Homomorphic Encryption-based Privacy-preserving Federated Learning in IoT-enabled Healthcare System

Li Zhang, Jianbo Xu, Pandi Vijayakumar, *Senior Member, IEEE*, Pradip Kumar Sharma, *Senior Member, IEEE*, and Uttam Ghosh, *Senior Member, IEEE*

Abstract—In this work, the federated learning mechanism is introduced into the deep learning of medical models in Internet of Things (IoT)-based healthcare system. Cryptographic primitives, including masks and homomorphic encryption, are applied for further protecting local models, so as to prevent the adversary from inferring private medical data by various attacks such as model reconstruction attack or model inversion attack, etc. The qualities of the datasets owned by different participants are considered as the main factor for measuring the contribution rate of the local model to the global model in each training epoch, instead of the size of datasets commonly used in deep learning. A dropout-tolerable scheme is proposed in which the process of federated learning would not be terminated if the number of online clients is not less than a preset threshold. Through the analysis of the security, it shows that the proposed scheme satisfies data privacy. Computation cost and communication cost are also analyzed theoretically. Finally, skin lesion classification using training images provided by the HAM10000 medical dataset is set as an example of healthcare applications. Experimental results show that compared with existing schemes, the proposed scheme obtained promising results while ensuring privacy preserving.

Index Terms—Federated Learning, Homomorphic Encryption, Privacy-preserving, Convolutional Neural Networks, IoT-enabled Healthcare System.

1 Introduction

The Internet of Things (IoT) is a new paradigm with a wide range of applications. IoT-connected healthcare applications provide real-time monitoring and smart medical IoT devices that are synchronized to a smartphone app, allowing doctors to obtain medical data from their patients at any time or location. It also provides computer-assisted diagnostics, medical image analysis, and remote medical support, etc. It enables medical centers to operate more efficiently, and enables patients to receive better care. There are certain advantages to utilizing IoT-based healthcare methods, which might enhance the treatment quality and efficiency, and therefore the health of patients [1].

The development of big data and artificial intelligence has made it convenient to carry out joint researches in various fields. Especially in the medical field, the sharing of electronic health data benefits doctors to predict some diseases and formulate related treatment plans [2]. To some extent, data sharing promotes the development of the entire medical industry. However, the sensitivity of medical data has become an obstacle to resource sharing. Most of the medical institutions are located in disperse geographical locations and abide by different administrative management. Therefore, they are reluctant to share patient medical data by taking the risk of violating privacy ethics or even losing the economic benefits, such as leaking information about AIDS patients, publishing ingredients of patented therapeutic drugs, etc. In other fields, privacy issues are also urgent problems needed to be resolved.

Thanks to paradigm-shifting advancements in machine learning, computer-aided diagnostic techniques have already reached unprecedented levels. In this context, as a common type of cancer that originates in the epidermal layer when abnormal cells are exposed to ultraviolet radiation, skin cancer has gotten significant attention, and deep learning algorithms have attained a level of precision that is equivalent to that of trained dermatologists [1] [3].

For IoT-enabled healthcare system, in pursuing the huge benefits data sharing brings but unwilling to violate privacy, scholars have proposed a federated learning framework, which allows data to contribute to federated machine learning despite being kept in local repositories [4] [5]. Federated learning provides broad prospects for the application of artificial intelligence in different industries. However, in terms of privacy preservation, federated learning is not foolproof. There are still some challenges in applying federated learning to real application scenarios, such as IoT-

- *Corresponding author: Jianbo Xu, Pandi Vijayakumar*
- *L. Zhang and J. Xu are with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China, and Hunan Laboratory for Service Computing and Novel Software Technology, Hunan University of Science and Technology, Xiangtan 411201, China.*
  E-mail: lzhang@mail.hnust.edu.cn, jbxu@hnust.edu.cn
- *P. Vijayakumar is with the Department of Computer Science and Engineering, University College of Engineering Tindivanam, Tindivanam, Tamilnadu 604001, India.*
  E-mail: vijibond2000@gmail.com
- *P. K. Sharma is with the Department of Computing Science, University of Aberdeen, UK.*
  E-mail: pradip.sharma@abdn.ac.uk
- *U. Ghosh is with the Electrical Engineering and Computer Science, Vanderbilt University, USA.*
  E-mail: ghosh.uttam@ieee.org

Manuscript received Oct. 10, 2021; revised April 21, 2022.

