Aalto University

OPEN ACCESS

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Ma, Jing; Naas, Si-Ahmed; Sigg, Stephan; Lyu, Xixiang

Privacy-preserving federated learning based on multi-key homomorphic encryption

Published in:
International Journal of Intelligent Systems

DOI:
10.1002/int.22818

Published: 01/09/2022

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Ma, J., Naas, S.-A., Sigg, S., & Lyu, X. (2022). Privacy-preserving federated learning based on multi-key homomorphic encryption. *International Journal of Intelligent Systems*, *37*(9), 5880-5901. https://doi.org/10.1002/int.22818

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

ORIGINAL ARTICLE
Journal Section

Privacy-preserving Federated Learning based on Multi-key Homomorphic Encryption

Jing Ma[1,†] | **Si-Ahmed Naas**[2] | **Stephan Sigg**[2] | **Xixiang Lyu**[1,*]

[1]School of Cyber Engineering, Xidian University, Xi'an, Shaanxi, 710071, China

[2]Department of Communications and Networking, Aalto University, Espoo, Uusimaa, 00076, Finland

Correspondence
Xixiang Lyu, School of Cyber Engineering, Xidian University, Xi'an, Shaanxi, 710071, China
Email: xxlv@mail.xidian.edu.cn

Present address
[†]Department of Communications and Networking, Aalto University, Espoo, Uusimaa, 00076, Finland

Funding information
China National Science Foundation, Grant/Award Number: 62072356; The National Key Research and Development Program of Shaanxi, Grant/Award Number: 2019ZDLGY12-08.

With the advance of machine learning and the Internet of Things (IoT), security and privacy have become critical concerns in mobile services and networks. Transferring data to a central unit violates the privacy of sensitive data. Federated learning mitigates this need to transfer local data by sharing model updates only. However, privacy leakage remains an issue. This paper proposes xMK-CKKS, an improved version of the MK-CKKS multi-key homomorphic encryption protocol, to design a novel privacy-preserving federated learning scheme. In this scheme, model updates are encrypted via an aggregated public key before sharing with a server for aggregation. For decryption, a collaboration among all participating devices is required. Our scheme prevents privacy leakage from publicly shared model updates in federated learning and is resistant to collusion between $k < N - 1$ participating devices and the server. The evaluation demonstrates that the scheme outperforms other innovations in communication and computational cost while preserving model accuracy.

KEYWORDS
privacy protection, federated learning, multi-key homomorphic encryption, IoT, smart healthcare

1 | INTRODUCTION

With the popularity of IoT devices, the amount of distributed data has increased dramatically, promoting the vigorous development of machine learning in many domains, including, for instance, smart healthcare, smart home, or traffic accident detection [1, 2, 3]. While the benefit of machine learning is undeniable, it requires large amounts of data to be collected and analyzed at central servers, which may be sensitive to sharing with other parties due to strategic (business) or privacy reasons.

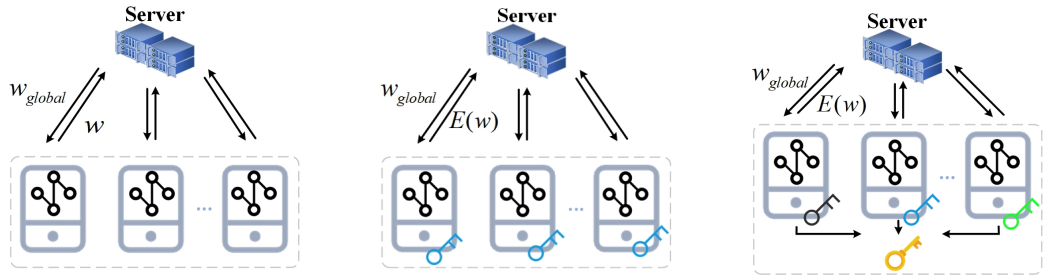
To improve data protection in machine learning, federated learning [4, 5, 6] has been proposed to train a machine learning model while protecting data across multiple distributed devices by sharing the model updates instead of local datasets (Fig. 1(a)). Nevertheless, federated learning still suffers from critical threats as the communicating model updates throughout the training process can reveal sensitive information. From the shared model updates, the adversary can steal the training data [7] or infer "unintended" information such as membership and property [8]. The adversary can also train a Generative Adversarial Network (GAN) to generate prototypical samples of the targeted local training set when only a small percentage of model updates is shared by clients [9]. Any curious device participating in training may pose a threat to the data breach by observing and analyzing these updates and external malicious adversaries can eavesdrop on the communication channel or comprise the server to access the model updates, thus leading to data leakage. These threats can damage personal privacy, bring unpredictable economic and life losses, and hinder data owners from contributing to the development of machine learning. Therefore, a secure and data-protecting federated learning framework is needed.

To fulfill the requirement of data protection in federated learning, this work proposes a data-protecting federated learning scheme based on multi-key homomorphic encryption (MK-HE) (Fig. 1(c)). We improve the existing MK-HE scheme to achieve more robust security and better efficiency, then establish a data-protecting federated learning scheme based on it. Our federated learning scheme guarantees the confidentiality of the model updates to prevent external malicious adversaries and, more importantly, can resist threats from internal curious participants and the collusion between curious participants and the server.

Although many secure federated learning schemes based on cryptographic technologies have been proposed in past years, these solutions are not directly applicable to distributed IoT scenarios or have an inevitable performance loss. For example, secure multi-party computation (SMC) based federated learning [10, 11, 12, 13] usually requires many rounds of interaction between participants to achieve secure aggregation, which is not practical distributed IoT scenarios. Federated learning based on differential privacy (DP) [14, 15, 16] has an inevitable loss of model accuracy due to the noise added. Moreover, federated learning based on homomorphic encryption [17, 18, 19, 20, 21] avoids the model accuracy reduction and complicated interactions between clients while protecting data by enabling the aggregation of encrypted model updates.

All these prior work cannot resist data leakage attacks from curious internal devices as well as collusion attacks between internal devices and the server since all clients share the same public and secret key (Fig. 1(b)). In contrast, xMK-CKKS based data-protecting federated learning achieves more robust security while preserving learning performance. Specifically, our contributions are:

(1) To avoid the risk of data leakage and to omit the need for an additional interactive protocol in the collaborative decryption process of the existing MK-CKKS multi-key homomorphic encryption scheme [22], we develop xMK-CKKS multi-key homomorphic encryption. Specifically, we propose an aggregated public key, the sum of all individual public keys, for encryption. For secure decryption, devices compute their decryption share, which implicitly contains the information of individual secret keys and the aggregated ciphertexts but can be publicly



(a) Federated learning without encryption

(b) Homomorphic encryption (HE)-based federated learning using the same keys for all devices

(c) Multi-key HE-based federated learning using distinct keys. Decryption is possible only by combining all keys

Fig. 1. Different federated learning schemes with/without encryption.

shared to the server. In this way, our xMK-CKKS scheme provides more robust security and avoids an additional interactive decryption protocol. It is suitable for federated learning and other distributed scenarios.

(2) We establish a privacy-preserving federated learning scheme based on xMK-CKKS to achieve secure and efficient sum aggregation of all participants' model updates. Using the FedAvg algorithm, the scheme reduces the communication cost during the whole process. Our scheme is resistant to attacks from internal curious participants as well as external malicious adversaries while achieving accuracy preservation. For a total of N participating devices, it is also robust against collusion attacks between $k < N - 1$ compromised participating devices and the server.

(3) We evaluated and compared the scheme to state-of-the-art federated learning schemes based on homomorphic encryption in a realistic federated learning scenario using Jetson Nano IoT devices. Results show a significant reduction in communication and computation cost while featuring reasonable energy consumption and preserving the model accuracy.

2 | RELATED WORK

Our work is mainly related to prior research conducted in multi-key homomorphic encryption and federated learning, particularly concerning privacy-preserving federated learning schemes.

2.1 | Multi-key homomorphic encryption

Popular additive homomorphic encryption (HE) schemes include Goldwasser and Micali [23], Paillier [24], Damgard and Jurik [25], and Kawachi et al. [26]. The first fully homomorphic encryption (FHE) has been proposed by Gentry [27], which has seen many follow-up improvements to reduce computational load. Homomorphic encryption schemes can be symmetric (same key for encryption and decryption), as well as asymmetric (different keys) [28].

