Efficient and Straggler-Resistant Homomorphic Encryption for Heterogeneous Federated Learning

Nan Yan*, Yuqing Li*, Jing Chen*, Xiong Wang†, Jianan Hong‡, Kun He*, Wei Wang§
* School of Cyber Science and Engineering, Wuhan University, Wuhan, China
† School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
‡ School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
§ Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong
*{nanyan, li.yuqing, chenjing, hekun}@whu.edu.cn, †xiongwang@hust.edu.cn, ‡hongjn@sjtu.edu.cn, §weiwa@cse.ust.hk

IEEE INFOCOM 2024 - IEEE Conference on Computer Communications | 979-8-3503-8350-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/INFOCOM52122.2024.10621440

***Abstract*—Cross-silo federated learning (FL) enables multiple institutions (clients) to collaboratively build a global model without sharing their 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 does not consider the *heterogeneity* among different clients, an intrinsic problem in cross-silo FL, often resulting in undermined training efficiency with slow convergence and stragglers. In this work, we propose FedPHE, an efficiently packed homomorphically encrypted FL framework with secure weighted aggregation and client selection to tackle the heterogeneity problem. Specifically, using CKKS with sparsification, FedPHE can achieve efficient *encrypted weighted aggregation* by accounting for *contributions* of local updates to the global model. To mitigate the straggler effect, we devise a *sketching-based client selection* scheme to cherry-pick representative clients with *heterogeneous models* and *computing capabilities*. We show, through rigorous security analysis and extensive experiments, that FedPHE can efficiently safeguard clients' privacy, achieve a training speedup of $1.85 - 4.44\times$, cut the communication overhead by $1.24 - 22.62\times$, and reduce the straggler effect by up to $1.71 - 2.39\times$.**

I. Introduction

Cross-silo federated learning (FL) [1, 2] 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 [3, 4]. 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 [5–7].

This research was supported in part by the National Key R&D Program of China under grant No. 2022YFB3102100, the Fundamental Research Funds for the Central Universities under grant No. 2042023kf0120, the National Natural Science Foundation of China under grants No. 62302343, 62202185, 62076187, 62172303, the Guangdong Basic and Applied Basic Research Foundation under grant No. 2022A1515110396, the Hubei Provincial Natural Science Foundation under grant No. 2022CFB611, and a RGC RIF grant under the contract R6021-20. (Corresponding author: Yuqing Li.)

To avoid *privacy leakage* during aggregation, many privacy-preserving techniques have been employed for FL [8–10]. Among them, *homomorphic encryption* (HE) is particularly attractive to cross-silo FL, as it provides stronger privacy guarantees without compromising the learning accuracy [11–13]. With HE, gradient 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 [14–16]. A promising approach to address this problem is *packed* HE (PHE), which packs and encrypts multiple plaintext values into a single ciphertext [17]. By facilitating parallel encryption/decryption operations on multiple plaintexts, PHE dramatically reduces the encryption and communication overheads.

Though effective, existing PHE solutions largely ignore the *heterogeneity* of participating clients [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 (known as 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 (known as 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. Without addressing the client heterogeneity problem, the power of PHE cannot be fully unleashed.

A large body of works have been proposed to tackle the client heterogeneity issues for expediting training convergence and mitigating the straggler impact [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. This, however, is challenging due to the following reasons.

First, existing HE solutions [14] 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 the amount of local data on a client does not reflect its potential *contribution* to the global model. Even with knowledge of clients' contributions, the challenge remains in assigning appropriate aggregation weights for each client. Besides, clients may misreport weights, leading to biasing the global model towards their local training. As such, it is desired to employ *homomorphic multiplication* on ciphertext for weighted aggregation on the server side. This introduces potential *communication bottlenecks* as homomorphic multiplication typically requires a large ciphertext space for encryption [24]. Second, 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. Therefore, we have to carefully navigate the tradeoff between *security* and *efficiency* in client selection.

In this paper, we propose FedPHE, a secure and efficient cross-silo FL framework with PHE to tackle client heterogeneity. FedPHE employs a *contribution-aware* weighted aggregation scheme using the 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. This enhances the model's ability to quickly incorporate new knowledge, thereby 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 mitigate the straggler effect, we devise a *sketching-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 models using the sketching techniques. Similar clients are clustered together and only the fastest client is selected from each cluster. Adopting the sketches of local models as a cluster feature is not only *communication-efficient* but also *privacy-preserving* as it maps high-dimensional model updates to a lower dimension through entry hashing. 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, an efficient and *straggler-resistant* homomorphically encrypted FL framework for heterogeneous clients. To our knowledge, this is the first attempt that enables *secure* client selection and weighted aggregation to effectively address the challenges caused by client *heterogeneity*, 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. A pack-level sparsification approach is implemented to optimize data transfer *efficiency*, addressing the issue of increased ciphertext size using vanilla CKKS.
- FedPHE leverages 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.
- Extensive experiments on real-world datasets show that compared to the state-of-the-art approaches, FedPHE accelerates the training speed by $1.85 - 4.44\times$, cuts the communication overhead by $1.24 - 22.62\times$, and mitigates the straggler effect by $1.71 - 2.39\times$, with a slight degradation of model accuracy (1.58% 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 $\mathcal{N} = \{1, \cdots, N\}$ 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

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

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

$$\min_{\boldsymbol{w}} \mathcal{F}(\boldsymbol{w}) \triangleq \min_{\boldsymbol{w}} \sum\nolimits_{i\in\mathcal{N}} p_i f_i(\boldsymbol{w}), \tag{2}$$

where $\mathcal{F}(\boldsymbol{w})$ 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 Eq. (2), clients perform synchronous update 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 exploit clients' updates to infer their private information even when the training data are kept locally [5–7].