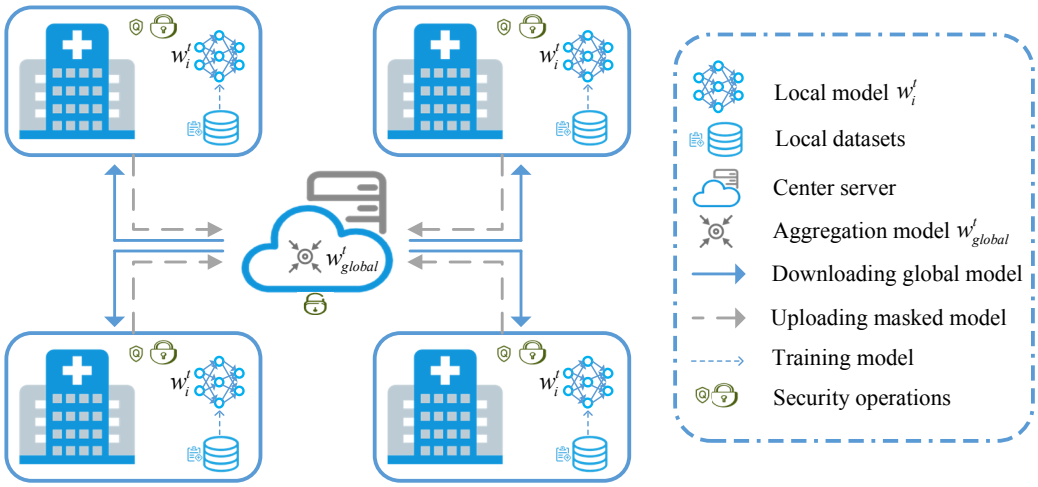


Fig. 1. The system overview for federated learning on medical data.

based healthcare systems [6] [7] [8]. Although local data is not directly shared, models trained on these data may also be snooped by malicious adversaries or honest but curious parties when local models are aggregated into a center. Moreover, under the circumstance of knowing the local model, snoopers may adopt some attacks to restore the original data, which indirectly leads to information leakage. Therefore, the federated learning mechanism with privacy preservation has attracted more and more attentions. In addition, there are several technical issues in federated learning application, such as how to deal with client dropouts, and how to ensure the accurate global model and low computation and communication overhead, etc.

Although some privacy preservation schemes for federated learning have been proposed in recent years, such as [9] [10] [11]. Most of them have some limitations more or less. For example, both [10] and [11] proposed homomorphic encryption for federated learning. However, both of them could not solve the problem of dropout clients. In [9], Bonawitz et al. proposed a dropout-tolerable scheme, and the masked parameters are transferred, but there is no effective method for improving the accuracy of the model. Therefore, from the perspective of the sensitivity of medical data, the paper proposes a privacy preservation scheme for the federated learning of multiple medical institutions in IoT healthcare applications, as can be seen in Fig. 1. The scheme introduces cryptographic primitives such as homomorphic encryption and secure multi-party computing into the framework of federated learning, so that the privacy of data can still be ensured even if there are collusions among honest but curious participants. Furthermore, in order to alleviate the cost problems that ubiquitously existed in traditional cryptography-based schemes, the paper proposed an improved scheme according to an existing privacy preservation algorithm. Simultaneously, the heterogeneous characteristics of data in different medical institutions are considered, and the model accuracy and training efficiency are promoted.

Specifically, the new algorithm compensates the lack of consideration about data quality in traditional model aggregation algorithms, which makes high-quality local models have a higher contribution rate in the global model aggregation, and reduces the negative impact caused by the heterogeneity of data on the convergence rate and accuracy of the global model. Different from Bonawitz et al.'s scheme [9], homomorphic encryption is utilized to calculate the sum of client's data quality. An appropriate adjustment to the ElGamal encryption algorithm is made so as to change the characteristics of the algorithm from multiplicative homomorphism to additive homomorphism. Therefore, communication overhead is reduced. Moreover, unlike traditional encryption in federated learning, the homomorphic encryption scheme does not encrypt every model parameter. Because the model in deep learning usually has high dimensions, and the homomorphic encryption on such high-dimensional data would bring huge computation overhead. There is only a variable called data quality to be encrypted for each client in each training epoch in our scheme. Therefore, the scheme proposed in the paper would not cause the computation overhead to increase sharply.

