FedCMK: An Efficient Privacy-Preserving Federated Learning Framework

Pengyu Lu[1], Xianjia Meng[1,2](✉), and Ximeng Liu[2]

[1] Northwest University, Xi'an, Shaanxi 710069, China
xianjiam@nwu.edu.cn
[2] Fuzhou University, University Town Fuzhou, Fujian 350108, China

Abstract. Federated learning emerged to solve the privacy leakage problem of traditional centralized machine learning methods. Although traditional federated learning updates the global model by updating the gradient, an attacker may still infer the model update through backward inference, which may lead to privacy leakage problems. In order to enhance the security of federated learning, we propose a solution to this challenge by presenting a multi-key Cheon-Kim-Kim-Song (CKKS) scheme for privacy protection in federated learning. Our approach can enable each participant to use local datasets for federated learning while maintaining data security and model accuracy, and we also introduce FedCMK, a more efficient and secure federated learning framework. FedCMK uses an improved client selection strategy to improve the training speed of the framework, redesigns the key aggregation process according to the improved client selection strategy, and proposes a scheme vMK-CKKS, to ensure the security of the framework within a certain threshold. In particular, the vMK-CKKS scheme adds a secret verification mechanism to prevent participants from malicious attacks through false information. The experiments show that our proposed vMK-CKKS schemes significantly improve security and efficiency compared with the previous encryption schemes. FedCMK reduces training time by 21% on average while guaranteeing model accuracy, and it provides robustness by allowing participants to join or leave during the process.

Keywords: Homomorphic encryption · Federated learning · Multi-key · CKKS · Machine learning · Secret sharing

1 Introduction

With improved computing power and increased data volume, deep learning has achieved remarkable success in computer vision, natural language processing, and other fields. However, large-scale data collection and storage often lead to privacy leakage and data security issues. To solve the problem of privacy leakage

Supported by National Natural Science Foundation of China under grant number 62276211.

in machine learning with a large amount of data and the problem of data island that a large amount of data cannot be applied, the concept of federated learning comes into being, aiming at distributed machine learning under the premise of protecting data privacy [14,19]. In contrast, federated learning empowers individual devices to train models locally and share only the model updates with a central server, preserving the users' privacy. In recent years, federated learning has received much attention from academia and industry due to its potential in various applications such as healthcare, finance, and the smart internet of things [4,15,23].

Nevertheless, federated learning still faces some challenges in practical applications [16]. In order to enhance the security of federated learning, many scholars have carried out research on homomorphic encryption, differential privacy, and secure multi-party computation [9,11,24]. Additionally, federated learning can also be attacked by malicious actors who may transmit faulty model updates or manipulate training processes, damaging overall model performance [1]. Therefore, it is crucial to design an effective security mechanism to detect and resist such attacks. Furthermore, federated learning involves multiple participants conducting model training locally, which may result in a slower overall convergence speed. Balancing the process of local training and global aggregation to improve training speed and model performance is also a problem worth studying.

In this study, we use improved client selection strategies and multi-key homomorphic encryption schemes to solve the problems of data leakage, malicious party attacks, model training, and convergence rate optimization. Specifically, we improve the client selection strategy to improve model training and convergence speed [20]. Based on the improved client selection strategy, we designed a multi-key homomorphic encryption scheme, namely vMK-CKKS, to protect data privacy and solve the problem of collusion attacks of malicious actors. We propose a complete federated learning framework, FedCMK, and verify its security and robustness through experiments. Our contribution is as follows:

(1) We propose a client selection strategy to solve the device heterogeneity problem and ensure the model's accuracy while maximizing the training efficiency. Experiments show that our client-selected federated learning framework can improve the training speed by about 21% compared with the traditional federated learning framework without more than 1% accuracy loss. For a specific model, the training speed can be improved by up to 33%.
(2) We design a MK-CKKS scheme: vMK-CKKS. Based on the client selection strategy, we redesigned the way of public key aggregation for different clients. The vMK-CKKS scheme is based on verifiable secret sharing, ensuring security within a limited threshold. It prevents the participants from maliciously sending false information to destroy the decryption. Experiments show that our multi-key CKKS scheme is more efficient and secure than the traditional encryption schemes.
(3) We propose a cross-device federated learning framework FedCMK based on the client selection strategy with vMK-CKKS encryption scheme. Under the premise of enhancing privacy security and training efficiency, the

framework also supports arbitrary training strategy and has high scalability and robustness. Experiments show that FedCMK can effectively complete different cross-device federated learning tasks and resist malicious attacks from participants within a certain threshold.

(4) We theoretically prove the security of our federated learning framework and compare its communication cost and computation cost with some existing homomorphic encryption schemes, evaluate the efficiency and performance of our federated learning framework, and discuss some potential risks.

The remainder of this paper is organized as follows. In Sects. 2 and 3, we introduce related works and the preliminaries. In Sect. 4, we introduce the FedCMK framework, improved client selection strategies, and vMK-CKKS scheme. Then we present the experimental environment and parameters in Sect. 5, and we evaluate the experimental results. Finally, we provide proof of the framework's security in Sect. 6 and conclude the paper in Sect. 7.

2 Related Work

Our research is mainly aimed at federated learning and multi-key homomorphic encryption. In this section, we will summarize the current federated learning framework based on multi-key homomorphic encryption. At the same time, we will introduce traditional single-key homomorphic encryption schemes.

