive selection,” in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.
[20] C. Chen, W. Wang, and B. Li, “Round-robin synchronization: Mitigating communication bottlenecks in parameter servers,” in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 532–540.
[21] C. Chen et al., “Communication-efficient federated learning with adaptive parameter freezing,” in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2021, pp. 1–11.
[22] Y. Deng et al., “Improving federated learning with quality-aware user incentive and auto-weighted model aggregation,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4515–4529, Dec. 2022.
[23] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, “Oort: Efficient federated learning via guided participant selection,” in *Proc. USENIX Symp. Operating Syst. Des. Implementation*, 2021, pp. 19–35.
[24] Y. Zheng, S. Lai, Y. Liu, X. Yuan, X. Yi, and C. Wang, “Aggregation service for federated learning: An efficient, secure, and more resilient realization,” *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 988–1001, Mar./Apr. 2023.
[25] E. Shi, H. Chan, E. Rieffel, R. Chow, and D. Song, “Privacy-preserving aggregation of time-series data,” in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2011, pp. 1–17.
[26] Y. Cai, W. Ding, Y. Xiao, Z. Yan, X. Liu, and Z. Wan, “SecFed: A secure and efficient federated learning based on multi-key homomorphic encryption,” *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 4, pp. 3817–3833, Jul./Aug. 2024.
[27] A. Ali et al., “Communication–computation trade-offs in PIR,” in *Proc. USENIX Secur. Symp.*, 2021, pp. 1811–1828.
[28] C. Fontaine and F. Galand, “A survey of homomorphic encryption for nonspecialists,” *EURASIP J. Inf. Secur.*, vol. 2007, pp. 1–10, 2007.
[29] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, “A survey on homomorphic encryption schemes: Theory and implementation,” *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1–35, 2018.
[30] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.
[31] J. Fan and F. Vercauteren, “Somewhat practical fully homomorphic encryption,” *Cryptol. ePrint Arch.*, 2012.
[32] 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.
[33] Z. Jiang, W. Wang, and Y. Liu, “FLASHE: Additively symmetric homomorphic encryption for cross-silo federated learning,” 2021, *arXiv:2109.00675*.
[34] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[35] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi, “Clustered sampling: Low-variance and improved representativity for clients selection in federated learning,” in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 3407–3416.
[36] X. Hu et al., “STYX: A hierarchical key management system for elastic content delivery networks on public clouds,” *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 2, pp. 843–857, Mar./Apr. 2021.
[37] J. Wang, R. Zhang, J. Li, Y. Xiao, and H. Ma, “SeUpdate: Secure encrypted data update for multi-user environments,” *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 6, pp. 3592–3606, Nov./Dec. 2022.
[38] J. Wu, W. Zhang, and F. Luo, “ESAFL: Efficient secure additively homomorphic encryption for cross-silo federated learning,” *IEEE Trans. Dependable Secure Comput.*, early access, Feb. 2025, doi: 10.1109/TDSC.2025.3541612.
[39] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, “Robust and communication-efficient federated learning from non-IID data,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.
[40] B. Neyshabur and N. Srebro, “On symmetric and asymmetric LSHs for inner product search,” in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1926–1934.
[41] H. Zeng, T. Zhou, Y. Guo, Z. Cai, and F. Liu, “FedCav: Contribution-aware model aggregation on distributed heterogeneous data in federated learning,” in *Proc. Int. Conf. Parallel Process.*, 2021, pp. 1–10.
[42] M. Slaney and M. Casey, “Locality-sensitive hashing for finding nearest neighbors [lecture notes],” *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 128–131, Mar. 2008.
[43] B. Kulis and K. Grauman, “Kernelized locality-sensitive hashing for scalable image search,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 2130–2137.
[44] R. Tibshirani, G. Walther, and T. Hastie, “Estimating the number of clusters in a data set via the gap statistic,” *J. Roy. Stat. Soc.: Ser. B (Statist. Methodol.)*, vol. 63, no. 2, pp. 411–423, 2001.
[45] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, “TenSEAL: A library for encrypted tensor operations using homomorphic encryption,” in *Proc. Int. Conf. Learn. Representations Workshop Distrib. Private Mach. Learn.*, 2021, *arXiv:2104.03152*.
[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[47] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms,” 2017, *arXiv:1708.07747*.
[48] A. Krizhevsky et al., “Learning multiple layers of features from tiny images,” Tech. Rep., Univ. Toronto, 2009.
[49] T.-M. H. Hsu, H. Qi, and M. Brown, “Measuring the effects of non-identical data distribution for federated visual classification,” in *NeurIPS Workshop Federated Learn.*, 2019.
[50] 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.
[51] A. Reisizadeh, I. Tziotis, H. Hassani, A. Mokhtari, and R. Pedarsani, “Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity,” *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 2, pp. 197–205, Jun. 2022.
[52] X. Feng, H. Liu, H. Yang, Q. Xie, and L. Wang, “Batch-aggregate: Efficient aggregation for private federated learning in VANETs,” *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 5, pp. 4939–4952, Sep./Oct. 2024.