The main contributions of this paper are summarized as follows:

(1) A novel masking scheme based on homomorphic encryption and the secure multi-party computation have been proposed for federated learning, which utilizes a weighted average algorithm based on data quality to replace the traditional weight calculation method based on the amount of data.

(2) A dropout-tolerable and participants collusion-resistible solution has been proposed in our scheme by employing Diffie-Hellman key exchange and Shamir secret sharing algorithm.

(3) A federated learning prototype system for medical data has been implemented, and extensive experiments using real skin cancer datasets have been conducted to validate the privacy preserving and efficiency of the proposed federated learning scheme.

The organization of this paper is as follows: the related work about federated learning is surveyed in Section 2. The basic principles of deep learning, federated learning, and cryptographic primitives are introduced in Section 3. The privacy preservation scheme proposed in the paper is

described in detail in Section 4. Security analysis on the scheme is conducted in Section 5. Theoretical computation cost and communication cost are analyzed in Section 6. Substantial experiments are performed for illustrating efficiency and accuracy in Section 7. Conclusion and future work are summarized in the final section.

2 Related work

In recent years, although big data, machine learning, artificial intelligence and other technologies have been developed rapidly, the privacy leakage brought by the traditional direct data sharing mechanism has also increased seriously [12] [13] [14] [15]. Privacy problem has led to the emergence of isolated data islands, which severely hindered the development of artificial intelligence. In order to solve the privacy problem, Google initially proposed the concept of federated learning in 2016 for the update of the mobile terminal user's local model. They used a federated average (FedAvg) technology to promote high-efficient machine learning among multiple nodes and guarantee privacy security for all data in personal smartphones [4] [5]. Since then, some open source projects for federated learning have flourished. The most representative projects are respectively the FATE project[1] developed by WeBank and the TensorFlow framework developed by Google[2], which provide different support strategies for privacy preservation in machine learning.

Although federated learning has been continuously developed, various attack models have also emerged. Bhowmick et al. [16] elaborated on reconstruction attack and countermeasures. Nasr et al. [17] and Shokri et al. [18] designed member inference attacks with the white-box model and black-box model, respectively. Fredrikson et al. [19] found that given a model and some statistical information, the dose guidance model in pharmacogenetics is vulnerable to model inversion attack. Melis et al. [20] explained how to implement attribute inference attack. Especially, Zhu et al. [21] proposed a depth gradient leakage scheme, in which the adversaries unknew any other information except the local model, they still can restore image with the high similarity to the original sample by training the dummy sample. Therefore, it can be seen that even if the original data is kept locally in federated learning, the update of the local model still gives the adversaries some opportunities to attack.

With the appearance of various attack models that violate the privacy and confidentiality of machine learning, miscellaneous countermeasures have been proposed. Zhang et al. [22] and Hardy et al. [23] applied additive homomorphic encryption algorithm to prevent the local model from being snooped by honest but curious participants in model aggregation process. Although homomorphic encryption provides the strongest guarantee for privacy preservation, it introduces a large amount of computational overhead. In federated deep learning with a high-dimensional model, homomorphic encryption has an obvious impact on performance. It has become an important reason why homomorphic encryption is difficult to be put into practice. Aimed at saving computational overhead, some homomorphic encryption schemes with gradient sparseness and gradient quantization have been proposed [24] [25].