2.1 Homomorphic Encryption Based FL

Federated learning based on homomorphic encryption offers more robust security without affecting model accuracy. Homomorphic encryption has become the most common privacy protection method in federated learning. Recently, many federated learning frameworks using homomorphic encryption have been proposed. For example, Dimitris et al. proposed MetisFL, a homomorphic encryption-based federated learning model for training neural models and predicting certain diseases [21]. However, their research did not optimize homomorphic encryption schemes or consider potential model leakage. Moreover, their focus was on personalized FL [22]. In recent years, many federated learning frameworks have used the Paillier semi-homomorphic encryption scheme [24–26]. However, the Paillier scheme's nature is unsuitable for large-scale machine-learning gradient encryption. In recent years, the CKKS scheme has become the mainstream homomorphic encryption framework for federated learning. Microsoft has implemented CKKS in the SEAL library, and the CKKS scheme has been widely researched and applied in recent years.

2.2 Multi-key Homomorphic Encryption Based FL

Multi-key homomorphic encryption is more suitable for large-scale multi-party federated learning scenarios than traditional single-key homomorphic encryption schemes [5]. Ma et al. have proposed the xMK-CKKS scheme, which simplifies the aggregation of public keys by adding them to form an aggregated public key for encrypting model updates [17]. This approach has the advantage of being easy to implement. It can be extended to multiple participants while maintaining a certain level of security against collusion between $K-1$ participants and the server. However, this approach also has some limitations. Firstly, when there are many participants, aggregating all public keys may lead to excessive noise, which can negatively impact the accuracy of the ciphertext and increase computation and communication costs. Secondly, the entire aggregate public key must be reset if a participant drops out. Finally, although it effectively prevents collusion between the server and other actors, it is assumed that the server is a trusted third party in federated learning. In extreme collusion cases, the server can send false information to obtain private data. In contrast, Du et al. have proposed the tMK-CKKS scheme, which uses Shamir's secret sharing to reduce overhead while ensuring security [7]. However, Shamir's secret-sharing scheme is not effective in preventing malicious secret sharing between participants, which could result in decryption failure. Some studies focus on vertical federation learning scenarios, such as CryptoBoost, an XGBoost framework proposed by Jin and Wang et al. based on multi-party homomorphic encryption technology [12]. CryptoBoost is end-to-end secure, and it proposes a new set of communication protocols to reduce costs. Applying multi-key homomorphic encryption under vertical federation learning is also a primary direction for future research [18]. Based on this, we improve the above algorithm and design a multi-key CKKS variant that addresses efficiency and security issues in existing schemes.

3 Preliminaries

In this section, we outline part of the notations used in the paper and introduce the FedCS client selection protocol, which is the basis of our improved client selection protocol, while we present the multi-key homomorphic encryption-related techniques.

We set the secret distribution *chi* to be the uniform distribution over the set of polynomials in R with coefficients $\{0, \pm 1\}$. Each coefficient of the error $e \leftarrow \psi$ is plotted according to a discrete Gaussian distribution centered at zero and standard deviation $\sigma = 3.2$. The model's weight used in the experiment is represented by 32 bits of floating point numbers.

3.1 Federated Learning

Federated learning is a cutting-edge artificial intelligence technology that prioritizes user privacy and data security by ensuring participants cannot access each other's data. As a mainstream algorithm in federated learning, FedAvg allocates a fixed number of training steps to each participant and aggregates locally trained models to compute a new global model. This approach allows for model updates without requiring the exchange of raw data between participants, ensuring privacy and security. The FedAvg algorithm has been widely adopted in practical applications of federated learning.

3.2 Client Selection

In cross-device federated learning, the participation of numerous edge devices and mobile terminals with limited performance capabilities can lead to a significant waste of computing resources or the exclusion of numerous devices that cannot perform multiple epochs within a given time frame. Nishiod et al. proposed the FedCS protocol to solve the heterogeneous problems in federated learning [20]. We improved on this to fit our framework.

3.3 MK-CKKS Scheme

Song and Dai proposed a multi-key homomorphic encryption (MK-HE) scheme based on CKKS in their work [3,5,6,8]. They designed multi-key variants of Brakerski-Fan-Vercauteren (BFV) and CKKS and provided a new relinearization scheme. Moreover, they applied the MKHE scheme to evaluate convolutional neural network (CNN) models.

However, directly applying the MK-CKKS scheme to federated learning may result in privacy risks since the server can decrypt model updates and access personal data during decryption. While the server is typically trustworthy, this contradicts the goals of federated learning. Therefore, our MK-CKKS scheme limits the decryption ability of the server to the ciphertext of a single client, preventing the potential privacy leakage risk on the server side. Specifically, for encrypted model updates, the server can only decrypt the sum of all model updates in ciphertext and cannot decrypt the model updates of individual participants separately.

4 FedCMK

In this section, we introduce our cross-device federated learning framework FedCMK, including its system model, our simulated attack model, improved client selection algorithm, and its encryption algorithm and overall system flow.

4.1 Problem Statement

Suppose there is an encrypted cross-device federated learning framework, a total of K clients participate in the training process, and the whole training process is based on the Federated Average (FedAvg) algorithm. The server randomly selects several clients for this round of learning and sends them the global model. The selected clients use the local data set for local training and upload the updated gradient information to the server after encryption. In such a system, we might face the following challenges:

Device Heterogeneity: In a cross-device federated learning system, the performance between devices and the amount of data is different. If the client is randomly selected, it may cause a lot of computing power or data waste. Therefore, how to choose the client will affect the accuracy and time of training.

Encryption and Decryption of Gradients: In this cross-device federated learning system, each participating client may have its key. If each client encrypts the gradient with its key and uploads it, the decryption process will be difficult for the server. If a uniform key is used, the security of the key is not satisfactory to every client, and the parties may be malicious and conspire to steal data. At the same time, if the server can decrypt the ciphertext of each client separately, then a not fully trusted server will easily steal all the data, which is also unacceptable. Therefore, selecting an appropriate encryption scheme is essential to ensure local data security.