B. Homomorphic Encryption

Homomorphic encryption (HE) is a powerful cryptographic primitive that enables computations to be performed directly on the encrypted data without decrypting them in advance [25, 26]. HE ensures that the calculations performed on ciphertexts, when decrypted, give the 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 [27], BFV [28], and CKKS [29], 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 [14–16, 30]. Among them, a promising approach is PHE which packs and encrypts multiple plaintext values into a single ciphertext, thus allowing for parallel encryption/decryption operations. However, current (packed) homomorphically encrypted FL solutions largely ignore the intrinsic *client heterogeneity* in a cross-silo FL system, 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 model, a phenomenon known as the *statistical heterogeneity*. In this case, simply performing unweighted aggregation, i.e., $\boldsymbol{w}^{t+1} = \sum_{i\in\mathcal{N}} \frac{1}{N} \boldsymbol{w}_i^t$ where $p_i = \frac{1}{N}$, results in an undesirable *bias* to the global model that hinders 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 [31] as an example, the weighted aggregation based on data size is performed to minimize the loss in Eq. (2), i.e.,

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

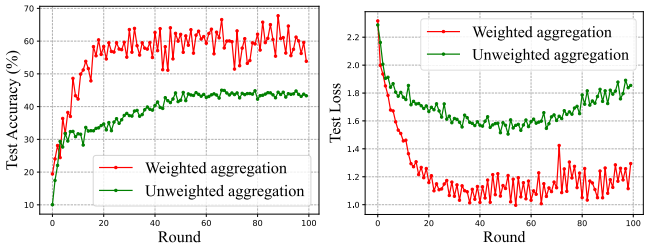


Fig. 1: Comparison results of weighted and unweighted aggregation.

TABLE I: Comparison results of different (packed) HE schemes.

Plaintext size	(Packed) HE scheme	Ciphertext size	Encryption time (s)	Decryption time (s)
109.89KB	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

where $\boldsymbol{w}_i^t$ and $\boldsymbol{w}^t$ are the local and the global models, respectively. To demonstrate 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 solutions [14] 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.

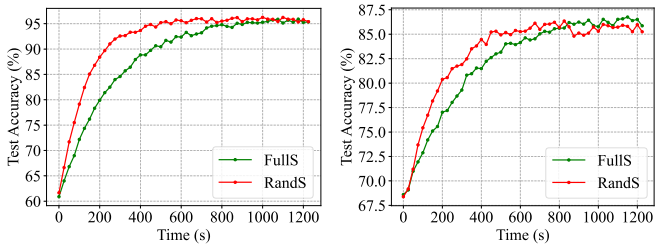
The need for efficient encrypted aggregation. HE-based FL methods offer strong privacy guarantees, albeit at the expense of efficiency. Table I shows the comparison results of Paillier, BFV, and CKKS as well as their packed implementation. Specifically, Paillier generates a ciphertext close to $205\times$ 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, and $211.3\times$ for BFV, $42.3\times$ for CKKS.

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

The need for client selection. One possible remedy for

TABLE II: Breakdown of training iteration time for normal clients and stragglers.

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



(a) MNIST (b) FashionMNIST

Fig. 2: Comparison results of **Full**Selection and **Rand**om**S**election.

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 [32]. 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, an efficient homomorphically encrypted FL framework designed for addressing client heterogeneity. We begin with system overview and then elaborate on how FedPHE co-designs secure weighted aggregation and client selection, followed by its security analysis.

A. Overview

Our design goal is to develop a secure and efficient FL framework for heterogeneous clients. Specifically, the following desirable objectives should be achieved.

- *Privacy Protection.* It should protect clients' privacy during aggregation. That is, the PS cannot reveal any sensitive information pertaining to individual clients, while clients cannot access any private data about others. We assume that the HE *key-pair* is shared among clients through a secure channel, and the PS does *not collude* with any client [14, 33].
- *Efficiency.* It is expected to be efficient for encrypting and transferring local models in each round since high computation and communication overheads make HE difficult to implement in practice.
- *Straggler Resistance.* It should effectively mitigate the negative impact of stragglers under client heterogeneity, accelerating the training process without sacrificing model accuracy.

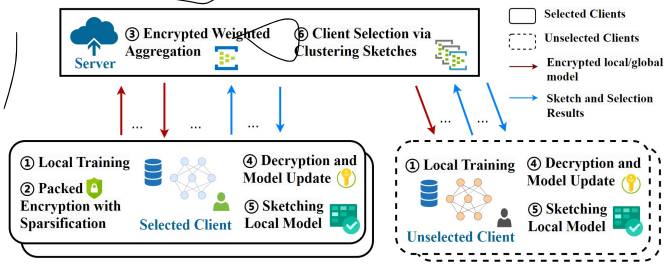


Fig. 3: A snapshot of FedPHE architecture.

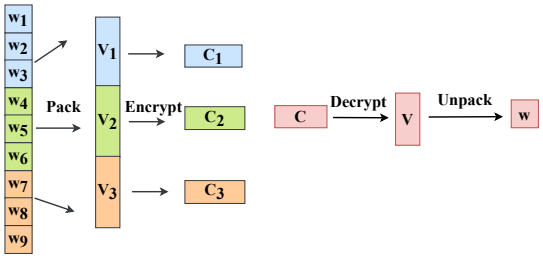


Fig. 4: Illustration of CKKS Packing Mode.