In addition to encryption technology, secure multi-party computation is also an important realization method for privacy preservation. Xu et al. [26] utilized oblivious transfer and garbled circuits in secure multi-party computation to implement homomorphic multiplication and division operations between two servers, and then completed secure model aggregation. In 2017, Bonawitz et al. [9] designed a secure double-masking aggregation scheme by using many technologies such as the Diffie-Herman key agreement, t-out-of-n Shamir secret sharing, pseudo-random generator, the public key infrastructure, authentication, and signatures, etc. Although the scheme permitted the aggregation to be successful when some participants dropped out, frequent requests were needed for unmasking in each training epoch, which caused a huge communication overhead. In order to decrease the cost, Choi et al. [27] proposed an improved scheme with a new secret sharing topology, in which the secret shares were only shared with their neighboring nodes instead of being shared with all nodes. Thereby communication overhead was reduced.

Another privacy preservation mechanism is differential privacy, which has received wide attention from scholars due to its significant reduction in computing costs compared with the previous two technologies. Differential privacy was first proposed by Dwork in 2006 [28]. Later, it was introduced into federated machine learning to ensure that sensitive information is not leaked. Nowadays, many scholars have carried out massive researches on the core issue involved in differential privacy technology, namely, the trade-off between noise intensity and model accuracy [29] [30].

Federated learning also provides an effective way to train medical models without sharing the patient's electronic health records. Undoubtedly, the entire medical community would benefit from disease prediction and treatment. At present, many literatures are focusing on the federated learning of medical data. Roy et al. [31] proposed a decentralized federated learning framework for brain segmentation in MRI T1 scans. Brisimi et al. [32] also solved a binary supervised classification problem to predict the hospitalization time for patients with heart disease. Although the data was kept locally, the privacy leakage perhaps caused by the model updates was still non-negligible. However, both of the two schemes did not consider the solutions. Lee et al. [33] proposed a multi-institutional federated training algorithm for the model of patient similarity that uses homomorphic encryption technology to ensure the privacy of patients. Choudhury et al. [34] applied differential privacy technology in the federated learning of medical data. Rieke et al. [35] discussed the vital role of federated learning in the development of digital healthcare and emphasized the main challenges currently faced. Finally, they looked forward to the prospects of digital healthcare. In the paper, we will dedicate to privacy preservation in federated learning for medical data.

The existing privacy preservation schemes for federated learning have some shortcomings more or less, and cannot perfectly solve all the problems in one scheme. For example,

1. https://github.com/FederatedAI/FATE
2. https://www.tensorflow.org/federated

Asad et al. proposed a federated learning scheme named FedOpt, which designed a sparse compression algorithm for efficient communication and also integrated the homomorphic encryption [10]. In [11], Fang et al. proposed a federated learning scheme based on homomorphic encryption for all parameters. However, both of them could not solve the problem of dropout clients. In this paper, motivated by Bonawitz et al.'s work [9], we try to learn the advantages of the existing schemes, and rely on the combination of various cryptographic primitives to support the privacy preservation for federated learning and also solve the problem of client dropout, ensuring the accuracy of the model at the same time, which is different from existing schemes.

3 PRELIMINARY

3.1 Federated deep learning

Deep machine learning commonly refers to machine learning based on neural networks such as Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) [36] [37], etc. The network model is generally divided into three layers, including input layer, hidden layer and output layer, and there are numerous neurons in each layer. Deep learning aims at training connection weight between two neurons in neighbor layers, namely network model, to the output of the model, which has a minimal error with the original label.

Assuming that $\mathcal{DS}$ is a dataset with m samples, denoted as $\mathcal{DS} = \{(\mathbf{x}_k, \mathbf{y}_k), k = 1, 2, ..., m\}$, where $\mathbf{x}_k$ is the feature vector of the k-th sample and $\mathbf{y}_k$ is the label. Given an initial weight vector $\mathbf{w}$, we firstly compute the output function $f(\mathbf{x}_k, \mathbf{w})$ of the neural network, and then calculate the error between the output function and the label. The error is defined as the loss function $\mathcal{L}$ which is described as formula (1).

$$\mathcal{L}_f(\mathcal{DS}, \mathbf{w}) = \frac{1}{|\mathcal{DS}|} \sum\nolimits_{<\mathbf{x}_k, \mathbf{y}_k> \in \mathcal{DS}} \|f(\mathbf{x}_k, \mathbf{w}) - \mathbf{y}_k\|_2 \quad (1)$$

where, $|\mathcal{DS}|$ and $\|\bullet\|_2$ represents the size of the dataset and the L_2-norm of a vector, respectively. In a neural network, the calculation process from output function to loss function is called as forward propagation. In order to minimize the loss function, the model parameters need to be adjusted continuously. This process is called as backward propagation, in which the stochastic gradient descent (SGD) algorithm is usually adopted for finding out the optimal model parameters. The specific model update in each iteration is shown in formula (2).