Robustness of the System: In this cross-device federated learning system, any client may join or leave during the process, and their actions should not affect the entire training.

4.2 Threat Model

In our threat model, we default the federation launcher to be a trusted entity. In the vMK-CKKS scheme, it will generate the secret to building the aggregate public key used for encryption. While the federation controller and federation learners are honest and curious, they will strictly follow the protocol but will be curious to infer the data of other learners. In order to better reflect the security, we introduce an active adversary $\mathcal{A}$ into the model. In the vMK-CKKS scheme, we set $A = t$. $\mathcal{A}$ should be several learners smaller than A, or a federation controller and several learners smaller than A. The goal of $\mathcal{A}$ is to obtain as many ciphertext decryption results as possible to steal the local data of learners. The following are some possible inferences:

1. $\mathcal{A}$ may consist of at most $A - 1$ learners who obtain each other's ciphertexts or decrypt shares by collusive attacks and wish to decrypt the ciphertexts to steal data from the remaining learners.
2. $\mathcal{A}$ is some maliciously participating learner who broadcasts the wrong secret shares to other learners.
3. $\mathcal{A}$ maliciously sends the wrong decrypt share and hopes to cause an error in the decryption process.
   We notice that such an opponent is very typical in the threat model [2].

4.3 Our Client Selection Design

The original FedCS scheme uses a greedy algorithm to strive for as many high-performance participants as possible to participate in the training task, but this also has some shortcomings. Secondly, it may lead to many devices being unable to participate in the training task. In the case of uneven data distribution, it may lead to certain data waste problems. Finally, some high-performance devices may be malicious, leading to persistent malicious attacks. Therefore, we make some optimizations based on Nishio et al. 's FedCS framework to be more suitable for our federated learning framework and homomorphic encryption scheme. Firstly, we notice that the choice of T_r in the original scheme has a significant impact on the final model update, so we compare the training time and accuracy of FedCMK under different T_r, and choose a more appropriate T_r value. We also add a safety margin $T_{r_s} = \frac{1}{60}T_r$ to account for the network fluctuations that communication may face in real-world applications. Second, we clustered all the clients and selected clients for each clustered client set to make use of as many devices as possible. Finally, we observe that clients with larger datasets tend to be underutilized, leading to severe data wastage and potentially lower model accuracy. To solve this problem, we introduce a weighted selection scheme in which we add several clients with large datasets according to a weight $W_s$in the S set. In addition, we cache the clients with excellent performance in subset $S'$and can directly schedule them for subsequent training if they are idle. The Settings of weight W_s and subset S' vary from device to device. Our experiments compare the accuracy and time of training under different W_s. Considering the balance of performance and efficiency, we chose $W_s = 0.2$ and $|S'| = 0.1\,|S|$.

4.4 Our MK-CKKS Scheme Design

We have optimized and proposed a MK-CKKS variant that will be used to build our federated learning framework FedCMK. The vMK-CKKS scheme is based on verifiable secret sharing (VSS), which is similar to Shamir's secret sharing but with an additional verification mechanism [10]. This mechanism enables participants to verify the correctness of the received secret fragments, helping to prevent malicious actors from tampering with or forging shares during the sharing process, thus enhancing the system's security. Moreover, the verification function of all parties of VSS improves the system's fault tolerance and robustness since even if some participants provide incorrect shares when restoring the aggregate public key, the final result will not be affected. We will discuss this method in more detail below.

- **SecretShare** : A trusted third party performs secret generation, and we refer to this third party as the generator hereinafter(GH). GH randomly selects $a_i \in \mathbb{Z}_p$, and construct a polynomial of degree t-1, satisfying $f(x) = a_0 + a_1x + \cdots + a_{t-1}x^{t-1}(\text{mod}p)$, sets $a_0 = z$. For a client k_i, its secret share is $z_i = f(i)$. Any t participants can jointly reconstruct the secret.
- **Setup** : For a given security parameter λ, set the RLWE dimension n, ciphertext modulus q, key distribution χ and error distribution ψ over R. Then, takes all the security parameters as input and returns the public parameterization(n,q,χ,ψ,R).
- **KeyGen** : For the generator hereinafter, randomly selects a secret $z \in \mathbb{Z}_p$, this secret will be split into n shares, and each of which will be held by one participant. Simultaneously GH computes $A_j = g^{a_j}$, where $j = 0, 1 \ldots, t-1$,

and exposes those parameters. Therefore, the aggregated public key can be expressed as $\tilde{b} = -s_i \cdot a + e \pmod{q}$, where $s_i = z \cdot s \pmod{q}$, $s \leftarrow \chi$.

- **Verify** : For a client k_i verifies the correctness of the secret after receiving it and refuses to perform subsequent operations if the equality $g^{z_i} = \Pi_{j=0}^{t-1} A_j^{x_i^j}$ is not met.
- **Encryption** : Let $a = \mathbf{a}[0]$, $b = \mathbf{b}[0]$. Sample $v \leftarrow \chi$ and $e_0, e_1 \leftarrow \psi$, For a client k_i, encoding a plaintext $m_i \in \mathcal{M}$ and outputs a ciphertext $ct_i \in \{0, 1\}$, where

$$ct_i = (c_0^{k_i}, c_1^{k_i}) = (v'^{k_i} \cdot \tilde{b} + m_i + e_0^{k_i}, v'^{k_i} \cdot a + e_1^{k_i}) \pmod{q} \tag{1}$$

- **Add** : The sum of ciphertext is as follows:

$$C_{sum} = \sum_{i=1}^{K} ct_i \triangleq (C_{sum_0}, C_{sum_1},) = \left(\sum_{i=1}^{K} (v'^{k_i} \cdot \tilde{b} + m_i + e_0^{k_i}), \sum_{i=1}^{K} (v'^{k_i} \cdot a + e_1^{k_i}) \right) \pmod{q} \tag{2}$$

- **Decryption** : Any K participants can jointly reconstruct the ciphertext, and the decryption share is calculated as follows:

$$D_i = s_i \cdot C_{sum_1} + e_i^* = s_i \cdot \sum_{i=1}^{K} (v'^{k_i} \cdot a + e_1^{k_i}) + e_i^* \pmod{q}, e^* \leftarrow \psi \tag{3}$$

Then the sum of all plaintexts C_{sum} can be decrypted as the same.