Unlike traditional homomorphic encryption that evaluates the arithmetic circuits of ciphertexts encrypted with the same key, multi-key homomorphic encryption (MK-HE) allows different parties to use different keys for encryption. As a result, decryption requires the collaboration of all participants. López-Al et al. [29] proposed the first MK-FHE scheme known as LTV12, which is based on the NTRU encryption system [30]. It uses relinearization and modu-

lus switching technologies to obtain a leveled multi-key fully homomorphic encryption scheme but suffers from the high computational complexity of decryption operations. While more efficient protocols such as DHS16 [31] have been proposed, NTRU-based MK-FHE schemes are not practical due to their large decryption complexity and high communication load. The use of other types of FHE schemes for MK-FHE has been first discussed by Michael et al. [32]. Further simplifications in the ciphertext extension process and multiple rounds of multi-party computation (MPC) have been introduced in [33]; however, the size of the ciphertext in these schemes increases exponentially with the increase in the number of participants, which causes large communication and storage costs that are infeasible for application in a practical setting. Chen et al. [22] proposed MK-BFV and MK-CKKS, multi-key variants of the fully homomorphic encryption schemes BFV [34, 35] and CKKS [36]. In these schemes, the length of the ciphertext increases linearly with the number of participants. The implementations of MK-BFV and MK-CKKS provide the first experimental results of MK-HE with packed ciphertexts. However, the distributed decryption of a multi-key ciphertext needs an additional secure method, otherwise has the risk of privacy leakage. The proposed scheme xMK-CKKS in this paper provides simple and secure decryption for an aggregated ciphertext generated by only addition.

2.2 | Privacy-preserving federated learning

Federated learning was first introduced by Konečnỳ et al. [4] in 2016, as a distributed machine learning scheme in which distributed devices collaborate with a central coordinator in sharing locally trained updates to a global model for aggregation. To reduce the uplink communication cost in federated learning, structured and sketched updates were introduced. In addition, McMahan et al. proposed Federated Averaging (FedAvg) [37], which achieves a reduction in the communication load by one to two orders of magnitude through iterative model averaging. Their implementation is robust to unbalanced and not independent and identically distributed (non-iid) data and further obfuscates the shared information through aggregated model updates.

Despite this, concerns about information leakage and privacy issues of federated learning have been raised. For example, Aono et al. [17] pointed out an adversary capable of observing only a small fraction of the shared gradients can lead to information leakage and data privacy issues for the distributed devices sharing their updates. Indeed, Melis et al. [8] demonstrated that in distributed federated learning, even if only model update information is shared, sensitive information of the participant may be leaked. It is even possible to recover original training data from the publicly shared gradients [7].

To address this weakness and to enhance the privacy protection of federated learning, cryptographic methods have been proposed, such as multi-party computation [38], differential privacy [39, 40, 41], as well as homomorphic encryption [42], to protect the shared model updates.

In secure multi-party computation, a group of mutually distrustful parties $u \in \mathcal{U}$ collaborate to compute an aggregated sum $\mathcal{A} = \sum_{u \in \mathcal{U}} x_u$ for their private values x_u without revealing x_u. The scheme of secure aggregation has been adapted for federated learning, for instance, in [10, 11], to ensure that globally, an individual update from any distributed party cannot be derived from the aggregation of all updates. Related to this, Li et al. [12] proposed to chain local gradient updates across all remote participating devices in order to mask the individual values. However, the protocol requires secure and reliable communication channels and honest devices. In addition, the greedy generation of the chain of devices might fail in partly disconnected topologies. Xu et al. [13] proposed privacy-preserving and verifiable collaborative learning, which uses double masking to ensure the confidentiality of the individual gradients. In particular, distributed devices secretly establish random numbers for each device pair to mask their local data. However, the masking process is computationally expensive and constraints practical use by requiring that all devices are within pairwise communication range. Furthermore, a trusted third party is needed to establish asymmetric key

pairs initially.

Differential privacy [14] may protect the privacy of individual data of deep learning algorithms via a differentially private gradient descent mechanism that adds noise to gradient updates. Geyer et al. [15] developed an implementation of differential privacy for federated learning, which realizes client differential privacy guarantees at the cost of severely reduced model performance.

Dowlin et al. [19] presented a neural network based on homomorphic encryption [29]. In particular, the model inference from a pre-trained network is computed on encrypted data by an untrusted party. Network training, however, has to be conducted offline and does not involve the untrusted party. Improved schemes have been proposed in [20], [17] using additive homomorphic encryption to protect model updates, by Zhang et al. [18] exploiting privacy-preserving and verifiable federated learning using the Paillier cryptosystem [24], as well as by Froelicher et al. [21] who proposed a decentralized system for privacy-conscious statistical analysis on distributed datasets by applying the ElGamal Elliptic Curve additive homomorphic cryptosystem [43]. In these approaches, all participating devices share the same encryption and decryption key. As a result, private information may leak among devices. Furthermore, any curious party colluding with the server will breach the privacy of other parties. Hao et al. [16] proposed an efficient and privacy-enhanced federated learning (PEFL) scheme. PEFL can resist collusion between an adversary and multiple devices by applying differential privacy and additive homomorphic encryption. However, the model accuracy will be seriously degraded when the collusion rate is greater than one-half. In contrast, in this paper, we enhance the privacy protection of federated learning by applying multi-key homomorphic encryption.

3 | PRELIMINARIES

We propose xMK-CKKS, a multi-key homomorphic encryption scheme based on the MK-CKKS homomorphic encryption, to increase privacy protection in distributed machine learning, specifically in federated learning. In the following, we discuss our assumptions on federated learning and introduce the concept of the MK-CKKS homomorphic encryption scheme.

3.1 | Federated learning

Federated Learning is a distributed machine learning concept. It enables the training of a machine learning model from data that is kept at decentralized devices. As shown in Fig. 1(a), a remotely participating device first downloads the global model from the server, trains it with local data, and then summarizes the results into a model update (model weights or gradients) which is returned to the server. At the server, model updates from remote devices are aggregated into a new global model, to be shared again. The server and remote devices iteratively exchange model updates and the global model computed during the training.

Stochastic Gradient Descent (SGD) can be naively applied in distributed learning but requires many communication rounds between distributed devices and the server. To address this, McMahan et al. [37] proposed FederatedAveraging (FedAvg, Algorithm 1), which exploits iterative model averaging in federated learning to reduce the communication load as follows. For N remote devices $d_i, i \in \{1, \ldots, N\}$, a fraction of these devices $\hat{N}$ performs local model training in each round. In particular, in each round, $\hat{N}$ remote devices d_i conduct L training epochs on their local datasets before sharing the model parameters w^i_{t+1} with the server. In particular, when $L = 1$, the algorithm is referred to as FedSGD. Then the server computes an average of all devices' model parameters as the updated global model. This process iterates until the global model converges.

Algorithm 1 FederatedAveraging(FedAvg)

Let η be the learning rate, and $\mathcal{B}$ be the minibatch size for local model training.

The server initializes w_0 // Global model parameters

for each round $t = 1, 2, \ldots$ **do**

$S_t \leftarrow$ (random set of remote devices d_i)

for each remote device $d_i \in S_t$ in parallel **do**

$w_{t+1}^i \leftarrow \text{DeviceUpdate}(d_i, w_t^i)$

$w_{t+1} \leftarrow \frac{1}{\hat{N}} \sum_{i=1}^{\hat{N}} w_{t+1}^i$

Remote devices compute their local updates.

DeviceUpdate(d_i, w): $\mathcal{B} \leftarrow$ (select batches of size $\mathcal{B}$) //Run by each remote device d_i locally

for local epochs $I = 1, \ldots, L$ **do**

for batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$ // update the weight using the loss $\ell(w; b)$ and the learning rate η

return w to server

3.2 | Multi-key Homomorphic encryption

An encryption scheme $E(k, x)$ for a key k and an input x is called homomorphic if for the encryption algorithm E and operation f, there is an efficient algorithm G such that

$$E\left(k, f\left(x_1, \ldots, x_n\right)\right) = G\left(k, f\left(E(x_1), \ldots, E(x_n)\right)\right). \tag{1}$$

If equation (1) only holds for either addition or multiplication, the scheme is called partially homomorphic encryption. It is fully homomorphic encryption if it holds for both addition and multiplication (FHE).