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \beta \cdot \nabla \mathcal{L}_f(\mathcal{DS}, \mathbf{w}^t) \quad (2)$$

Here, t is the number of iterations, and β is the learning rate which denotes the step length of the model adjustment in each iteration. $\nabla \mathcal{L}_f(\mathcal{DS}, \mathbf{w}^t)$ is model gradient which means the partial derivative of the loss function in each model dimension. Given a d-dimension model $\mathbf{w}$=$<w_1$,w_2,...,$w_d>^T$, the gradient can be calculated according to formula (3).

$$\nabla \mathcal{L}_f(\mathcal{DS}, \mathbf{w}^t) = < \frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, ..., \frac{\partial \mathcal{L}}{\partial w_d} >^T \quad (3)$$

For massive medical images, CNN is the preferred choice for model training. First, the image is sampled and compressed in the convolutional layer and pooling layer, so as to achieve the purpose of reducing the number of model parameters, saving computational complexity and preventing overfitting. After that, the flatten layer is connected to the full connected layer in the traditional neural network for subsequent training.

In federated learning, assuming that there are N clients to collaborate for learning, we denoted them as $\{\mathcal{P}_i\}$, $i \in \{1, 2, ..., N\}$. Each client trains the local model w_i on its own dataset $\mathcal{DS}_i$ through performing several iterations with SGD. Then, the local model is uploaded to the server. The server aggregates all of the local models submitted by N clients into a global model using an arithmetic average algorithm or weighted average algorithm [38] [39]. Given the aggregation weight α_i for the client $\mathcal{P}_i$, the global model $\mathbf{w}_{global}$ can be generated according to formula (4), and then distributed to each client.

$$\mathbf{w}_{global} = \sum_{i=1}^{N} \alpha_i \cdot w_i \quad (4)$$

After that, every client continues to perform iterative training until the model converges or termination condition is reached. In the entire process, the original dataset located in each client is not shared with the server, instead of submitting the model parameters, which preserves the data's privacy to a certain extent.

3.2 Cryptographic primitives

3.2.1 Shamir secret sharing

As the most classic e-out-of-n algorithm, Shamir(e, n) secret sharing [40] has been widely applied in cryptography. The main idea of the algorithm is to divide a secret s into n shares, which are distributed to n different parties. Only if no less than e parties contribute their shares, the secret s can be reconstructed. Otherwise, no parties can maliciously leak any information about s even collusion.

Shamir's secret sharing is composed of two algorithms, namely, secret shares generation and secret reconstruction. In cryptography, the secret s is usually a number sampled on a finite field Z_q^* with a large prime order q. Assuming that $\mathcal{P}$ is the set of parties allocated the secret shares and $\mathcal{V}$ is the set participating in the secret reconstruction, satisfying $e \leq |\mathcal{V}| \leq |\mathcal{P}|$. The secret shares generation algorithm is defined as $\mathrm{SSS.share}(s, e, \mathcal{P}) \rightarrow \{(i, s_i)\}_{i \in \mathcal{P}}$, which takes the secret s and the threshold e as input parameters, and then generates $|\mathcal{P}|$ numbers of secret shares $\{(i, s_i)\}_{i \in \mathcal{P}}$ for the corresponding parties in set $\mathcal{P}$. The secret reconstruction algorithm is shown as $\mathrm{SSS.recons}(e, \{(i, s_i)\}_{i \in \mathcal{V}}) \rightarrow s$, in which the constructor needs to acquire e shares (i, s_i) from set $\mathcal{V}$ for restoring the secret s. If $|\mathcal{V}| \leq e$, for two randomly secrets $\forall s_1, s_2 \in Z_q^*$, the security of the algorithm can be guaranteed only when the following requirements are satisfied:

$$\{\{(i, s_i)\}_{i \in \wp} \leftarrow \mathrm{SSS.share}(s_1, e, \mathcal{P}) : \{(i, s_i)\}_{i \in \mathcal{V}}\}$$
$$\equiv \{\{(i, s_i)\}_{i \in \mathcal{P}} \leftarrow \mathrm{SSS.share}(s_2, e, \mathcal{P}) : \{(i, s_i)\}_{i \in \mathcal{V}}\}$$