4.5 FedCMK Design

Based on the above discussion, we have designed a federated privacy-preserving learning framework using the vMK-CKKS scheme. In this framework, a trusted server acts as the federation launcher, serving as the entry point for the entire federated learning process. Before the federated learning process begins, the federation launcher initializes the model and defines the required machine learning architecture. It also generates hyperparameters and a secret for aggregating public keys distributed to all learners. The federation controller is responsible for scheduling learners to perform federated learning tasks and aggregating the local model updates of each learner to compute a new global model. Prior to each round of training, the federation controller selects K' learners, who transmit their performance status and resource information to the federation controller. The federation controller then selects S learners to participate in the current round of training. Each participant in the federated learning process is referred to as a federated learner, and communication between learners is limited to the broadcast phase during secret verification. The learner receives the global model from the federation controller and trains it locally using their private dataset and the tasks assigned by the federation controller. After completing one round of training, the learner sends the ciphertext of their model update to the federation controller.

Therefore, a complete round of federated learning process will be expressed as follows:

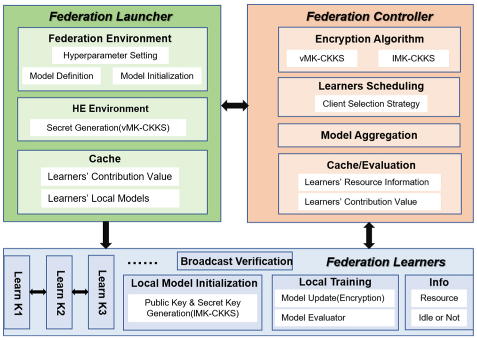


Fig. 1. The Federated Learning Framework Based on our MK-CKKS Homomorphic Encryption (FedCMK)

Initialization: The federation launcher completes the setting of hyperparameters, such as the dimension of RLWE, the ciphertext modulus, and the sampling distribution, and sets up the federation environment. At the same time, the federation launcher generates secrets for aggregated public keys.

Client Selection: The federation controller randomly selects K' learners, and the selected K' learners send their current resource information to the federation controller, such as whether the CPU/GPU is occupied, the approximate size of the local dataset. The federation controller then selects S learners according to the client selection strategy for this round of training.

Local Training: After determining the learners for this round, the federation controller selects the training model, and the selected learners download the global model and conduct local training on their private dataset, generating the local model.

Model Update Encryption: The learners encrypt their local model updates using the secret and public key distributed by the federation launcher and upload the encrypted model updates to the federation controller.

Ciphertext Aggregation: After the federation controller receives the model update ciphertexts of all participating learners, it adds all the ciphertexts into C_{sum}.

Decryption: The federation controller sends C_{sum_1} to learners in this round (if learners in this round S are less than the decryption threshold t, learners in K' are selected in turn), then the selected learners calculate their decryption shares and upload to federation controller. After the federation controller gets all the decrypted shares, it uses C_{sum} and the decrypted shares to restore the plaintext and then updates the global model for $w + 1$ rounds. Then federation controller distributes the new global model to learners participating in the next round (Fig. 1).

5 Performance Evaluation

5.1 Experimental Setup

Our evaluation of the federated framework was conducted on a server with an Intel i5-11400F CPU, NVIDIA RTX 3060Ti GPU, and 16GB RAM, running the Ubuntu 22.04 operating system. We implemented the FedAvg algorithm using Pytorch to evaluate our federated framework. Our multi-key encryption scheme was built using the HEAAN library and compared with several previous multi-key CKKS schemes. We also compared the privacy-preserving learning of Paillier's scheme.

5.2 Results

To evaluate our federated framework, we first measured the accuracy and time cost of one round of FedAvg without multi-key homomorphic encryption. Next, we measured the accuracy and time cost of the round of communication after adding the vMK-CKKS scheme. We used three datasets, MNIST, Shakespeare, and CIFAR100, to conduct four experiments. Additionally, we compared the performance of our federated learning framework with several recent federated frameworks.

Client Selection: We first compared the classical FedAvg federated learning scheme without introducing the client selection strategy and the federated learning scheme with the introduction of the client selection strategy. After comparison, the average time to reach convergence of the federated learning scheme with the introduction of the client selection strategy is significantly reduced. Due to different data sets and different parameter Settings, the convergence will be greatly affected. $T_r = 1\,\text{min}$, and the number of clients selected in each round $S = 0.1K'$. Under this parameter setting, the average accuracy difference between the experiment and the federal learning scheme without client selection is less than 1%, but the training time is reduced by 23% on average.

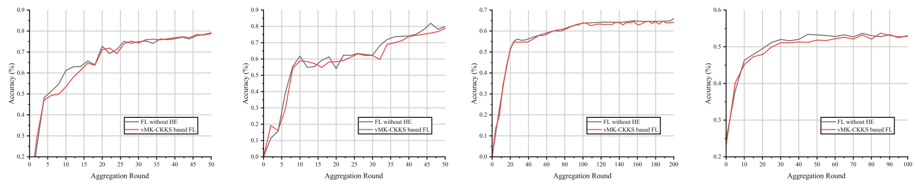
Accuracy: To compare the accuracy and security of our federated learning framework, we evaluated the performance of different models and different datasets and compared it with several other federated learning frameworks.