Multi-key homomorphic encryption (MK-HE) allows different participants to use different keys for encryption. The aggregated ciphertext obtained after performing polynomial operations on different individual ciphertexts can only be jointly decrypted by combining the respective secret keys associated with these ciphertexts. MK-CKKS is an MK-HE scheme [22] which is a multi-key variant of the CKKS FHE scheme [36] that supports approximate fixed-point arithmetic. Since homomorphic multiplication is not involved in our federated learning mechanism, we introduce the additive homomorphism of MK-CKKS only. MK-CKKS is a Ring Learning with Errors [44] (RLWE)-based homomorphic encryption scheme. Let

$$R = \mathbb{Z}[X]/(X^n + 1)$$

be the cyclotomic ring in which n has the power of two dimension, and $\mathbb{Z}[X]$ is the polynomial ring with integer coefficients and the elements in R satisfy $X^n = -1$. $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ is the residue ring of R with coefficients modulo an integer q. For parameters (n, q, χ, ψ), our RLWE assumption is that, given polynomials of the form $(a, b = s \cdot a + e) \in R_q^2$, the term b is computationally indistinguishable from uniformly random elements of R_q when a is chosen uniformly at random from R_q, s is chosen from the key distribution χ over R_q, and e is drawn from the error distribution ψ over R [44].

We denote vectors in bold and use $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ to denote the dot product of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. $x \leftarrow \Gamma$ denotes the sampling of x according to the distribution Γ. λ denotes the security parameter throughout the paper: all known valid

attacks against the cryptographic scheme under scope should take $\Omega(2^\lambda)$ bit operations. $g \in Z^d$ is an integral vector, called the gadget vector. MK-CKKS assumes the Common Reference String (CRS) model so all devices share a random polynomial vector $\boldsymbol{a} \leftarrow U(R_q^d)$, here $U(\cdot)$ represents the uniform distribution. Let $\boldsymbol{sk_i} = (1, s_i)$ for the secret key s_i, $\overline{\boldsymbol{sk}} = (1, s_1, \ldots s_N)$ for the concatenation of multiple secret keys. Let $\boldsymbol{ct_i} = (c_0^{d_i}, c_1^{d_i})$ be the ciphertext of plaintext m_i from remote device d_i, $i = 1, \ldots, N$.

Setup. For a given security parameter λ, set the RLWE dimension n, ciphertext modulus q, key distribution χ and error distribution ψ over R. Generate a random vector $\boldsymbol{a} \leftarrow U(R_q^d)$. Return the public parameter $(n, q, \chi, \psi, \boldsymbol{a})$. A remote device d_i generates its secret key $s_i \leftarrow \chi$, and computes its public key as $\boldsymbol{b}_i = -s_i \cdot \boldsymbol{a} + \boldsymbol{e_i} \in R_q^2$, here $\boldsymbol{e_i}$ is an error vector drawn from the error distribution ψ over R.

Encoding and decoding. Before encryption, a complex number is first expanded into a vector and then encoded as a polynomial of ring R based on the complex canonical embedding map. The decoding transfers a polynomial into a complex vector after decryption.

Encryption. After encoding a message vector into a plaintext m_i, which is an element of a cyclotomic ring, d_i then encrypts m_i as a ciphertext $\boldsymbol{ct_i} = (c_0^{d_i}, c_1^{d_i})$ where $c_0^{d_i} = v_i \cdot b_i + m_i + e_0^{d_i} \pmod q$ and $c_1^{d_i} = v_i \cdot a + e_1^{d_i} \pmod q$. Here $a = \boldsymbol{a}[0]$ and $b_i = \boldsymbol{b}_i[0]$, $v_i \leftarrow \chi$ and $e_0^{d_i}, e_1^{d_i} \leftarrow \psi$. Tiny errors are inserted to ensure security, and the rounding operation can remove them after carrying out homomorphic operations. In MK-CKKS, an additive ciphertext associated to N different parties is of the form $\boldsymbol{C_{sum}} \stackrel{\text{def}}{=} \left(\sum_{i=1}^{N} c_0^{d_i}, c_1^{d_1}, c_1^{d_2}, \ldots, c_1^{d_N} \right) \in R_q^{N+1}$.

Decryption of individual ciphertext. d_i computes a dot product of $\boldsymbol{sk_i} = (1, s_i)$ and $\boldsymbol{ct_i} = (c_0^{d_i}, c_1^{d_i})$ as follows.

$$\begin{aligned} <\boldsymbol{ct_i}, \boldsymbol{sk_i}> \pmod q &= c_0^{d_i} + c_1^{d_i} \cdot s_i \pmod q \\ &= v_i \cdot b_i + m_i + e_0^{d_i} + v_i \cdot a \cdot s_i + e_1^{d_i} \cdot s_i \pmod q \\ &= v_i \cdot (-s_i \cdot a + e_i) + m_i + e_0^{d_i} + v_i \cdot a \cdot s_i + e_1^{d_i} \cdot s_i \pmod q \\ &= m_i + v_i \cdot e_i + e_0^{d_i} + e_1^{d_i} \cdot s_i \pmod q \\ &\approx m_i \end{aligned}$$

Additive homomorphism. Let $\boldsymbol{ct_i} = (c_0^{d_i}, c_1^{d_i})$ and $\boldsymbol{ct_j} = (c_0^{d_j}, c_1^{d_j})$ be two ciphertexts of plaintext messages m_i and m_j from remote devices d_i and d_j. The sum of the ciphertexts is $\boldsymbol{C_{sum}} \stackrel{\text{def}}{=} (c_0^{d_i} + c_0^{d_j}, c_1^{d_i}, c_1^{d_j})$. It can be decrypted by computing a dot product of $\boldsymbol{C_{sum}}$ and $\overline{\boldsymbol{sk}} = (1, s_i, s_j)$. The correctness is proved as follows:

$$\begin{aligned} <\boldsymbol{C_{sum}}, \overline{\boldsymbol{sk}}> \pmod q &= (c_0^{d_i} + c_0^{d_j}) + c_1^{d_i} \cdot s_i + c_1^{d_j} \cdot s_j \pmod q \\ &= (c_0^{d_i} + c_1^{d_i} \cdot s_i) + (c_0^{d_j} + c_1^{d_j} \cdot s_j) \pmod q \\ &\approx m_i + m_j \end{aligned}$$

Decryption of sum. The distributed decryption based on noise flooding is introduced in MK-CKKS since it is not reasonable to assume that any party holds multiple secret keys. The decryption consists of two algorithms: partial decryption and merge.

MK – CKKS.PartDec$(c_1^{d_i}, s_i)$**:** Given a polynomial $c_1^{d_i}$ and a secret s_i, sample an error $e_i^* \leftarrow \phi$ and return $\mu_i = c_1^{d_i} \cdot s_i + e_i^* \pmod q$.

MK – CKKS.Merge$(\sum_{i=1}^{N} c_0^{d_i}, \{\mu_i\}_{1 \le i \le N})$**:** Compute and return $\mu = \sum_{i=1}^{N} c_0^{d_i} + \sum_{i=1}^{N} \mu_i \pmod q \approx <\boldsymbol{C_{sum}}, \overline{\boldsymbol{sk}}> \pmod q$.

Here e_i^* is generated from error distribution ϕ which has a larger variance than the standard error distribution ψ.

4 | MULTI-KEY HOMOMORPHIC ENCRYPTION FOR FEDERATED LEARNING

MK-CKKS is not directly applicable to federated learning since the server would be capable of decrypting the individual model updates and learning about the private data. In this section, We improve the MK-CKKS protocol so that the server becomes only able to decrypt the aggregation of the shared encrypted model updates, thereby obfuscating the individual model updates with updates from all other remote participating devices. The proposed xMK-CKKS simplifies the decryption process of MK-CKKS while maintaining strong privacy protection. Based on xMK-CKKS, we propose a privacy-preserving federated learning scheme.

4.1 | Threat model

In our work, we apply homomorphic encryption in a federated learning scenario to protect data privacy. In this context, we assume that the server and all remotely participating devices are honest-but-curious. This means that they follow the scheme honestly but intend to infer the private information of other devices from the information shared during the execution of the protocol. We further assume that collusion may exist between compromised devices and the server. In particular, we consider an attack in which $k < N-1$ devices collude together with the server to jointly attack particular devices, where N is the total number of devices and k is the number of compromised colluding devices.