Here, the symbol "$\equiv$" represents computational indistinguishability. In order to satisfy the security requirements,

the executor of the algorithm $\mathrm{SSS.share}(s, e, \mathcal{P})$ needs to construct a polynomial with degree (e-1) shown as formula (5).

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{e-1} x^{e-1} \quad (5)$$

Let's set $a_0 = s$, other coefficients a_1, a_2, a_3, $\cdots$, a_{e-1} are given random values, and then $|\mathcal{P}|$ different random numbers $\{(x_i)\}_{i\in\mathcal{P}}$ are sampled in Z_q^*. After that, the executor calculates $f(x_i)$ and distributes the secret shares $\{(x_i, f(x_i))\}_{i\in\mathcal{P}}$ to the corresponding parties in $\mathcal{P}$. When $|\mathcal{V}|$ secret shares are collected, the constructor selects e shares arbitrarily, and then reconstructs the secret s according to the Lagrange interpolation polynomial shown as formula (6).

$$f(x) = \sum_{i=1}^{e} \prod_{1\le j\le e, j\neq i} \frac{x - x_j}{x_i - x_j} f(x_i) \quad (6)$$

By formula (5), we know $f(0) = a_0 = s$. So, let $x = 0$ in formula (6), then the algorithm $\mathrm{SSS.recons}(e, \{(i, s_i)\}_{i\in\mathcal{V}})$ can be implemented for restoring s, which is shown as formula (7).

$$s = \sum_{i=1}^{e} \prod_{1\le j\le e, j\neq i} \frac{-x_j}{x_i - x_j} f(x_i) \quad (7)$$

3.2.2 Diffie-Hellman key agreement

The Diffie-Hellman key agreement protocol [41] allows two parties to obtain the symmetric key without exchanging any secret information. The protocol mainly includes three phases: initialization, generation of the public key and the private key, as well as key agreement. First, the key generation center (KGC) selects a security parameter k as input of the initialization function, and outputs a cyclic group G with a generator g and a large prime order q. Then it sets (q, g, G) as public parameters. Subsequently, KGC randomly selects a number $s \in Z_q^*$ as private key s_i^{sk} and calculates $g_s \in G$ as public key s_i^{pk} for each client i, then sends (s_i^{sk}, s_i^{pk}) to the corresponding client through a secure channel. After receiving the pairwise keys, client i broadcasts s_i^{pk} to others. In the phase of key agreement, assuming that there are two clients (denoted as a and b) to execute key agreement, client a calculates $s_{a,b} = (s_b^{pk})^{s_a^{sk}} = (g^{s_b})^{s_a}$ with client b's public key s_b^{pk} and its own private key s_a^{sk}. Similarly, client b computes $s_{b,a} = (s_a^{pk})^{s_b^{sk}} = (g^{s_a})^{s_b}$ using client a's public key s_a^{pk} and its own private key s_b^{sk}. Finally, both of two clients acquire the same key $s_{a,b} = s_{b,a}$.

Definition (Decisional Diffie-Hellman assumption). Assuming that $\mathcal{O}(k) \to (q, g, G)$ is an algorithm about security parameter generation, a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ executes the following random oracle experiment $\mathrm{DDH} - \mathrm{Exp}_{\mathcal{O},\mathcal{A}}^{c}(k)$ which is parameterized by a group of public secure elements (q, g, G) and a bit c.

$\mathrm{DDH} - \mathrm{Exp}_{\mathcal{O},\mathcal{A}}^{c}(k)$:
(1) $(q, g, G) \leftarrow \mathcal{O}(k)$;
(2) $a \leftarrow Z_q^*, A \leftarrow g^a$;
(3) $b \leftarrow Z_q^*, B \leftarrow g^b$;
(4) if $c = 1$, $s \leftarrow g^{ab}$, else $s \leftarrow (\forall s' \in G \text{ and } s' \neq g^{ab})$;
(5) $(q, g, G, A, B, s) \to c'$;
(6) if $c' = c$, $output = 1$, otherwise $output = 0$.