We propose FedPHE to attain these three objectives by jointly designing secure weighted aggregation and client selection to tackle the client heterogeneity challenges. Fig. 3 shows the main architecture, where FedPHE proceeds in rounds of communication as described below.

① *Local Training*. At the beginning of each round t, every client $i \in \mathcal{N}$ runs E steps of local stochastic gradient descent in Eq. (2) to compute the local model $\boldsymbol{w}_i^t$;

② *Packed Encryption with Sparsification*. Any selected client $i \in \mathcal{S}^t$ flattens and packs the local model $\boldsymbol{w}_i^t$ into $\{\mathcal{P}_i^1, \cdots, \mathcal{P}_i^K\}$, and then performs *sparsification* at the pack granularity; After that, it encrypts the sparse packed local model, and finally sends the ciphertext C_i^t as well as mask M_i^t to the PS for aggregation;

③ *Encrypted Weighted Aggregation*. The PS aggregates the received encrypted local models along with the encrypted weights, then computes and dispatches the new encrypted global model $\boldsymbol{C}^t$ and mask $\boldsymbol{M}^t$ to all clients;

④ *Decryption and Model Update*. Every client decrypts and unpacks the global ciphertext to obtain the global model $\boldsymbol{w}^t$, and then updates the local model $\boldsymbol{w}_{i,0}^t = \boldsymbol{w}^t$;

⑤ *Sketching Local Model*. Then each client $i \in \mathcal{N}$ computes and sends the sketch of local model h_i^t to the PS;

⑥ *Client Selection via Clustering Sketches*. The PS selects a subset $\mathcal{S}^{t+1}$ out of N clients as participants and derives the aggregation weight p_i^{t+1} based on its contribution.

The details of FedPHE are illustrated in Alg. 1. In each global round t, only selected clients perform PHE and sparsification, and then transmit the encrypted model updates and masks (lines 14–16) to the PS. Given clients' contributions, the PS carries out encrypted weighted aggregation and sends the results to all clients (lines 3-5). After decryption and model updates, clients send *sketches* of their local models to PS (lines 17-19). The PS then selects a subset of clients as participants $\mathcal{S}^{t+1}$ for the next round (lines 6–8).

Algorithm 1: FedPHE

Input: Clients $\mathcal{N}$, global round T, local steps E, learning rate η

Output: Global model $\boldsymbol{w}^T$

Initialize models $\{\boldsymbol{w}_i\}_{\mathcal{N}}$ and selected clients $\mathcal{S}^0 \leftarrow \mathcal{N}$;

// **Server**

for *each round* $t \in \{0, \cdots, T-1\}$ **do**

  Receive encrypted local models $\mathcal{C}_i^t$ and masks M_i^t from selected clients $i \in \mathcal{S}^t$;

  $\boldsymbol{C}^t, \boldsymbol{M}^t \leftarrow$ Run weighted aggregation by Alg. 2;

  Dispatch $\boldsymbol{C}^t$ and $\boldsymbol{M}^t$ to all clients;

  Receive sketches $\{h_i^t\}_{i \in \mathcal{N}}$ of clients' local models;

  $\mathcal{S}^{t+1} \leftarrow$ Run client selection by Alg. 3;

  Send $\mathcal{S}^{t+1}$ to clients;

// **Client** $i \in \mathcal{N}$

for *each round* $t \in \{0, \cdots, T-1\}$ **do**

  for $j = 0, \cdots, E-1$ **do**

    $g_i(\boldsymbol{w}_{i,j}^t) \leftarrow \nabla f_i(\boldsymbol{w}_{i,j}^t)$;

    $\boldsymbol{w}_{i,j+1}^t \leftarrow \boldsymbol{w}_{i,j}^t - \eta g_i(\boldsymbol{w}_{i,j}^t)$;

  $\boldsymbol{w}_i^t \leftarrow \boldsymbol{w}_{i,E}^t$;

  if $i \in \mathcal{S}^t$ **then**

    $\mathcal{C}_i^t, M_i^t \leftarrow$ Run PHE and sparsification by Alg. 2;

    Send $\mathcal{C}_i^t, M_i^t$ to the PS;

  Receive encrypted global model $\boldsymbol{C}^t$ and mask $\boldsymbol{M}^t$;

  $\boldsymbol{w}_i^t \leftarrow$ Decrypt and update with global model $\boldsymbol{w}^t$;

  Send sketch h_i^t of $\boldsymbol{w}_i^t$ to the PS by Alg. 3;

  Receive the selection set $\mathcal{S}^{t+1}$ from the PS;

B. Contribution-Aware Encrypted Weighted Aggregation

Building upon CKKS with sparsification, FedPHE conducts an efficient weighted aggregation on the ciphertexts received from selected clients as illustrated in Alg. 2. To accommodate client heterogeneity, the aggregation weights are adjusted based on the contributions of local updates to the global model.

CKKS-based PHE. To improve the efficiency of general HE schemes, PHE [17] is proposed by *packing* and *encrypting* multiple plaintext values $\{w_1, w_2, \cdots, w_{\mathcal{B}}\}$ into a single ciphertext, where $\mathcal{B}$ is the pack size. By facilitating parallel encryption/decryption operations on multiple plaintexts, PHE can greatly reduce the computation and communication overheads. In this study, we choose CKKS as the basis of PHE, which offers several advantages over Paillier and BFV. On one hand, CKKS allows encryption of *real numbers* directly on *vectors*, while Paillier and BFV take *integers* as input plaintext, which requires *quantizing* floating-point numbers. This increases the effectiveness of CKKS encryption and decryption operations by packing multiple vector items into a single polynomial. 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. In contrast, 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 and then encrypting the result.