4.2 | xMK-CKKS

As described in Section 3, the decryption of the sum of the ciphertexts requires two steps, partial decryption and merge. In a federated learning scenario, each device sends encrypted model updates to the server for aggregation. In this way, the server obtains $\boldsymbol{ct_i} = (c_0^{d_i}, c_1^{d_i})$. If $\mu_i = c_1^{d_i} \cdot s_i + e_i^* \pmod q$ were also to be shared, the server could directly decrypt m_i (cf. equation (2)).

$$c_0^{d_i} + \mu_i = c_0^{d_i} + c_1^{d_i} \cdot s_i + e_i^* \approx m_i \pmod q \tag{2}$$

To resolve this problem, the server shall not obtain $c_0^{d_i}$ and μ_i at the same time. For instance, either $\sum_{i=1}^{N} c_0^{d_i}$ or $\sum_{i=1}^{N} \mu_i$ may instead be computed collaboratively by the remote devices. However, this would introduce the risk of private information m_i leaking if devices and the server collude.

xMK-CKKS. To avoid privacy leakage during decryption, we propose xMK-CKKS. In xMK-CKKS, an aggregated public key is computed for encryption by aggregating the public keys of all devices. For decryption, the server requires decryption share D_i computed by each device d_i. D_i combines the sum over all ciphertexts as well as the secret key s_i and error term e_i^* of d_i (see equation (6)). The following methods define the xMK-CKKS scheme in detail.

Setup(1^λ)**:** Given a security parameter λ, set the RLWE dimension n, ciphertext modulus q, key distribution χ and error distribution ψ over R. Generate a random vector $\boldsymbol{a} \leftarrow U(R_q^d)$. Return the public parameters $(n, q, \chi, \psi, \boldsymbol{a})$.

KeyGen$(n, q, \chi, \psi, \boldsymbol{a})$**:** Each device d_i $(1 \le i \le N)$ generates its secret key s_i and computes its individual public key

$\boldsymbol{b}_i = -s_i \cdot \boldsymbol{a} + \boldsymbol{e_i} \pmod q$. We define the aggregated public key $\widetilde{\boldsymbol{b}}$ as

$$\widetilde{\boldsymbol{b}} = \sum_{i=1}^{N} \boldsymbol{b}_i = \sum_{i=1}^{N} (-s_i) \cdot \boldsymbol{a} + \sum_{i=1}^{N} \boldsymbol{e_i} \pmod q \tag{3}$$

Here $\boldsymbol{e_i} \leftarrow \psi^d$.

$\mathrm{Enc}(m_i, \widetilde{\boldsymbol{b}}, \boldsymbol{a})$**:** The plaintext m_i of a device d_i is encrypted as

$$\boldsymbol{ct_i} = (c_0^{d_i}, c_1^{d_i}) = (v^{d_i} \cdot \widetilde{b} + m_i + e_0^{d_i}, v^{d_i} \cdot a + e_1^{p_i}) \pmod q \tag{4}$$

Here $a = \boldsymbol{a}[0]$, $\widetilde{b} = \widetilde{\boldsymbol{b}}[0]$, $v \leftarrow \chi$, $e_0^{d_i}, e_1^{d_i} \leftarrow \psi$.

$\mathrm{Add}(\boldsymbol{ct_1}, \ldots, \boldsymbol{ct_N})$**:** The sum of all ciphertexts is

$$\begin{aligned} \boldsymbol{C_{sum}} &= \sum_{i=1}^{N} \boldsymbol{ct_i} \triangleq (C_{sum_0}, C_{sum_1}) \\ &= \left(\sum_{i=1}^{N} c_0^{d_i}, \sum_{i=1}^{N} c_1^{d_i} \right) \\ &= \left(\sum_{i=1}^{N} (v^{d_i} \cdot \widetilde{b} + m_i + e_0^{d_i}), \sum_{i=1}^{N} (v^{d_i} \cdot a + e_1^{d_i}) \right) \pmod q \end{aligned} \tag{5}$$

$\mathrm{Dec}(\boldsymbol{C_{sum}}, s_1, \ldots, s_N)$**:** Each device d_i computes its decryption share D_i

$$D_i = s_i \cdot C_{sum_1} + e_i^* = s_i \cdot \sum_{i=1}^{N} (v^{d_i} \cdot a + e_1^{d_i}) + e_i^* \pmod q \tag{6}$$

Here $e_i^* \leftarrow \psi$.

Then, the sum of all plaintexts can be recovered as follows.

$$\begin{aligned} & C_{sum_0} + \sum_{i=1}^{N} D_i \mod q \\ = \; & C_{sum_0} + \sum_{i=1}^{N} s_i \cdot C_{sum_1} + \sum_{i=1}^{N} e_i^* \mod q \\ = \; & \sum_{i=1}^{N} (v^{d_i} \cdot \widetilde{b} + m_i + e_0^{d_i}) + \sum_{i=1}^{N} s_i \cdot \sum_{i=1}^{N} (v^{d_i} \cdot a + e_1^{d_i}) + \sum_{i=1}^{N} e_i^* \mod q \\ = \; & -\sum_{i=1}^{N} v^{d_i} \cdot s_i \cdot a + \sum_{i=1}^{N} v^{d_i} \cdot \sum_{i=1}^{N} e_i + \sum_{i=1}^{N} m_i + \sum_{i=1}^{N} e_0^{d_i} + \sum_{i=1}^{N} v^{d_i} \cdot s_i \cdot a + \sum_{i=1}^{N} \left(s_i \cdot e_1^{d_i} + e_i^* \right) \mod q \\ = \; & \sum_{i=1}^{N} m_i + \sum_{i=1}^{N} v^{d_i} \cdot \sum_{i=1}^{N} e_i + \sum_{i=1}^{N} e_0^{d_i} + \sum_{i=1}^{N} \left(s_i \cdot e_1^{d_i} + e_i^* \right) \mod q \\ \approx \; & \sum_{i=1}^{N} m_i \end{aligned} \tag{7}$$

In xMK-CKKS, the aggregated public key is computed for encryption. The decryption requires each device to compute its decryption share. The decryption share implicitly contains the information of individual secret key of each

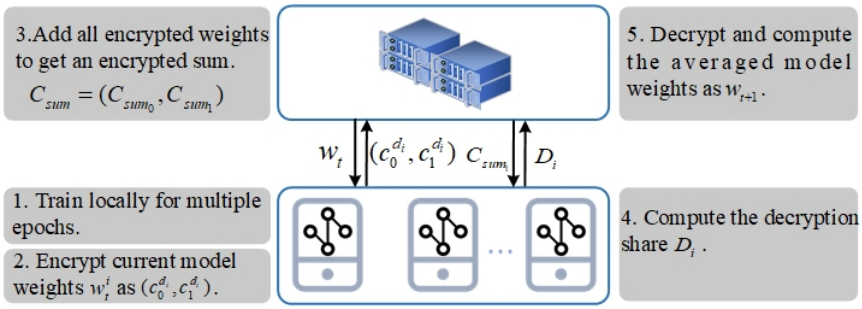


Fig. 2. Privacy-preserving federated learning based on xMK-CKKS multi-key homomorphic encryption

participant and the aggregated ciphertexts C_{sum}, and an error is added for security, thus is useless to decrypt any other ciphertext, including individual ciphertext. Therefore, the risk of privacy leakage during decryption is mitigated by the protocol. Consequently, in federated learning and similar collaborative learning scenarios, xMK-CKKS provides stronger security than MK-CKKS. We show in section 5 that it is also robust against any collusion between $k < N - 1$ compromised devices and the server. Note that xMK-CKKS does not require any interaction among devices and is suited to scenarios in which devices are not fully interconnected.

4.3 | Privacy-preserving federated learning based on xMK-CKKS

We propose a privacy-preserving federated learning scheme based on xMK-CKKS. As a federated learning mechanism, we apply the FedAvg scheme (cf. section 3.1) with the fraction of remotely participating devices as 1 (all devices contribute to model training in each round).