Table 1. Comparison of convergence time of different FL schemes

FL Scheme	Total training time	Accuracy
Paillier based FL	105 min	78.9%
xMK-CKKS based FL	73 min	79.5%
tMK-CKKS based FL	61 min	79.6%
vMK-CKKS based FL	52 min	78.8%

We contrast our federated framework with a federated learning framework without privacy protection. Four experiments were conducted for each scheme, using the MNIST, Shakespeares, and CIFAR100 datasets. After comparing the experimental results, before adding homomorphic encryption, the federated learning framework has an accuracy rate of 79.2%, 79.8%, 65.5% and 53.0% in the four experiments, and then added our two multi-key After the CKKS scheme, the accuracy rates are 78.9%, 79.1%, 65.2%, 52.8%, and 78.8%, 78.9%, 64.2%, 52.9%. It means that our multi-key CKKS scheme keeps the accuracy of federated learning model training the same. We list some parameters of the experiment and the final result curve, and we can see that our federated learning framework curve is very similar to the original framework without privacy protection. We detailed our experimental parameter Settings in Table 1, and the results obtained are shown in Fig. 2.

Efficiency: In order to evaluate the performance of the two MK-CKKS schemes, we compared it with the mainstream federal environment homomorphic encryption scheme Paillier, and we also compared it with the xMK-CKKS scheme and the tMK-CKKS scheme. Our experiments mainly compare the following aspects: first, the model update ciphertext size under the Paillier scheme and the model update ciphertext size under different multi-key CKKS schemes, which are compared in detail in Table 2, and second, the time cost of encryption and decryption in the process of other encryption schemes and our encryption scheme, that is, the computational cost.



(a) MNIST - LR (b) MNIST - CNN (c) CIFAR (d) Shakespeare

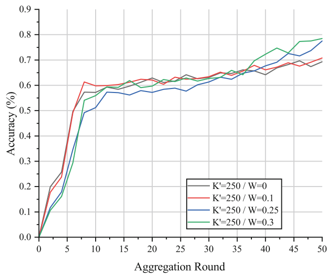
Fig. 2. Performance of different datasets and models under several FL frameworks

According to Table 2, although the encryption and decryption speed of the Paillier scheme is faster than that of the MK-CKKS scheme within a specific range, the average encryption time of the CKKS scheme(0.04ms) is much smaller than that of the Paillier scheme(0.34ms) because of more plaintext can be packaged in the ciphertext. For several MK-CKKS schemes, xMK-CKKS, tMK-CKKS, and vMK-CKKS are homomorphic schemes that meet a threshold. The threshold of MK-CKKS is K, and the threshold of tMK-CKKS and vMK-CKKS can be unified into t so that the decryption time will change according to the values of t and K. Theoretically, when t is less than K, The xMK-CKKS scheme takes longer to decrypt. In general, S will be much smaller than t and K, so the decryption time per round will be shorter than several other MK-CKKS schemes.

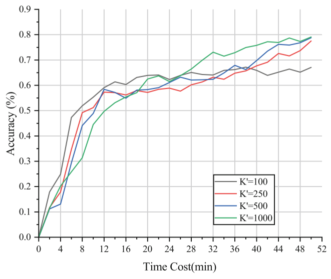
Table 2. Different Homomorphic Scheme Parameters And Time Costs

Scheme	Library	Security level	Packing Size	Key size	Ciphertext size	Enc(ms)	Dec(ms)	Add(ms)
Paillier	Python-Paillier	128	60	3072	6144	31.3	15.7	0.1
xMK-CKKS	HEAAN	128	2048	4096	8192	77.1	19.2	2.5
tMK-CKKS	HEAAN	128	2048	4096	8192	77.1	19.2	2.5
vMK-CKKS	HEAAN	128	1024	2048	4096	33.7	12.4	1.8
vMK-CKKS	HEAAN	128	2048	4096	8192	77.1	19.2	2.5

Figure 3 shows the effect of different client weights and clustering on federated learning training time and accuracy. We notice that when the considerable data weight reaches 0.25, which means that there is at least one-quarter of big data clients, our federated learning model can guarantee almost the same accuracy as the original model, but the training time decreases by about 17%. When the weight reaches 0.3, the accuracy of the model is improved by 0.01%, but the training time is only decreased by 10% compared to the original model. Considering the efficiency requirement in practical applications, we set the weight to 0.25 to ensure the balance between accuracy and time overhead. The model can obtain high accuracy quickly for the clustering strategy when the number of clusters is 4. That is, 1000 clients are grouped into 4 clusters of 250 clients per group. Compared with the unclustered algorithm, the accuracy of the model is reduced by less than 0.01% when divided into four clusters, but the iteration time of each round is reduced by about 27%, and the overall training time is reduced by about 10%. In summary, the improved client selection strategy ensures the accuracy of the model while reducing the training time and making more clients participate in the training.