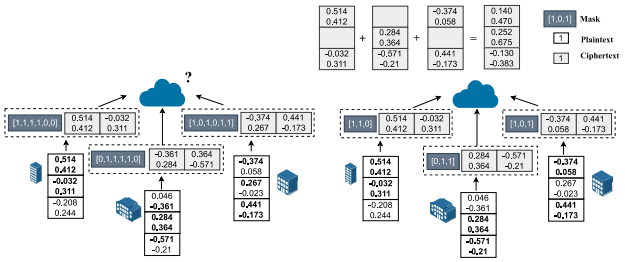


Fig. 5: Top-k sparsification for PHE.

Pack-level sparsification. Sparsification is a promising approach for reducing the communication traffic in FL [34]. In top-k sparsification, each client can sparsify its model updates by only selecting the top-k model parameters to send to the PS. However, sparsification is mainly implemented at the *scalar* level, and becomes *infeasible* once data are 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 [30]. An alternative method is packing model updates and then sparsifying the packs. Fig. 5 provides a visualization of this process, where each client conducts PHE and sparsifies encrypted packs. In this case, the PS can align the ciphertexts based on the packs' masks, i.e., the sparsity granularity is at the *pack* level.

Each selected client starts by flattening and packing the local model and then calculates the threshold θ to identify the mask given the sparsification ratio $s\%$. The mask will be set to 1 for those packs with mean values greater than θ, and the masked packs are encrypted accordingly. The ciphertext 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.

Contribution-aware weighted aggregation. To accommodate client heterogeneity, the PS aggregates the 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) [35] has been widely employed in many applications to approximate *Jaccard Similarity*, i.e., $JS(X, Y) = |X \cap Y|/|X \cup Y|$. The PS calculates the probability of sketch *collision* to estimate the Jaccard similarity of two local updates denoted by $JS(w_i^{t-1}, w_i^t)$, i.e.,

$$\Pr_{\mathcal{H}}(h_i^{t-1} = h_i^t) = JS(w_i^{t-1}, w_i^t). \tag{4}$$

According to [36], lower similarity implying higher inference loss is likely to achieve better performance improvement. Thus it should be assigned a larger aggregation weight p_i^t. That is,

$$p_i^t = \frac{\exp(-\beta \cdot JS(w_i^{t-1}, w_i^t))}{\sum_{j \in \mathcal{S}^t} \exp(-\beta \cdot JS(w_j^{t-1}, w_j^t))}, \tag{5}$$

Algorithm 2: Encrypted Weighted Aggregation

Input: Selected clients $\mathcal{S}^t$, local values $\{\mathcal{C}_i^t, M_i^t\}_{i\in\mathcal{S}^t}$, round t, packs K, sparsity ratio $s\%$
Output: Encrypted global model $\boldsymbol{C}^t$ and mask $\boldsymbol{M}^t$

// **Server**
Calculate weights $\{p_i^t\}_{i\in\mathcal{S}^t}$ based on Eq. (5);
$\boldsymbol{C}^t \leftarrow \sum_{i\in\mathcal{S}^t} \boldsymbol{E}(p_i^t)\cdot C_i^t$;
$\boldsymbol{M}^t \leftarrow \sum_{i\in\mathcal{S}^t} p_i^t \cdot M_i^t$;
// **Client** $i \in \mathcal{S}^t$
$\mathcal{P}_i := \{\mathcal{P}_i^1 \cdots \mathcal{P}_i^K\} \leftarrow$ Flatten and pack model $\boldsymbol{w}_i^t$;
$\theta \leftarrow$ Compute threshold $s\%$ largest in $\mathcal{P}_i$;
for *each pack* $\mathcal{P}_i[l] \in \mathcal{P}_i$ **do**
  $M_i^t[l] \leftarrow$ mask $|\mathcal{P}_i[l]| \geq \theta$;
  if $M_i^t[l]$ **then**
    $\mathcal{C}_i^t \leftarrow$ Append $\boldsymbol{E}(\mathcal{P}_i[l])$;
Send $\mathcal{C}_i^t, M_i^t$ to the PS;

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

After receiving ciphertexts $\{C_1^t, \cdots, C_{\mathcal{S}^t}^t\}$ and masks $\{M_1^t, \cdots, M_{\mathcal{S}^t}^t\}$ from selected clients $\mathcal{S}^t$, the PS performs encrypted weighted aggregation to get the global model, i.e.,

$$\boldsymbol{E}(\boldsymbol{w}^{t+1}) = \sum\nolimits_{i\in\mathcal{S}^t} \boldsymbol{E}\left(p_i^t\right) \times \boldsymbol{E}\left(\boldsymbol{w}_i^t\right), \tag{6}$$

which $\boldsymbol{E}\left(p_i^t\right)$ denotes the encrypted weight assigned to client i. Building upon CKKS's homomorphic multiplication, such aggregation in Eq. (6) is equivalent to performing weighted aggregation on plaintexts and then encrypting the result, i.e.,

$$\boldsymbol{E}(\boldsymbol{w}^{t+1}) = \sum\nolimits_{i\in\mathcal{S}^t} \boldsymbol{E}\left(p_i^t \times \boldsymbol{w}_i^t\right). \tag{7}$$

The encrypted global model $\boldsymbol{C}^t$ and mask $\boldsymbol{M}^t$ are subsequently distributed back to all clients for decryption and model updates. We summarize how FedPHE conducts contribution-aware encrypted weighted aggregation in Alg. 2.