[53] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 4, pp. 1078–1088, Dec. 2021.

[54] G. Kollias, T. Salonidis, and S. Wang, "Sketch to skip and select: Communication efficient federated learning using locality sensitive hashing," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 72–83.

Yuqing Li (Member, IEEE) received the PhD degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2019. She is currently an associate professor with the School of Cyber Science and Engineering, Wuhan University, Wuhan, China. From 2019 to 2020, she was a postdoctoral fellow with the Hong Kong University of Science and Technology. From 2020 to 2022, she was a researcher with Huawei Hong Kong Research Center. Her research interests include AI security and privacy protection.

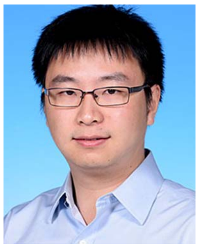
Nan Yan received the BS degree from the School of Cyber Science and Engineering, Shandong University, Jinan, China, in 2023. He is currently working toward the MS degree with the School of Cyber Science and Engineering, Wuhan University, Wuhan, China. His research interests include federated learning, and privacy-preserving computing.

Jing Chen (Senior Member, IEEE) received the PhD degree in computer science from the Huazhong University of Science and Technology, Wuhan, China. Since 2015, he has been a full professor with Wuhan University. He has authored or coauthored more than 100 research papers in many international journals and conferences, such as *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Parallel and Distributed Systems*, *USENIX Security*, ACM CCS, and IEEE INFOCOM. His research interests include computer science, network security, and cloud security.

Xiong Wang (Member, IEEE) received the PhD degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2019. From 2019 to 2021, he was a postdoctoral fellow with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. He is currently an associate professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His research interests include machine learning system, LLM training, and inference.

Jianan Hong (Member, IEEE) received the PhD degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2018. From 2018 to 2021, he was a research engineer with Huawei Shanghai Research Institute, China. He is currently an associated researcher with The School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include secure cloud computing and privacy preserving authentication.

Kun He (Member, IEEE) received the PhD degree in computer science from the Computer School, Wuhan University, Wuhan, China. He is currently an associate professor with Wuhan University, Wuhan. He has authored or coauthred more than 30 research papers in many international journals and conferences, such as *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *USENIX Security*, ACM CCS, and IEEE S&P. His research interests include cryptography, network security, mobile computing, and cloud computing.



Wei Wang (Member, IEEE) received the BEng and MEng degrees from the Department of Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2007 and 2010, respectively, and the PhD degree from the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, in 2015. Since 2015, he has been with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology (HKUST), where he is currently an associate professor. He is also affiliated with the HKUST Big Data Institute. His research interests include the broad area of distributed systems, with focus on serverless computing, machine learning systems, and cloud resource management. His research was the recipient of the Best Paper Runner Up awards of IEEE ICDCS 2021 and USENIX ICAC 2013.

Bo Li (Fellow, IEEE) received the BEng degree *(summa cum laude)* in computer science from Tsinghua University, Beijing, China, and the PhD degree in electrical and computer engineering from the University of Massachusetts at Amherst, Amherst, MA, USA. From 2010 to 2016, he has held a Cheung Kong visiting chair professor with Shanghai Jiao Tong University, and was the chief technical advisor with ChinaCache Corporation (NASDAQ:CCIH), one of the world leading CDN service providers. He is currently a chair professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. He has also made pioneering contributions in multimedia communications and the Internet video broadcast, in particular Coolstreaming system, which was credited as first large scale Peer-to-Peer live video streaming system in the world. It attracted significant attention from both industry and academia and was the recipient of the Test-of-Time Best Paper Award from IEEE INFOCOM (2015). He was also the recipient of the eight Best Paper awards including IEEE INFOCOM (2021).