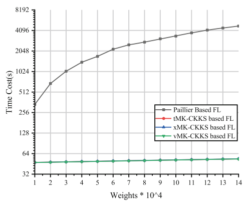
(a) Accuracy and time cost under different client weights



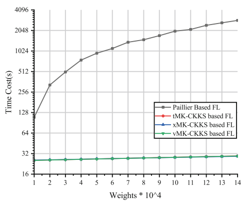
(b) Accuracy and time cost under different client clusters

Fig. 3. Accuracy and time cost under different client clusters and weights

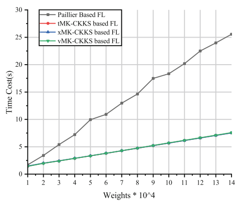
Figure 4 shows the computational cost of encryption, decryption, ciphertext addition, and ciphertext decryption under several schemes. We compare the computational cost of our two schemes with Paillier's scheme, the xMK-CKKS scheme, and the tMK-CKKS scheme. For Paillier's scheme, as the CKKS scheme packs more ciphertexts simultaneously (based on polynomial dimension), as the amount of data increases, It is faster than Paillier's scheme in encryption and decryption, and the gap increases linearly with the number of models to be encrypted. For xMK-CKKS and tMK-CKKS schemes, there is no obvious difference in the speed of encryption, decryption, and ciphertext addition under the same parameters. However, in the decryption phase, the xMK-CKKS scheme requires all K clients to calculate the decryption share, so the computational cost is high. For the tMK-CKKS scheme and vMK-CKKS scheme, more than t clients must jointly decrypt the calculation because t is usually less than K, and the computational cost is low. At the same time, due to client selection, clients involved in vMK-CKKS decryption share calculation often have better performance. Therefore, the decryption speed is faster than tMK-CKKS. We show a more detailed comparison of several MK-CKKS schemes in Fig. 5.



(a) Average encryption time cost



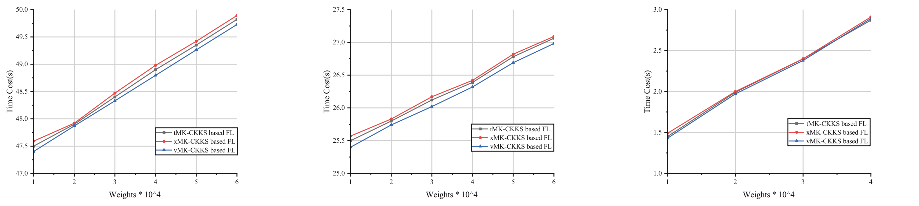
(b) Average decryption time cost



(c) Average addition time cost

Fig. 4. Comparison of calculation time cost between Paillier and different MK-CKKS schemes

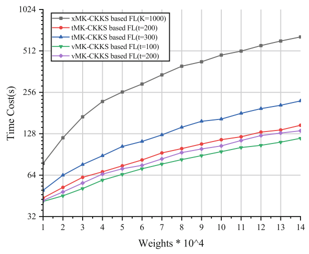
Figure 6 shows the computational overhead on the server side. In Fig. 6(a), we compare the influence of different client numbers K, threshold size t, and the number of clients selected in each round of the client selection strategy S on the decryption of the aggregated ciphertext. It can be seen that the value of K is much larger than t and S in general. The decryption cost of xMK-CKKS is higher than that of other schemes because it requires all clients to aggregate. For different choices of t, a smaller value of t will bring faster decryption speed but reduce the security of collusion attacks. The threshold-based secret sharing method for decryption is still faster than the aggregation method in xMK-CKKS. For our multi-key homomorphic scheme, since S in the client selection strategy is smaller than t, the decryption still requires at least t clients to participate. Therefore, the whole is still faster than the tMK-CKKS scheme under the same threshold t. In Fig. 6(b), we compare the time cost of different models. It can be seen that the vMK-CKKS scheme reduces the time cost by about 6% compared with tMK-CKKS. In practical application, considering the balance between security and efficiency, the weight of vMK-CKKS can further reduce the time cost.



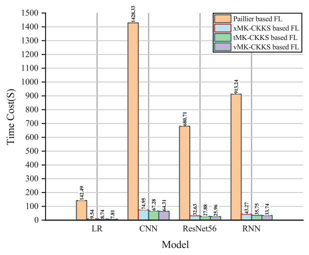
(a) Average encryption time cost of different MK-CKKS schemes (b) Average decryption time cost of different MK-CKKS schemes (c) Average addition time cost of different MK-CKKS schemes

Fig. 5. Calculation cost of different MK-CKKS schemes

Figure 7 shows the communication cost under different encryption schemes, and we take the ciphertext size simplicity of different schemes as the cost of the communication overhead. In the Paillier scheme, the cost of the ciphertext grows linearly much more than the other CKKS schemes. The xMK-CKKS scheme ($K = 1000$) always has a higher ciphertext cost than tMK-CKKS and vMK-CKKS based on a threshold ($t = 300$). In the tMK-CKKS scheme, due to the client selection strategy of vMK-CKKS), the size of plaintext used for encryption in each round is less than that of the tMK-CKKS scheme, and the ciphertext size is also slightly reduced. However, in practice, because of the verification mechanism, vMK-CKKS needs to broadcast between clients to verify the correctness of secret fragments, and the client selection strategy requires the client to inform the server of its corresponding resource information, which increases the communication requirement of each round by about 11KB compared with other schemes. Suppose the total number of aggregated rounds is 50. It will incur about 0.53 MB of communication overhead, which is still an order of magnitude smaller than ciphertext. Therefore, the communication overhead of the vMK-CKKS scheme is still smaller than that of the other two MK-CKKS schemes when the number of aggregation rounds is small.