The complete model training process consists of multiple aggregation rounds between the server and the devices. Fig. 2 details the process of one aggregation round. In each aggregation round t, all devices obtain the current global model from the server, train it for multiple epochs on local data (step 1) and encrypt the resulting local model weights with the aggregated public key $\widetilde{\boldsymbol{b}}$ (step 2). All remote devices then send the ciphertexts to the server, where they are aggregated as the sum over all encrypted model weights (step 3). In order to obtain the plaintext from this encrypted sum, the server requires all devices to compute their decryption shares D_i (step 4). After obtaining the decryption shares from all devices, the server merges them with the ciphertext to decrypt the encrypted sum over all shared model weights (step 5). In particular, after decryption, the server obtains the sum of all devices' weights $\sum_{i=1}^{N} w_t^i$ before computing averaged weights w_{t+1} as the new global model weights for the next aggregation round. This procedure (steps 1 to 5) is executed iteratively until the global model converges. We introduce each step in detail below.

Setup: For a given security parameter λ, set the RLWE dimension n, ciphertext modulus q, key distribution χ and error distribution ψ over R. Generate a random vector $\boldsymbol{a} \leftarrow U(R_q^d)$. Return the public parameters $(n, q, \chi, \psi, \boldsymbol{a})$. Each device d_i samples a secret key $s_i \leftarrow \chi$ an error vector $\boldsymbol{e_i} \leftarrow \psi^d$, then computes its public key $\boldsymbol{b}_i = -s_i \cdot \boldsymbol{a} + \boldsymbol{e_i} \pmod q$. Then all participating devices collaboratively compute the aggregated public key as

$$\widetilde{\boldsymbol{b}} = \sum_{i=1}^{N} \boldsymbol{b}_i = \sum_{i=1}^{N} (-s_i) \cdot \boldsymbol{a} + \sum_{i=1}^{N} \boldsymbol{e_i} \pmod q \tag{8}$$

Step 1: Local training: In each aggregation round t, each device d_i first downloads the current global model weights w_t from the server. Every device applies the model optimization algorithm, for instance, SGD or Adam, to train the obtained global model on its local data. After multiple training epochs, each device d_i derives a local model with weights w_t^i.

Step 2: Encryption of model weights: Let $m_i \in R$ be an encoded plaintext input of w_t^i and let $a = \boldsymbol{a}[0]$, $\widetilde{b} = \widetilde{\boldsymbol{b}}[0]$. Sample $v^{d_i} \leftarrow \chi$ and $e_0^{d_i}, e_1^{d_i} \leftarrow \psi$. Compute the ciphertext

$$\boldsymbol{ct^{d_i}} = (c_0^{d_i}, c_1^{d_i}) = (v^{d_i} \cdot \widetilde{b} + m_i + e_0^{d_i}, v^{d_i} \cdot a + e_1^{d_i}) \pmod q \tag{9}$$

Then, each device d_i shares $\boldsymbol{ct^{d_i}} = (c_0^{d_i}, c_1^{d_i})$ with the server.

Step 3: Computation of the homomorphic sum: After receiving encrypted model weights from all devices, the server aggregates them by adding them together to get an encrypted sum $\boldsymbol{C_{sum}}$. The server then publishes C_{sum_1} to all devices.

$$\boldsymbol{C_{sum}} = \sum_{i=1}^{N} \boldsymbol{ct^{d_i}} = \left(\sum_{i=1}^{N} c_0^{d_i}, \sum_{i=1}^{N} c_1^{d_i}\right) \triangleq (C_{sum_0}, C_{sum_1}) \pmod q \tag{10}$$

Step 4: Computation of the decryption share: All the devices d_i are required to decrypt the multi-key ciphertext. For this, each device samples $e_i^* \leftarrow \psi$, then computes its decryption share D_i and sends it to the server.

$$D_i = s_i \cdot C_{sum_1} + e_i^* = s_i \cdot \sum_{i=1}^{N} (v^{d_i} \cdot a + e_1^{d_i}) + e_i^* \pmod q \tag{11}$$

Step 5: Model aggregation: After receiving all decryption shares, the server merges all decryption shares $D_i, i \in \{1, \ldots, N\}$ with $C_{sum_0} = \sum_{i=1}^{N} c_0^{d_i}$ to recover the plaintext.

$$\sum_{i=1}^{N} m_i \approx C_{sum_0} + \sum_{i=1}^{N} D_i \pmod q \tag{12}$$

Finally, the server decodes $\sum_{i=1}^{N} m_i$ to obtain the sum over the weights $\sum_{i=1}^{N} w_t^i$ before computing the averaged weights w_{t+1} as the updated global model weights for the next aggregation round.

$$w_{t+1} = \frac{1}{N} \sum_{i=1}^{N} w_t^i \tag{13}$$

5 | SECURITY AND FUNCTIONALITY ANALYSIS

We discuss how our scheme guarantees privacy for the data hosted at distributed devices by ensuring the confidentiality of the model weights. In particular, the following theorems describe the security of the scheme for various potential adversaries. Then we show the functionality of our scheme by comparing it to other latest privacy-preserving federated learning schemes.

5.1 | Security analysis

Theorem 1. Security against an honest-but-curious server: In our scheme, an honest-but-curious server cannot infer any private information about the devices' data.

Proof: In our xMK-CKKS based federated learning scheme, remotely participating devices d_i send two types of information to the server. First, in step 2, d_i shares ciphertext $\boldsymbol{ct^{d_i}}$ with the server, which is encrypted by xMK-CKKS. Then, in step 4, d_i sends its decryption share D_i to the sever. Here we have $\boldsymbol{ct^{d_i}} = (c_0^{d_i}, c_1^{d_i})$, where $c_0^{d_i} = v^{d_i} \cdot \widetilde{b} + m_i + e_0^{d_i} \pmod q$ and $c_1^{d_i} = v^{d_i} \cdot a + e_1^{d_i} \pmod q$, $D_i = s_i \cdot C_{sum_1} + e_i^* = s_i \cdot \sum_{i=1}^{N} (v^{d_i} \cdot a + e_1^{d_i}) + e_i^* \pmod q$. All messages contain an added error to guarantee security according to RLWE assumption. RLWE guarantees that the $c_0^{d_i}$ and D_i are computationally indistinguishable from uniformly random elements of R_q. They do not leak any information of m_i and s_i to the server S. After collaborative decryption, the server can only get a sum of all model weights, which does not leak information about the individual model weights.

Therefore, our scheme can ensure the confidentiality of individual model weights, thereby guaranteeing the privacy of the data distributed at remote devices. The server cannot infer any private information about the device from the information it receives.

Theorem 2. Security against honest-but-curious distributed devices: In our scheme, an honest-but-curious device cannot infer any private information by stealing the shared information of other devices.

Proof: In our scheme, the model updates of each device d_i are encrypted via xMK-CKKS, which is based on RLWE. Each device individually chooses a secret key s_i to establish a public key $\boldsymbol{b_i}$. In addition, all devices collaboratively compute an aggregated public key to encrypt their model weights. An error is added to the decryption share to protect the secret key of each device. Therefore, an honest-and-curious device cannot infer any information by stealing the information uploaded by other devices.

Theorem 3. Security against the collusion between compromised devices and the server: Collusion between $k < N-1$ devices and the server does not leak information about model updates from other devices, where N is the total number of devices and k is the number of compromised colluding devices.

Proof: In our scheme, model updates of each device are encrypted by the aggregated public key $\widetilde{\boldsymbol{b}} = \sum_{i=1}^{N} \boldsymbol{b}_i = \sum_{i=1}^{N} (-s_i) \cdot \boldsymbol{a} + \sum_{i=1}^{N} \boldsymbol{e_i} \pmod q$ and shared with the server. Then, the server computes a sum over the model weights. The decryption of the individual encrypted weights $\boldsymbol{ct^{d_i}}$ and the encrypted sum $\boldsymbol{C_{sum}}$ can only be done by combining the decryption shares of all devices.