In the experiment, the advantage that the adversary can break the semantic security of the Diffie-Hellman protocol is defined as formula (8).

$$Adv_{\mathcal{O},\mathcal{A}}^{\mathrm{DDH}}(k) = \left| Prob(DDH - Exp_{\mathcal{O},\mathcal{A}}^{c}(k) = 1) - \frac{1}{2} \right| \quad (8)$$

Here, $Prob(\bullet)$ represents a probability function. Suppose there exists a negligible function $\varepsilon(k)$ which satisfies that $Adv_{\mathcal{O},\mathcal{A}}^{\mathrm{DDH}}(k) \le \varepsilon(k)$. In that case, it concludes that no PPT adversary $\mathcal{A}$ can solve the Decisional Diffie-Hellman hard problem, so the key agreement protocol is secure in semantics.

3.2.3 Homomorphic encryption

Homomorphic encryption is a special encryption algorithm [42]. The result obtained by operating on multiple ciphertexts is equivalent to the effect generated by directly operating on corresponding plaintexts.

Assuming that $\mathcal{M}$ and $\mathcal{C}$ respectively means the plaintext space and the ciphertext space, the homomorphic encryption algorithm can be defined in a cryptographic way as expression (9).

$$\begin{aligned} \forall m_a, m_b \in \mathcal{M}, Enc_{pk}(m_a) \odot_{\mathcal{C}} Enc_{pk}(m_b) = \\ Enc_{pk}(m_a \odot_{\mathcal{M}} m_b) \end{aligned} \quad (9)$$

Here, $\odot_{\mathcal{C}}$ and $\odot_{\mathcal{M}}$ represent the operators on the ciphertext space and the plaintext space, respectively. According to the symbols $\odot_{\mathcal{M}}$, homomorphic encryption is divided into additive homomorphism and multiplicative homomorphism. In short, if $\odot_{\mathcal{M}}$ is operator "+", the algorithm is called additive homomorphism; if $\odot_{\mathcal{M}}$ is operator "*", it is called multiplicative homomorphism.

ElGamal algorithm is classified as an asymmetric cryptosystem based on Diffie-Hellman key agreement, which was proposed by Tather ElGamal in 1985 [43]. It has properties of multiplicative homomorphism. The algorithm is mainly composed of four steps.

- Initialization: KGC generates the public parameters (q, g, G) with a secure parameter k, where G is a cyclic group with a large prime q and generator g.
- Key generation: KGC randomly selects a number $\mu \in Z_q^*$ as private key, calculates $y = g^\mu \in G$ as public key, where the public key y is used for encryption, and the private key μ is for decryption.
- Encryption: For encrypting the plaintext m, the message sender selects a random number $r \in Z_q^*$ and calculates $c_1 = g^r$, $c_2 = my^r$, then sends the ciphertext (c_1, c_2) to the receiver.
- Decryption: After receiving the ciphertext (c_1, c_2), the receiver decrypts the plaintext m by computing $m = c_2/c_1^\mu = my^r/(g^r)^\mu$ with private key μ.

According to the basic principle of the ElGamal algorithm, it is only appropriate for multiplicative homomorphism. However, it usually acquires to implement additive aggregation of local models with ciphertext form in the federated learning system. Therefore, it is necessary to take a slight modification on the ElGamal algorithm to achieve additive homomorphism. In fact, it just needs to transform the plaintext m into an exponential form with integer 2

as the base, i.e., $c_2 = 2^m y^r$. From formula (10), it can be seen that the modified ElGamal algorithm satisfies additive homomorphism. After aggregation with ciphertext form, the sum of plaintexts can be restored by decryption formula (11).