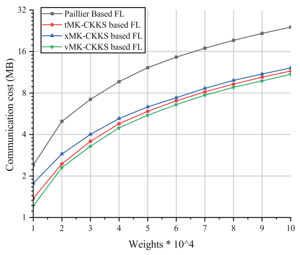


(a) Decryption time cost on server

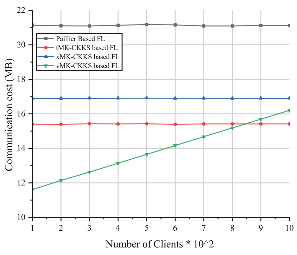


(b) Average computation time cost

Fig. 6. Decryption time cost on server



(a) Communication cost of different encryption schemes



(b) Communication cost with different number of clients

Fig. 7. Communication cost of different encryption schemes

6 Security and Functionality Analysis

6.1 Analysis of vMK-CKKS

In Sect. 6.1, we analyze the multi-key CKKS scheme and prove its security, based on which we will analyze the security of the vMK-CKKS scheme.

Theorem 1. *The vMK-CKKS scheme is semantically secure, based on the hardness of the RLWE problem [3].*

Proof. The security of the vMK-CKKS scheme follows directly from the security of the CKKS homomorphic encryption scheme. We can see that $ct_i = (c_0^{k_i}, c_1^{k_i}) = (v'^{k_i} \cdot \tilde{b} + m_i + e_0^{k_i}, v'^{k_i} \cdot a + e_1^{k_i}) \pmod{q}$, and the decryption share $D_i = s_i \cdot C_{sum_1} + e^* = s_i \cdot \sum_{i=1}^{K} (v'^{k_i} \cdot a + e_1^{k_i}) + e^* \pmod{q}$, these messages are all added with errors. The security of the CKKS scheme relies on the hardness of the Ring Learning with Errors (RLWE) problem. RLWE problem is believed to be hard in the worst-case sense, even in the presence of quantum computers.

The aggregate public key security of the vMK-CKKS scheme is guaranteed by the Feldman threshold secret sharing scheme, which is information-theoretic secure. If the secret is divided into s shares, any $s-1$ shares can not recover the original secret so that the vMK-CKKS scheme can resist a certain threshold of collusion attack.

6.2 Security Analysis of FedCMK

We consider here a four-party scenario, with learners K_1, K_2, K_3, K_4 and federation launcher (i.e. S_1) and federation controller (i.e. S_2).

According to the previous definition, the federation initiator is a trusted third-party server, assuming that the participants K_1, K_2, K_3 are honest and curious, they will abide by the corresponding communication protocol but try to obtain the private information of other participants, while K_4 is a malicious party, it may not transmit the correct information. Let $F = \{S_1, S_2, K_1, K_2, K_3, K_4\}$ be the set consisting of this federated learning framework. We consider potentially several kinds of adversaries $\mathcal{A} = \{\mathcal{A}_{s_2}, \mathcal{A}_{K_1}, \mathcal{A}_{K_2}, \mathcal{A}_{K_3}, \mathcal{A}_{K_4}\}$, where $\mathcal{A}_{K_1}$ represents a possible inference attack by learner K_1, and so on.

If the encryption adopts the vMK-CKKS scheme, now consider the following scenario. Firstly, $S = \{K_1, K_2, K_4\}$ is selected as the learner of this round according to the client selection strategy. Ideally, the trusted federation launcher generates and distributes the secret s. K_1, K_2 and K_4 encrypt the model update information m_1, m_2 and m_4 through the aggregate public key $\tilde{b}$ formed by s and output the ciphertexts ct_1, ct_2 and ct_4. But in the secret distribution phase, K_1, K_2, and K_4 will verify whether their secret shares are correct by broadcasting. Finally, suppose the secret shares of all participants reach the threshold t of the secret sharing scheme. In that case, at least t participants have the correct secret shares, and the federation controller decrypts and outputs the sum of plaintexts m on the premise that at least t participants jointly decrypt. We consider the algorithm to be secure.

6.3 The Security of FedCMK

Here, we perform the security proof of the federated learning framework FedCMK based on the analysis in Sect. 6.2.

Theorem 2. *Any private privacy information of the parties involved in FedCMK will not be inferred, in the presence of honest and curious adversaries* $\mathcal{A} = \{\mathcal{A}_{s_2}, \mathcal{A}_{K_1}, \mathcal{A}_{K_2}, \mathcal{A}_{K_3}\}$.

Proof. We here analyze the effect of inference attacks by semi-honest adversaries on the overall system. In the vMK-CKKS scheme, since the aggregate public key used for encryption is based on the threshold secret sharing technique, neither individual semi-honest federation controllers nor learners can decrypt the ciphertext independently because they cannot reconstruct the secret independently. Therefore, a semi-honest adversary cannot steal the private data of other learners alone.

Theorem 3. *Even if at most n - 1 learners perform a collusion attack, any private privacy information of the parties involved in FedCMK will not be inferred, in the presence of honest and curious adversaries* $\mathcal{A} = \{\mathcal{A}_{s_2}, \mathcal{A}_{K_1}, \mathcal{A}_{K_2}, \mathcal{A}_{K_3}\}$. *(In the vMK-CKKS scheme, n represents the threshold t of secret sharing)*

Proof. We here analyze the impact of collusion attacks among multiple members on the overall system. In the vMK-CKKS scheme, considering the worst case, $t-1$ learners conduct a collusion attack with the federation controller, hoping to infer the private information of the remaining learner. We introduce the model in Sect. 6.4 for illustration, that is, the federation controller S_2 conspires with learners K_2 and K_4 to infer the private information of K_1. Since the vMK-CKKS scheme builds on the VSS scheme, any holder of $t+1$ secret shares can recover it by polynomial modulo q, while the holder of $t-1$ shares cannot. The VSS scheme is based on the discrete logarithm problem, and there is no probability of cracking through the polynomial complexity algorithm, so it is computationally secure. Therefore, $t-1$ malicious attackers cannot obtain the data of other learners through joint collusion.