The first collusion attack is that colluding parties try to infer m_i of the victim d_i from its individual ciphertext $\boldsymbol{ct^{d_i}}$. In the worst case, we assume there is a victim d_i, and other $N-1$ devices collude with the server S. A possible attempt is to compute $c_1^{d_j} \cdot s_j$ for $j \neq i$, then to merge them with $c_0^{d_i}$.

$$\begin{aligned} c_0^{d_i} + \sum_{j \neq i}^{N} c_1^{d_1} \cdot s_i \pmod q &= v^{d_i} \cdot \widetilde{b} + m_i + e_0^{d_i} + \sum_{j \neq i} (v^{d_j} \cdot a + e_1^{d_j}) \cdot s_j \pmod q \\ &= -\sum_{i=1}^{N} v^{d_1} \cdot s_i \cdot a + m_i + e_0^{d_i} + \sum_{j \neq i} v^{d_j} \cdot s_j \cdot a + \sum_{j \neq i} e_1^{d_j} \cdot s_j \pmod q \\ &= -v^{d_i} \cdot s_i \cdot a + m_i + e_0^{d_i} + \sum_{j \neq i} e_1^{d_j} \cdot s_j \end{aligned} \tag{14}$$

The result is roughly equivalent to a partial ciphertext encrypted by the individual public key b_i, thereby having no privacy risk. Even if the other devices collude and own the secret keys $s_j, j \neq i$, they cannot jointly decrypt the

Table (1) Comparison of existing privacy-preserving federated learning schemes and xMK-CKKS based federated learning scheme

Scheme	Aono et al. [17]	Keith et al. [11]	Zhang et al. [18]	PEFL [16]	Our scheme
Condentiality of model updates	✓	✓	✓	✓	✓
Resistance to collusion	×	✓	×	✓	✓
Non-interaction among participants	✓	×	✓	✓	✓
Non-secure communication channel	×	✓	✓	✓	✓
Model accuracy preservation	✓	✓	✓	×	✓

individual ciphertext encrypted with b_i by the victim d_i.

We note that the colluding parties might try to infer m_i from the decrypted aggregation result $\sum_{i=1}^{N} m_i$. We argue that the attack cannot be successful as long as there are two uncompromised devices. In the worst case, the $N-2$ colluding devices subtract their plaintexts m_i from the sum $\sum_{i=1}^{N} m_i$. The result is the sum of plaintexts of two uncompromising devices, therefore has no risk of privacy leakage for individual data.

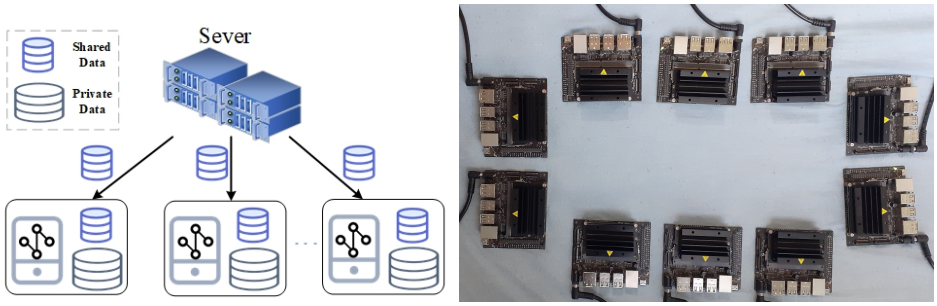
Therefore, our protocol can resist collusion between $k < N-1$ compromised devices and the server.

5.2 | Functionality analysis

Our scheme enhances the privacy protection of federated learning by applying multi-key homomorphic encryption. Table (1) compares our scheme and the latest privacy-preserving federated learning schemes. Although the scheme proposed in [11] and PEFL [16] are resistant to collusion between compromised devices and the server, [11] requires many interactions among participants to build secret shares, and the model's accuracy of PEFL will be seriously degraded when the collusion ratio is greater than half. The privacy-preserving deep learning schemes based on additive homomorphic encryption [17, 18] are vulnerable to collusion attacks between compromised devices and the server.

6 | EVALUATION

This section evaluates the proposed xMK-CKKS based federated learning scheme on a smart healthcare scenario, particularly elderly-fall detection. To validate the potency of our scheme, we compare three schemes: traditional federated learning without encryption, federated learning based on MK-CKKS, and federated learning based on Paillier homomorphic encryption. The traditional federated learning has no additional privacy protection for model updates, which risks privacy leakage. Federated learning based on MK-CKKS encrypts model updates by MK-CKKS but has privacy risks due to the noise flooding technique for decryption. For federated learning based on Paillier, we use Paillier to encrypt model weights, different from the privacy-preserving deep learning scheme proposed by Le et al. [17] which uses Paillier to encrypt the gradients. However, in Paillier, all devices share the same secret and public keys, which compromises privacy. We then report fall detection model accuracy, communication cost, energy consumption, and computational cost during the whole training process.



(a) Distribution strategy to remote devices. The server shares a small fraction of the data to decrease weights divergences

(b) Environment testbed setting with 10 Jetson Nano IoT devices

Fig. 3. Illustration of the experimental setting and distribution strategy utilized in our case study. 10 Jetson Nano IoT devices are utilized as remote devices storing the data while a server is computing the model aggregations.

Table (2) Data Distribution for each distributed device in acceleration samples.

Node	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	Test set
Data size	28K	27K	28K	27K	28K	25K	28K	26K	27K	27K	37K

6.1 | Experimental Setup

We set up 10 Jetson Nano nodes as our IoT devices, with 128 NVIDIA CUDA® cores, Quad-core ARM CortexA57 MPCore processor, 4 GB 64-bit LPDDR4 RAM, and 64 GB of storage (see example Fig. 3(b)). The Jetson Nano IoT devices are compatible with various machine learning frameworks and are mainly used for model training. We employ as a server a laptop of Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz (8 CPUs). We use the TensorFlow 2.0 framework and Keras to build our CNN baseline.

Dataset. We focus on fall detection for elderly persons as universally relevant health and general well-being problem. In particular, we employ a multimodal dataset called "UP-FALL" [45] which includes 17 healthy young individuals. Each subject performs 10 different activities and falls with three trials. All the falls activities are recorded in 10-seconds duration. The data is collected from wearable and ambient sensors and vision devices. In our evaluation, we utilize the dataset from the data of the accelerometer sensors of 10 devices. We allocate one Jetson Nano IoT device with the data of one acceleration sensor. In practice, the data generated by IoT devices is exclusive, and the accuracy of federated learning with not independent and identically distributed (non-iid) data significantly decreases due to weight divergence. For our scheme, we share a small amount of data called the 'global set' with all devices as suggested in [46] (Fig. 3(a) depicts the process). In Table (2), we detail the size of data distributed to each device which is on average approximately 25,000 samples per device. To evaluate the model accuracy, we construct a global testing set consisting of a subset of around 37,000 samples where no redundancy to data is used for the global testing set and with the data of the distributed devices.

Model and implementation. A fall detection model is trained to recognize five types of falls, 1) *falling forward using hands*, 2) *falling forward using knees*, 3) *falling backward*, 4) *falling sitting in an empty chair*, and 5) *falling sideward*. If the activity belongs to no class, it classifies as *unknown activity*. To recognize different types of falls, we employ the "categorical cross-entropy" as a loss function. The model was trained with Adam optimizer [47] at a learning rate of 0.01 with 1, 20 local epochs in one aggregation round. The batch size is 32 in each epoch. Our model structure is

Table (3) Description of our Fall Detection model on UP-FALL dataset.

Layer	Description
Input	An input vector of dimension of 3.
Layer-1	Fully-connected layer with ReLU activation function with 160 parameters. It has the dimension of 20.
Output layer	Fully-connected layer with softmax activation function with 492 parameters. It has the dimension of 12.

summarized in Table (3). The fully-connected layer is trained with ReLU (Rectified Linear Unit) activation function [48]. We employ Softmax [49] as an activation function for our output layer where the size of the output vector equals all the classification classes.

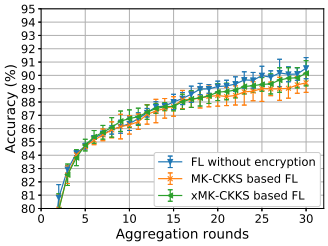
After distributed model training, each device encrypts the model weights by different homomorphic encryption algorithms and then sends them to the server for decryption and aggregation. The implementations of MK-CKKS and xMK-CKKS homomorphic encryption are based on the HEAAN library (https://github.com/snucrypto/HEAAN). Specifically, the security parameter is $\lambda = 128$ bits, the RLWE dimension is $n = 2^{16}$ and the bit length of ciphertext modulus is $\log q = 800$. We set the secret distribution χ as the uniform distribution over the set of polynomials in R whose coefficients are in $\{0, \pm 1\}$, and use the Gaussian distribution of standard 3.2 deviations to sample error polynomials. For Paillier homomorphic encryption, we apply the Paillier library (http://hms.isi.jhu.edu/acsc/libpaillier/), in which the two primes for key generation are set to be 1536 bits. Therefore, the encryption key is 3072 bits that guarantees 128-bits security level. All of these homomorphic encryption implementations are written by C++ language. For arbitrary-precision arithmetic operations, we employ GMP framework (https://gmplib.org/), NTL library that provides algorithms and structures for big integers (https://libntl.org/).