C. Sketching-Based Client Selection

We employ client selection to address the straggler issue arising from client heterogeneity. In practice, different clients might send similar or redundant model updates to the PS, causing *unnecessary* communication costs. Existing selection approaches are largely conducted on plaintext, which contradicts the principles of privacy-preserving FL. To this end, we leverage the similarities of local updates to facilitate a sketching-based client selection in a privacy-preserving manner. Specifically, in each round, clients calculate and transfer the sketches of model updates to the PS, and then the PS clusters similar sketches together and select the fastest client from each cluster. The main steps are summarized in Alg. 3.

Sketching local models. After decryption and model updates, each client i first flattens the local model w_i^t to a d element tensor $\overline{w_i^t}$ and then creates 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

Algorithm 3: Sketching-based Client Selection

Input: Clients $\mathcal{N}$, round t, shared seed s, dimension k, weights $p_1, \cdots, p_N$, cluster threshold γ
Output: Selected clients $\mathcal{S}^{t+1}$ in the next round

// **Server**
C $\leftarrow \min(\mathbb{G}(\{\boldsymbol{h}_1^t, \cdots, \boldsymbol{h}_N^t\}), \gamma N)$;
$\mathcal{A}^t := \{\mathcal{A}_1^t \cdots \mathcal{A}_C^t\} \leftarrow$ Clustering $\{\boldsymbol{h}_i^t\}_{\mathcal{N}}$;
for *cluster* $\mathcal{A}_i^t \in \mathcal{A}^t$ **do**
  $s_i^{t+1} \leftarrow$ Select client with $\mathbb{F}_{\max}(\delta_j^{t-1}, T_j^t), j \in \mathcal{A}_i^t$;
  $\mathcal{S}^{t+1} \leftarrow$ Append s_i^{t+1};
// **Client** $i \in \mathcal{N}$
$\overline{\boldsymbol{w}_i^t} \leftarrow$ Flatten local model $\boldsymbol{w}_i^t$;
$\mathcal{M} \leftarrow$ Matrix $Gen(s, k)$;
$\boldsymbol{h}_i^t \leftarrow$ Sketching $\mathcal{H}(\overline{\boldsymbol{w}_i^t}, \mathcal{M})$;
Send $\boldsymbol{h}_i^t$ to the PS;

achieves *dimensionality reduction* on $\overline{w_i^t}$ and obtains a sketch $\boldsymbol{h}_i^t$. In particular, LSH is a family $\mathcal{F}$ of functions $\mathcal{H}$: $\mathbb{R}^d \to \mathbb{S}$, 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 [37, 38]. For any two clients' model updates, w_p and w_q, any hash function h chosen uniformly at random from $\mathcal{F}$ should satisfy

- If $d(w_p, w_q) < R$, then $\Pr_{\mathcal{H}}(h(w_p) = h(w_q)) \geq p_1$;
- If $d(w_p, w_q) \geq cR$, then $\Pr_{\mathcal{H}}(h(w_p) = h(w_q)) \leq p_2$;

where R is the threshold and c is an approximation factor. We exploit the properties of LSH functions to signify higher similarity during collisions when two inputs have the same hash code. Notice that the sketch $\boldsymbol{h}_i^t$ captures a *condensed* representation of local updates, ensuring no sensitive information is exposed. Moreover, this sketch achieves communication efficiency as it is far more compact than the original update.

Clustering sketches. After receiving sketches of local updates, the PS computes the cluster number $C = \min(\mathbb{G}(\{h_1^t, \cdots, h_N^t\}), \gamma N)$, where $\mathbb{G}(\cdot)$ denotes the *gap statistic*, a standard technique to determine the optimal cluster number [39]. Gap statistic compares the actual intra-cluster variation to expected values based on a null reference distribution, which is generated using Monte Carlo simulations. Here, γ is the threshold that limits the maximum cluster number. For any given $k = 1, \cdots, k_{\max}$, the gap statistic is defined as

$$\mathbb{G}_n(k) = \mathbb{E}_n^*(\log W_k) - \log W_k, \tag{8}$$

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 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 $s_k = sd(k) \times \sqrt{(1+B)/B}$. Finally, we choose the smallest k as the number of clusters such that

$$\mathbb{G}_k \geq \mathbb{G}_{k+1} - s_{k+1}. \tag{9}$$

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. According to the assumption that LSH hashes similar input items into the same *buckets* with a high likelihood, similar sketches mean similar local models.

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 $T_i^0, \cdots, T_i^{j-1}$, the priority $\mathbb{F}_i^t$ of being selected can be determined by

$$\mathbb{F}_i^t = \frac{1}{\alpha\delta_i^{t-1} + (1-\alpha) \times T_i^t}, \tag{10}$$

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 sketches of local models as a cluster feature is not only communication-efficient but also privacy-preserving as it maps high-dimensional model updates to a lower dimension through entry hashing. This advantage becomes particularly pronounced when applied to more sophisticated machine learning models.

D. Security Analysis

In this work, we assume that the PS does *not collude* with any client. The PS and clients are *honest-but-curious*, which is a widely adopted threat model in FL literature [40].