Theorem 4. *Even if some malicious adversary shares the wrong secret share, it will not derive any private information of the parties involved in FedCMK or break the decryption, in the presence of honest and curious and malicious adversaries* $\mathcal{A} = \{\mathcal{A}_{s_2}, \mathcal{A}_{K_1}, \mathcal{A}_{K_2}, \mathcal{A}_{K_3}\}$.

Proof. The vMK-CKKS scheme is based on Feldman's verifiable secret sharing scheme. After each client receives the secret share, it needs to verify whether z_i satisfies the equation $g^{z_i} = \Pi_{j=0}^{t-1} A_j^{x_i^j}$. The equality is derived as follows:

$$\begin{aligned}\Pi_{j=0}^{t-1} A_j^{x_i^j} \pmod q &= g^{\left(a_j x_i^{t-1} + b_j x_i^{t-1} + c_j x_i^{t-1} + v_j\right)} \pmod q \\ &= g^{z_i} \pmod q\end{aligned} \tag{4}$$

According to the difficulty of discrete logarithm calculation, all parameters are hard to be calculated, so if one party provides the wrong secret share, it can not participate in the final decryption calculation, nor can he steal the data. Since only a threshold of t parties with secret shares is needed for secret reconstruction, the wrong secret shares sent by malicious parties do not affect the final decryption.

7 Conclusion

We have improved xMK-CKKS and tMK-CKKS, the two previous multi-key encryption schemes, and optimized the algorithm for efficiency and security. We have improved the aggregation mode of public keys to better adapt to the characteristics of distributed training of federated learning. The overall federated learning framework is more secure, robust, and efficient.

We evaluate our scheme in terms of accuracy, computation cost, and communication cost and compare our scheme with the mainstream Paillier encryption scheme and several different multi-key CKKS schemes. Experiments show that our federated learning framework is more efficient than the traditional federated learning framework while ensuring accuracy and can conduct secure federated training under the condition of having a trusted server.

However, when the number of clients is large, the verifiable secret sharing mechanism requires broadcasting between clients to verify the correctness of the obtained secret snippets, which can incur significant additional overhead, and the communication between clients may cause potential security problems.

In future work, we may optimize scenarios for large-scale federated learning for large-scale clients to ensure that there is not a large amount of additional communication overhead. Optimizing the client selection mechanism may be an option [13]. At the same time, we hope that our encryption scheme can be more suitable for distributed scenarios, such as federated learning without the participation of a trusted third party and vertical federated learning, where each participant holds different keys.

References

1. Bagdasaryan, E., et al.: How to backdoor federated learning. In: International Conference on Artificial Intelligence and Statistics. PMLR, pp. 2938–2948 (2020)
2. Bonawitz, K., et al.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1175–1191 (2017)
3. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_29
4. Brisimi, T.S., et al.: Federated learning of predictive models from federated electronic health records. Int. J. Med. Inf. **112**, 59–67 (2018)
5. Chen, H., et al.: Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 395–412 (2019)
6. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 409–437. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_15
7. Du, W., et al.: A efficient and robust privacy-preserving framework for cross-device federated learning. In: Complex & Intelligent Systems, pp. 1–15 (2023)
8. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)
9. Fang, H., Qian, Q.: Privacy preserving machine learning with homomorphic encryption and federated learning. Future Internet **13**(4), 94 (2021)
10. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th Annual Symposium on Foundations of Computer Science (SFCS 1987), pp. 427–438. IEEE (1987)
11. Geyer, R.C., Klein, T., Nabi, M.: Differentially private federated learning: a client level perspective. arXiv preprint arXiv:1712.07557 (2017)
12. Jin, C., et al.: Towards End-to-end secure and efficient federated learning for XGBoost (2022)
13. Konečný, J., et al.: Federated learning: strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)

14. Federated Learning: Collaborative machine learning without centralized training data. Publication date: Thursday, April 6 (2017)
15. Li, T., et al.: Federated learning: challenges, methods, and future directions. IEEE Signal Process. Mag. **37**(3), 50–60 (2020)
16. Lyu, L., Yu, H., Yang, Q.: Threats to federated learning: a survey. arXiv preprint arXiv:2003.02133 (2020)
17. Ma, J., et al.: Privacy-preserving federated learning based on multi-key homomorphic encryption. Int. J. Intell. Syst. **37**(9), 5880–5901 (2022)
18. Matsumoto, M., Oguchi, M.: IoT device friendly leveled homomorphic encryption protocols. In: IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), pp. 525–532. IEEE (2022)
19. McMahan, B., et al.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
20. Nishio, T., Yonetani, R.: Client selection for federated learning with heterogeneous resources in mobile edge. In: ICC 2019–2019 IEEE International Conference on Communications (ICC), pp. 1–7. IEEE (2019)
21. Stripelis, D., et al.: Secure federated learning for neuroimaging. arXiv preprint arXiv:2205.05249 (2022)
22. Tan, A.Z., et al.: Towards personalized federated learning. IEEE Trans. Neural Networks Learn. Syst. **32**, 9587–9603 (2022)
23. Yuan, B., Ge, S., Xing, W.: A federated learning framework for healthcare IoT devices. arXiv preprint arXiv:2005.05083 (2020)
24. Zhang, C., et al.: Batchcrypt: efficient homomorphic encryption for cross-silo federated learning. In: Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 2020) (2020)
25. Zhang, J., et al.: PEFL: a privacy-enhanced federated learning scheme for big data analytics. In: IEEE Global Communications Conference (GLOBECOM), pp. 1–6. IEEE (2019)
26. Zhang, X., et al.: A privacy-preserving and verifiable federated learning scheme. In: ICC 2020–2020 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2020)