6.2 | Discussion of results

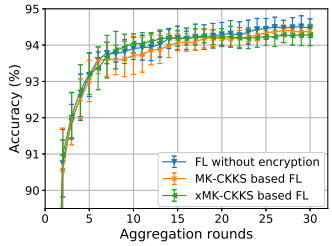
We evaluated fall detection model accuracy to ensure the effectiveness of our scheme and compared it to the federated learning scheme without encryption. In particular, we analyzed the communication cost and tested the aggregation rounds with different numbers of local training epochs (L). In addition, since our scheme is deployed on IoT devices with constrained hardware resources, we evaluated its energy consumption. We compared it to other federated learning schemes to validate its applicability for IoT devices. We further evaluated the computational cost of each scheme during encryption, decryption, computation of cipher sums, and computation of decryption share.

Accuracy. We compared the accuracy of our scheme to traditional federated learning and xMK-CKKS based federated learning(cf. Fig. 4). For each scheme, we executed five trials and reported the accuracy along with the standard deviation. When we employ FedAvg with the local epochs $L = 20$ in one aggregation round, the xMK-CKKS based scheme provides 94.28% of accuracy, which is almost the same as the federated learning scheme, which has 94.47% of accuracy. In the case of FedSGD (with one local epoch), the xMK-CKKS based scheme provides 90.15% of accuracy, which is very close to the federated learning scheme, which has 90.54% of accuracy. The results demonstrate the efficiency of our scheme, which preserves model accuracy while protecting the model updates in federated learning.

Communication cost. In our implementations, the local model weights are represented by 64 bits. As shown in Fig. 5, in one aggregation round, as the number of model weights increases, the communication cost of our scheme is much lower than Paillier based scheme. Take the fall detection model we use with 492 model weights as an example;



(a) Accuracy of the federated learning schemes utilizing $L = 1$ local training epochs($FedSGD$)



(b) Accuracy of the federated learning schemes utilizing $L = 20$ local training epochs

Fig. 4. Classification accuracy comparison with different number of training epochs executed locally(L) by each device in one aggregation round. Note that the y-axis range in figures (a), (b) is adapted for better visibility.

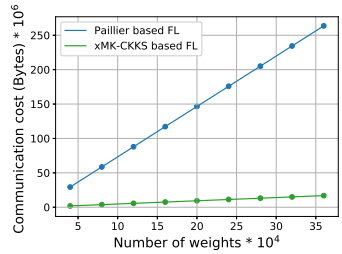


Fig. 5. Communication cost for Paillier based federated learning and xMK-CKKS based federated learning as the number of weights increases.

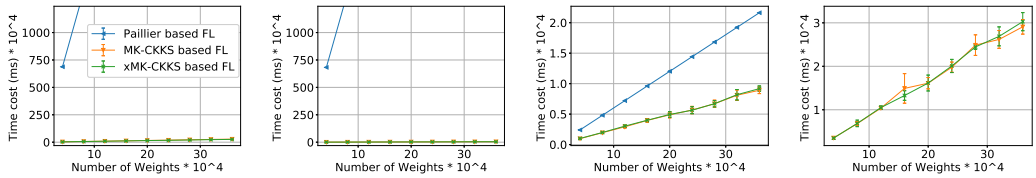
Table (4) Energy consumption of different federated learning (FL) schemes executed on distributed Jetson Nano IoT devices.

Scenario	xMK-CKKS based FL	MK-CKKS based FL	Paillier based FL	FL w/o encryption	No activity (idle)
Energy (W)	2.4	2.4	2.3	2.3	1.8

in xMK-CKKS based federated learning, the ciphertext size of the model weights is 24KB. This is much smaller than the ciphertext size of the Paillier based Federated learning, which is 378KB. Although our scheme needs one more communication round for collaborative decryption, the additional message size is only 43 KB for the cipher sum C_{sum_1} and 43 KB for decryption share D_i in each aggregation round. To further reduce the communication cost in general, we apply FedAvg with 20 local epochs (Fig. 4(b)) in one aggregation round instead of FedSGD (Fig. 4(a), each minibatch update requires one aggregation round). Our scheme achieved a 90.55% model accuracy after three aggregation rounds, which is significantly smaller than the FedSGD based federated learning scheme using 27 aggregation rounds to achieve closer accuracy of 90.11%.

Energy Consumption. We monitored the energy consumption during the operation of the IoT devices for MK-CKKS, xMK-CKKS, Paillier based federated learning schemes, federated learning without encryption, and when the IoT device is idle (Table (4)). The energy consumption is sampled every two seconds, and we obtain the average consumption of these five measurements. The energy consumption of xMK-CKKS based federated learning is around 2.4 watts, which is only 24% of the maximum energy of the Jetson Nano IoT device (10 Watts). Therefore, we conclude that the scheme is applicable for IoT devices.

Computational cost. We discuss the computational cost when varying the number of model weights (up to nearly 1/3 million) in Fig. 6. We report computational cost at the encryption phase (Fig. 6(a)), aggregation phase (Fig. 6(b)), decryption share calculation phase (Fig. 6(c)), and decryption phase (Fig. 6(d)), where each experiment is executed four times. We compare our scheme with MK-CKKS based scheme and Paillier based scheme. We notice that our scheme behaves approximately similar to MK-CKKS based scheme while significantly taking less time than the Paillier based scheme. The results show that our proposed scheme can be deployed in large-scale data scenarios due to its reduced time cost during all scheme phases.



(a) Encryption cost for different number of weights

(b) Decryption cost for different number of weights

(c) Ciphers sum cost for different number of weights

(d) Decryption share cost for different number of weights

Fig. 6. Computational cost for Paillier based federated learning, MK-CKKS based federated learning, and xMK-CKKS based federated learning in different phases.

7 | CONCLUSION AND FUTURE WORK

We have proposed a novel privacy-preserving federated learning scheme based on multi-key homomorphic encryption to protect data privacy. We improved the MK-CKKS homomorphic encryption scheme, overcoming the risk of privacy leakage when used in the federated learning scenario to protect the privacy of individual model updates. Our xMK-CKKS scheme defines the aggregated public key and the decryption share to achieve secure and simple encryption and decryption. Therefore, it is more suitable for privacy protection in federated learning scenarios. Furthermore, the proposed xMK-CKKS based federated learning scheme guarantees the confidentiality of model updates by applying multi-key homomorphic encryption and is resistant to collusion attacks between $k < N - 1$ participating devices and the server. Therefore, the proposed scheme provides strong privacy protection for federated learning.

We evaluated this xMK-CKKS based federated learning scheme in terms of accuracy, communication cost, energy consumption, and computational cost. Furthermore, we compared it with federated learning based on MK-CKKS homomorphic encryption and Paillier homomorphic encryption. In particular, in an experiment with 10 Jetson Nano IoT devices and a single dedicated server, we executed a large-scale data scenario situated in the domain of elderly care. We extensively evaluated and compared the schemes. The experiment demonstrates that our scheme is effective, applicable in IoT domains, and efficient in accuracy, communication cost, energy consumption, and computational cost. It enables the implementation of secure federated learning on IoT devices.

The proposed xMK-CKKS based scheme is robust against honest-but-curious participating devices; however, it may not withstand attacks of malicious participants that sabotage the learning process. For instance, malicious devices may send incorrect or random values or even values specifically designed to interfere and bias the federated learning process instead of sharing correct model updates. A common mechanism to address this problem attempts to identify incorrect parameters to discard them by designing Byzantine robust aggregation rules. For further details, we refer to Median [50] and Krum [51]. Nevertheless, these defense mechanisms assume that the training data of each device is independent and identically distributed (iid), which is difficult to be guaranteed in actual IoT scenarios. More importantly, the server needs to obtain the model updates from each device, which again introduces a risk of privacy leakage to the server or other participants overhearing the communication. However, protecting the individual model updates via cryptographic routines, as the xMK-CKKS based federated learning scheme we have proposed, prevents individual model updates from being analyzed for potential anomalies or suspicious patterns. Therefore, further research is needed to investigate how federated learning can not only resist Byzantine attacks but also protect the privacy of the remotely participating devices.