Theorem 1. *An honest-but-curious server cannot infer any private information about the clients.*

Proof. In the proposed FedPHE, the PS can only access the ciphertexts since the received local updates are encrypted before transmission. Consequently, the PS can not infer the clients' data and local models. □

Theorem 2. *An honest-but-curious client cannot eavesdrop on any private information about others.*

Proof. At the beginning of FedPHE, the PS and clients establish secure HTTPS connections, where the transferred data are encrypted using TLS/SSL protocol, ensuring that attackers are unable to access the data. □

Theorem 3. *The sketches and masks transferred in communication cannot disclose the privacy of clients.*

Proof. In LSH, the matrix $\mathcal{M}$ is generated from a secret shared seed s to map the parameters into an encrypted form. The mask hints at a package, not a scalar, and does not reveal privacy. An attacker lacks knowledge of the seed s and can only infer the matrix $\mathcal{M}$ through guesswork. □

IV. Performance Evaluation

In this section, we evaluate the performance of FedPHE, including secure weighted aggregation and client selection.

A. Evaluation Setup

Platform and parameters. Evaluations are conducted on a Dell server with NVIDIA GeForce RTX 3060 Ti GPUs using Pytorch. Consider a cross-silo FL scenario where ~~$N = 8$~~ clients collaboratively train a model. We study client heterogeneity of the Dirichlet Non-IID data setting (α = 1), similar to [41]. We implement BFV and CKKS with *TenSEAL* [42], 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 $2-5$ rounds' training time. The batch size is $B = 64$ and learning rate is $\eta = 1e-3$. The number of LSH hash functions is $k = 200$.

Datasets and models. We evaluate the results on three real datasets: MNIST [43], FashionMNIST [44] and CIFAR-10 [45]. In particular, we partition MNIST and FashionMNIST into $60,000$ training data and $10,000$ test data. For CIFAR-10, we use $50,000$ and $10,000$ images as the training data and test samples, respectively. A straightforward LeNet-5 neural network architecture [43] is employed for MNIST. For FashionMNIST, a CNN model with 2 convolutional layers and 1 fully connected layer is utilized. The ResNet-20 model [46] is employed for conducting experiments on CIFAR-10 dataset.

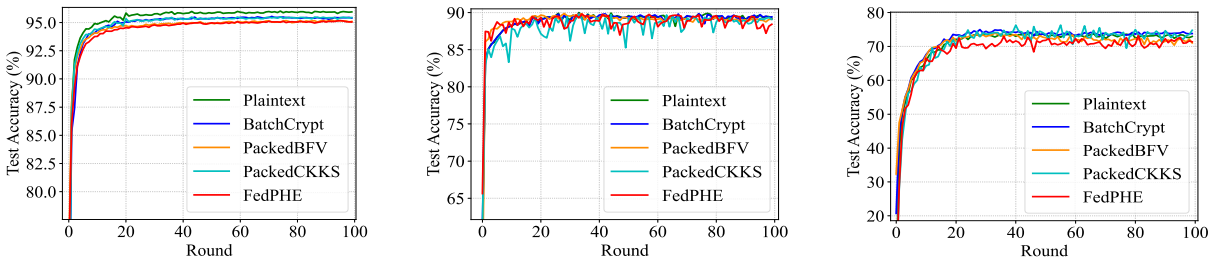
Baselines. To validate the proposed FedPHE, we introduce the following FL algorithms for comparison.

- *Plaintext*: an ideal upper bound for computation and communication overheads, where parameter transmission and weighted aggregation are conducted in plaintext.
- *BatchCrypt*: Paillier-based PHE [14], where clients quantize first, then pack and encrypt the model updates, while the PS performs aggregation on the ciphertext.
- *PackedBFV*: BFV-based PHE [28], where the model updates also need to be quantized and weighted on the client side before encryption, since BFV only supports integer operations.
- *PackedCKKS*: CKKS-based PHE [29], which leverages the ciphertext multiplication of CKKS to facilitate encrypted weighted aggregation on the PS side.
- *FedAvg*: federated averaging [31], where the PS randomly selects the subset of clients for aggregation.
- *FLANP*: straggler-resilient FL with adaptive client participation [47], which starts the training process with faster clients and gradually includes slower clients over time once the accuracy of current participants' data is reached.

B. Evaluations on Secure Weighted 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 with Paillier, PackedBFV, and PackedCKKS. The experiments were conducted until reaching convergence.

Accuracy. Fig. 6 illustrates the training process of the global model on different datasets, i.e., MNIST, FashionMNIST, and CIFAR-10. 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



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

Fig. 6: Accuracy versus global rounds of FedPHE and the baselines on different datasets.

TABLE III: Network traffic, accuracy, and training time in 100 rounds of FedPHE and the baselines on different datasets.

Dataset	Metric	Plaintext	BatchCrypt	PackedBFV	PackedCKKS	FedPHE
MNIST	Traffic (MB)	81	**217**	3959	2886	**175**
	Accuracy	95.94%	95.50%	95.10%	95.44%	95.04%
	Time (s)	342.34	**1377.03**	652.78	885.90	**743.26**
FashionMNIST	Traffic (MB)	73	**196**	3300	2550	**151**
	Accuracy	89.22%	89.42%	89.10%	89.07%	88.96%
	Time (s)	333.15	**1590.23**	650.68	823.88	**690.65**
CIFAR-10	Traffic (MB)	523	**1256**	22107	17089	**5165**
	Accuracy	72.95%	73.79%	71.02%	74.77%	71.37%
	Time (s)	1016.62	**7126.14**	1491.74	2419.18	**1605.71**

accuracy. While for FedPHE, the accuracy *fluctuates* within an acceptable range of $0.26\% - 1.58\%$, which arises from pack-level sparsification and client selection.

Network traffic and training time. We present the network traffic and training time in 100 rounds of FedPHE and the baselines on different datasets in Table III.