ACKNOWLEDGEMENTS

This work is partially supported by China National Science Foundation under grant number 62072356, and the National Key Research and Development Program of Shaanxi under grant number 2019ZDLGY12-08. We would like to acknowledge partial funding by the Academy of Finland in the project ABACUS (ICT 2023). The support provided by China Scholarship Council (CSC) during a visit of Jing Ma to Aalto University is acknowledged (file No.201906960151).

References

[1] Kianoush S, Raja M, Savazzi S, Sigg S. A cloud-IoT platform for passive radio sensing: Challenges and application case studies. *IEEE Internet of Things J*. 2018;5(5):3624-3636.

[2] Li J, Jin J, Yuan D, Palaniswami M, Moessner K. EHOPES: Data-centered Fog platform for smart living. *2015 International Telecommunication Networks and Applications Conference (ITNAC)*. 2015:308-313.

[3] Li J, Jin J, Yuan D, Zhang H. Virtual fog: A virtualization enabled fog computing framework for Internet of Things. *IEEE Internet of Things J*. 2017;5(1):121-131.

[4] Konečný J, McMahan HB, Yu FX, Richtárik P, Suresh AT, Bacon D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint*. 2016;arXiv:1610.05492.

[5] Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol*. 2019; 10(2):1-19.

[6] Li J, Lyu L, Liu X, Zhang X, Lv X. FLEAM: A Federated Learning Empowered Architecture to Mitigate DDoS in Industrial IoT. *IEEE Trans. Ind. Inform*. 2021.

[7] Zhu L, Han S. Deep leakage from gradients. *Federated learning*. 2020:17-31.

[8] Melis L, Song C, De Cristofaro E, Shmatikov V. Exploiting Unintended Feature Leakage in Collaborative Learning. In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019:691-706.

[9] Hitaj B, Ateniese G, Perez-Cruz F. Deep models under the GAN: information leakage from collaborative deep learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017:603-618.

[10] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K. Practical secure aggregation for federated learning on user-held data. *arXiv preprint*. 2016;arXiv:1611.04482.

[11] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K. Practical secure aggregation for privacy-preserving machine learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017:1175-1191.

[12] Li Y, Zhou Y, Jolfaei A, Yu D, Xu G, Zheng X. Privacy-Preserving Federated Learning Framework Based on Chained Secure Multiparty Computing. *IEEE Internet Things J*. 2020;8(8):6178-6186.

[13] Xu G, Li H, Liu S, Yang K, Lin X. Verifynet: Secure and verifiable federated learning. *IEEE Trans. Inf. Forensic Secur*. 2019;15:911-926.

[14] Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L. Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016:308-318.

[15] Geyer RC, Klein T, Nabi M. Differentially private federated learning: A client level perspective. *arXiv preprint*. 2017;arXiv:1712.07557.

[16] Hao M, Li H, Luo X, Xu G, Yang H, Liu S. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Trans. Ind. Inform*. 2019;16(10):6532-6542.

[17] Aono Y, Hayashi T, Wang L, Moriai S. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. *IEEE Trans. Inf. Forensic Secur*. 2017;13(5):1333-1345.

[18] Zhang X, Fu A, Wang H, Zhou C, Chen Z. A privacy-preserving and verifiable federated learning scheme. *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. 2020:1-6.

[19] Gilad-Bachrach R, Dowlin N, Laine K, Lauter K, Naehrig M, Wernsing J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. *International conference on machine learning*. 2016:201-210.

[20] Li P, Li J, Huang Z, Li T, Gao CZ, Yiu SM, Chen K. Multi-key privacy-preserving deep learning in cloud computing. *Futur. Gener. Comp. Syst*. 2017;74:76-85.

[21] Froelicher D, Troncoso-Pastoriza JR, Sousa JS, Hubaux JP. Drynx: Decentralized, secure, verifiable system for statistical queries and machine learning on distributed datasets. *IEEE Trans. Inf. Forensic Secur*. 2020;15:3035-3050.

[22] Chen H, Dai W, Kim M, Song Y. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019:395-412.

[23] Goldwasser S, Micali S. Probabilistic encryption & how to play mental poker keeping secret all partial information. *Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali*. 2019:173-201.

[24] Paillier P. Public-key cryptosystems based on composite degree residuosity classes. *International conference on the theory and applications of cryptographic techniques*. 1999:223-238.

[25] Damgrd I, Jurik M, Generalisation A. a Simplification and Some Applications of Paillier's Probabilistic Public-Key System, *PKC*. 2001:119-136.

[26] Kawachi A, Tanaka K, Xagawa K. Multi-bit cryptosystems based on lattice problems. *International Workshop on Public Key Cryptography*. 2007:315-329.

[27] Gentry C. A fully homomorphic encryption scheme. *Stanford university*. 2009.

[28] Rothblum R. Homomorphic encryption: From private-key to public-key. *Theory of cryptography conference*. 2011:219-234.

[29] López-Alt A, Tromer E, Vaikuntanathan V. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. 2012:1219-1234.

[30] Hoffstein J, Pipher J, Silverman JH, NTRU. A ring-based public key cryptosystem. *Algorithmic Number Theory*. 2000:267-288.

[31] Doröz Y, Hu Y, Sunar B. Homomorphic AES evaluation using the modified LTV scheme. *Designs Codes Cryptogr.* 2016;80(2):333-358.

[32] Clear M, McGoldrick C. Multi-identity and multi-key leveled FHE from learning with errors. *Annual Cryptology Conference*. 2015:630-656.

[33] Mukherjee P, Wichs D. Two round multiparty computation via multi-key FHE. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. 2016:735-763.

[34] Brakerski Z. Fully homomorphic encryption without modulus switching from classical GapSVP. *Annual Cryptology Conference*. 2012:868-886.

[35] Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.2012*. 2012:144.

[36] Cheon JH, Kim A, Kim M, Song Y. Homomorphic encryption for arithmetic of approximate numbers. *International Conference on the Theory and Application of Cryptology and Information Security*. 2017:409-437.

[37] McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. *Artificial intelligence and statistics*. 2017:1273-1282.

[38] Cramer R, Damgård IB. Secure multiparty computation. *Cambridge University Press*. 2015.

[39] Dwork C. Differential privacy: A survey of results. *International conference on theory and applications of models of computation*. 2008:1-19.

[40] Dwork C, McSherry F, Nissim K, Smith A. Calibrating noise to sensitivity in private data analysis. *Theory of cryptography conference*. 2006:265-284.

[41] Dwork C, Kenthapadi K, McSherry F, Mironov I, Naor M. Our data, ourselves: Privacy via distributed noise generation. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. 2006:486-503.

[42] Rivest R L, Adleman L, Dertouzos ML. On data banks and privacy homomorphisms. *Foundations of secure computation*. 1978;4(11):169-180.

[43] ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*. 1985;31(4):469-472.

[44] Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings. *Annual international conference on the theory and applications of cryptographic techniques*. 2010:1-23.

[45] Martínez-Villaseñor L, Ponce H, Brieva J, Moya-Albor E, Núñez-Martínez J, Peñafort-Asturiano C. UP-fall detection dataset: A multimodal approach. *Sensors*. 2019;19(9):1988.

[46] Zhao Y, Li M, Lai L, Suda N, Civin D, Chandra V. Federated learning with non-iid data. *arXiv preprint*. 2018;arXiv:1806.00582.

[47] Kingma DP, Ba J. Adam: A method for stochastic optimization. *arXiv preprint*. 2014;arXiv:1412.6980.

[48] Maas A L, Hannun A Y, Ng A Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. *Proc. icml*. 2013;30(1):3.

[49] Hinton G E, Salakhutdinov R R. Replicated Softmax: an Undirected Topic Model. *Adv. Neural Inf. Process. Syst*. 2009;22:1607-1614.

[50] Yin D, Chen Y, Kannan R, Bartlett P. Byzantine-robust distributed learning: Towards optimal statistical rates. *International Conference on Machine Learning*. 2018:5650-5659.

[51] Blanchard P, El Mhamdi E M, Guerraoui R, Stainer J. Machine learning with adversaries: Byzantine tolerant gradient descent. *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017:118-128.