We observe that FedPHE reduces the network footprint for MNIST, FashionMNIST, and CIFAR-10 by up to $16.49\times$, $16.89\times$, and $3.31\times$, respectively, compared to PackedCKKS. Moreover, it outperforms PackedBFV for $4.28-22.62\times$ across three datasets. It is worth noting that the ciphertext size is only $0.81\times$, $0.77\times$, and $4.11\times$ 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 is previously in "memory out" state as shown in Table I, has been reduced to only $2.07 - 9.88\times$ larger than the plaintext baseline, making FedPHE applicable to FL in practice. In conclusion, FedPHE achieves communication overhead *reduction* ranging from $1.24\times$ to $22.62\times$ compared to these baselines.

As shown in Table III, BatchCrypt requires $4.02 - 7.01\times$ more training time compared to plaintext. In contrast, FedPHE incurs only $1.58 - 2.17\times$ 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 $1.85-4.44\times$. With an apt sparsification threshold, FedPHE does not adversely affect the trained model quality. Instead, it achieves significant compression while maintaining high performance.

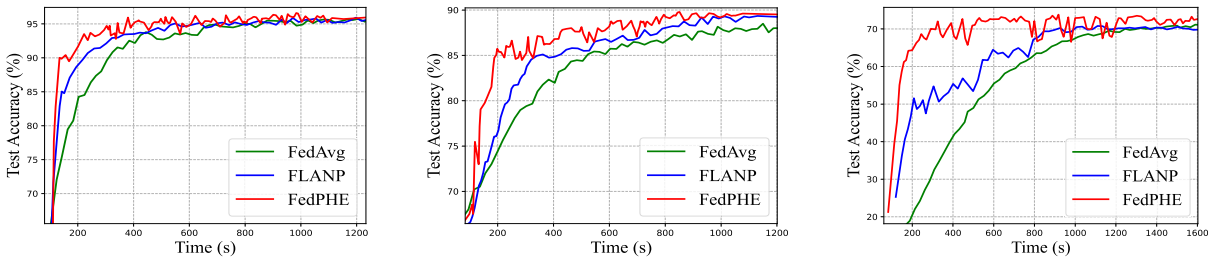
C. Evaluations on Client Selection

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

Accuracy. Fig. 7 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 sketching-based client selection to cherry-pick representative clients hosting diverse models and having the capability to train quickly.

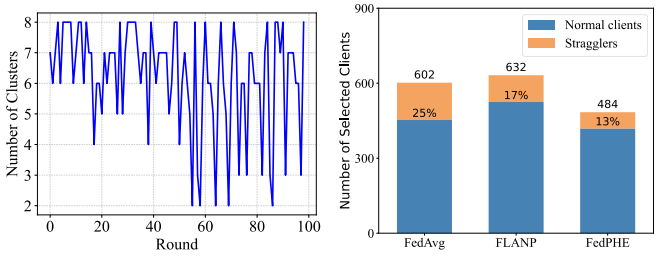
Number of clusters. From Fig. 8 (a), we can see that the cluster number fluctuates between 1 and 8, where the cluster threshold is set to $\gamma = 1$. When there are 8 clusters, it indicates the low similarity between clients' local models. In this case, clustering cannot be utilized to replace certain clients' models, and each client is assigned to a separate cluster. 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. 8 (b) with $\gamma = 0.625$, we observed that the proportions of selected stragglers are 25%, 17%, 13% for FedAvg, FLANP and FedPHE, respectively. FedAvg randomly selects a subset of clients to participate, which reduces the overall number of clients. However, many stragglers are still



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

Fig. 7: Accuracy versus clock time of FedPHE and the baselines on different datasets.



(a) Number of Clusters (b) Number of Selected Clients

Fig. 8: Number of clusters in each iteration and number of selected clients of FedPHE and the baselines.

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 $1.71-2.39\times$ 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.

V. Related Work

Communication-efficient HE. HE is a widely used privacy-preserving technique, yet suffers from significant inefficiency in computation and communication. To address this, Ge *et al.* propose to pack multiple plaintexts into a long integer[17]. Recently, Zhang *et al.* design BatchCrypt for cross-silo FL, achieving a substantial decrease in encryption overhead and ciphertext volume [14]. Jiang *et al.* further propose FLASHE, an additively symmetric HE in double masking to address the compatibility issues with sparsification [30]. 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 [16], and Brakerski *et al.* further extend SIMD to the standard LWE to achieve nearly optimal homomorphic evaluation [15]. However, existing HE solutions largely neglect client heterogeneity, an intrinsic problem in cross-silo FL, and FedPHE complements these studies with secure weighted aggregation and client selection.

Weighted aggregation. Many efforts have been devoted to addressing the client heterogeneity challenges. Among them, weighted aggregation is a promising technique to accelerate convergence. Zeng *et al.* present a contribution-aware model aggregation scheme, considering higher loss values are indicative of more substantial performance improvements [36]. Wu *et al.* adaptively assign aggregation weights to clients based on their contributions, measured by the angle between local and global gradient vectors [48]. Deng *et al.* propose FAIR to quantify each client's learning quality and automatically determine the optimal aggregation weights to enhance the global model quality [49]. 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 [31] 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 [47]. Furthermore, 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[32]. Kollias *et al.* utilize the sketches of local models to select clients that are similar to ours at a high level [50]. 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 sketching-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.

VI. Conclusion

In this paper, we present FedPHE, an efficient homomorphically encrypted FL framework for heterogeneous clients. To address the slow convergence and straggler challenges posed by client heterogeneity, FedPHE adopts CKKS-based PHE with sparsification to support contribution-aware encrypted weighted aggregation, while in the meantime conducting sketching-based client selection to cluster similar clients together and then cherry-pick the fastest client from each cluster. Extensive evaluations conducted on real-world datasets corroborate the superiority of FedPHE over existing benchmarks in accelerating FL convergence and reducing communication overheads without sacrificing accuracy.

REFERENCES

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

[2] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized cross-silo federated learning on non-iid data," in *Proc. AAAI*, vol. 35, no. 9, 2021, pp. 7865–7873.

[3] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[4] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE TIFS*, vol. 13, no. 5, pp. 1333–1345, 2017.

[5] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Proc. NeurIPS*, vol. 32, 2019.

[6] 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 CVPR*, 2021, pp. 16 337–16 346.

[7] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Proc. NeurIPS*, vol. 33, pp. 16 937–16 947, 2020.

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

[9] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "Hybridalpha: An efficient approach for privacy-preserving federated learning," in *Proc. ACM AISEC*, 2019, pp. 13–23.

[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. ICPP*, 2021, pp. 1–10.

[11] A. Li, H. Peng, L. Zhang, J. Huang, Q. Guo, H. Yu, and Y. Liu, "Fedsdg-fs: Efficient and secure feature selection for vertical federated learning," *Proc. IEEE INFOCOM*, 2023.

[12] N. Jovanovic, M. Fischer, S. Steffen, and M. Vechev, "Private and reliable neural network inference," in *Proc. ACM CCS*, 2022, pp. 1663–1677.

[13] S. Carpov and R. Sirdey, "Another compression method for homomorphic ciphertexts," in *Proc. ACM SCC*, 2016, pp. 44–50.

[14] 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 ATC*, Jul. 2020, pp. 493–506.

[15] Z. Brakerski, C. Gentry, and S. Halevi, "Packed ciphertexts in lwe-based homomorphic encryption," in *Proc. PKC*, 2013, pp. 1–13.

[16] N. P. Smart and F. Vercauteren, "Fully homomorphic simd operations," *Designs, codes and cryptography*, vol. 71, pp. 57–81, 2014.

[17] T. Ge and S. Zdonik, "Answering aggregation queries in a secure system model," in *Proc. VLDB*, 2007, pp. 519–530.

[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 INFOCOM*, 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," *Proc. IEEE INFOCOM*, 2023.

[20] C. Chen, W. Wang, and B. Li, "Round-robin synchronization: Mitigating communication bottlenecks in parameter servers," in *Proc. IEEE INFOCOM*, 2019, pp. 532–540.

[21] C. Chen, H. Xu, W. Wang, B. Li, B. Li, L. Chen, and G. Zhang, "Communication-efficient federated learning with adaptive parameter freezing," in *Proc. IEEE ICDCS*, 2021, pp. 1–11.

[22] Y. Deng, F. Lyu, J. Ren, Y.-C. Chen, P. Yang, Y. Zhou, and Y. Zhang, "Improving federated learning with quality-aware user incentive and auto-weighted model aggregation," *IEEE TPDS*, vol. 33, no. 12, pp. 4515–4529, 2022.

[23] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *Proc. USENIX OSDI*, 2021, pp. 19–35.

[24] A. Ali, T. Lepoint, S. Patel, M. Raykova, P. Schoppmann, K. Seth, and K. Yeo, "Communication-computation trade-offs in pir," in *Proc. USENIX Security*, 2021, pp. 1811–1828.

[25] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *EURASIP Journal on Information Security*, vol. 2007, pp. 1–10, 2007.

[26] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–35, 2018.

[27] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Eurocrypt*, 1999, pp. 223–238.

[28] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *Cryptology ePrint Archive*, 2012.

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

[30] Z. Jiang, W. Wang, and Y. Liu, "Flashe: Additively symmetric homomorphic encryption for cross-silo federated learning," *arXiv preprint arXiv:2109.00675*, 2021.

[31] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.

[32] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi, "Clustered sampling: Low-variance and improved representativity for clients selection in federated learning," in *Proc. ICML*, 2021, pp. 3407–3416.

[33] Y. Zhang, Z. Wang, J. Cao, R. Hou, and D. Meng, "Shufflefl: Gradient-preserving federated learning using trusted execution environment," in *Proc. ACM CF*, 2021, pp. 161–168.

[34] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE TNNALS*, vol. 31, no. 9, pp. 3400–3413, 2019.

[35] B. Neyshabur and N. Srebro, "On symmetric and asymmetric lshs for inner product search," in *Proc. ICML*, 2015, pp. 1926–1934.

[36] 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. ICPP*, 2021, pp. 1–10.

[37] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *IEEE Signal processing magazine*, vol. 25, no. 2, pp. 128–131, 2008.

[38] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. IEEE ICCV*, 2009, pp. 2130–2137.

[39] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[40] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proc. IEEE SP*, 2013, pp. 463–477.

[41] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[42] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "Tenseal: A library for encrypted tensor operations using homomorphic encryption," *arXiv preprint arXiv:2104.03152*, 2021.

[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[44] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[45] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778.

[47] 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 JSAIT*, vol. 3, no. 2, pp. 197–205, 2022.

[48] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE TCCN*, vol. 7, no. 4, pp. 1078–1088, 2021.

[49] Y. Deng, F. Lyu, J. Ren, Y.-C. Chen, P. Yang, Y. Zhou, and Y. Zhang, "Improving federated learning with quality-aware user incentive and auto-weighted model aggregation," *IEEE TPDS*, vol. 33, no. 12, pp. 4515–4529, 2022.

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