$$\begin{aligned} &\forall m_a, m_b \in \mathcal{M}, Enc(m_a) \times Enc(m_b) \\ &= (c_{1a} \cdot c_{1a}, c_{2a} \cdot c_{2a}) \\ &= (g^{r_a+r_b}, 2^{m_a+m_b} y^{r_a+r_b}) \\ &= Enc(m_a+m_b) \end{aligned} \tag{10}$$

$$\begin{aligned} m_a + m_b &= \log_2[(c_{2a} \cdot c_{2a})/(c_{1a} \cdot c_{1a})^{\mu}] \\ &= \log_2[2^{m_a+m_b} y^{r_a+r_b}/(g^{r_a+r_b})^{\mu}] \end{aligned} \tag{11}$$

It is noted that the premise of homomorphic encryption is that different plaintexts must be encrypted with the same public key. In federated learning system, if all clients encrypt their local parameters with the server's key, the server can decrypt all models with the corresponding private key, which means homomorphic encryption is meaningless. Thus, for privacy preservation in federated learning, each client needs to combine other secure measures with the ElGamal algorithm to ensure security, instead of directly encrypting the local parameters using the server's public key.

4 Proposed Scheme

4.1 System model and security requirements

The federated learning system in the medical environment is described in this section. It is mainly composed of two types of participants, one is a model aggregation server, and the other is distributed clients. The specific architecture is shown in Fig. 1. The server is responsible for collecting the masked local models submitted by clients, and then performing a series of operations to complete the secure aggregation of the models. The clients are some medical institutions with a large amount of raw medical data. They take charge of training local models on local medical datasets, and then submitting the masked local models and related secure parameters to the server. Ultimately, the optimal global model is generated through mutual cooperation between the server and the clients in each epoch. Different from the traditional federated learning framework, in order to enhance the privacy preservation of the local model, we introduce a third-party trust authority (TA) which is responsible for initializing some security parameters such as public keys and private keys, etc.

During the entire process of joint training, the server and all clients are considered to be honest but curious, which means that they can comply with the protocol execution but intend to snoop the privacy of others from any intercepted information. Therefore, there needs to fulfill the following security requirements so as to preserve privacy:

(1) In the process of local training, the clients cannot learn any authenticate models of other clients, except their local models and the aggregated global model, so that any sensitive raw data of other clients are not inferred.

(2) The server can only obtain masked models and related secure parameters submitted by the clients, but not the original local model and medical data. However, it can still generate an aggregated global model.

(3) The quality of data located in each client should be confidential. Data quality is an important metric in our system for measuring the contribution rate of different local models in the global model. It is also a significant factor for achieving secure aggregation. Due to the discrimination which server probably brings to some clients with poor data qualities, data quality should be kept confidential. In this way, it can ensure that all clients participate in federated learning fairly and impartially.

In our system, there perhaps exists some malicious and external adversaries who are trying to infer medical privacy by eavesdropping on the messages transmitted in the channel. The internal server in the system may also collude with some clients to satisfy their curiosity about private data. We dedicate to resist against passively malicious adversaries and collusion attacks. However, the deliberately disruptive behaviors on model training such as tampering attacks, impersonation attacks and poisoning attacks are beyond our consideration.

4.2 Data quality and contribution rate

In previous work, the weighted average scheme based on dataset size is commonly adopted for model aggregation. The clients with more low-quality data participate in global model training with the same contribution rate, the accuracy of the model is bound to be impaired. How to measure the contribution rate of local data quality to the global model is the prime problem to be solved. In Miao et al. [44] and Xu et al. [45], a truth discovery algorithm is used to calculate the distance between the observed value of the data and the true value for estimating the reliability of the data source. The algorithm has shown its good performance in the quality evaluation of heterogeneous data for multiple application scenarios such as crowdsourcing and medical treatment. In SGD algorithm, the gradient is an important indicator for optimizing model convergence. Hsieh et al. [46] utilized local gradient amplitude to measure the proportion of the local model in the global model. However, the consistency of the local gradient with the global convergence trend was not considered.

Indeed, the gradient amplitude only represents the convergence speed, and it is easily affected by the size of the dataset and the learning rate. Only the sign of gradient can reflect the optimization direction and the convergence trend. Therefore, we calculate the data quality by referring to the truth discovery algorithm. Nevertheless, we compute the distance between the local gradient and the global gradient in each training epoch instead of the distance between the observed value and the true value described in the truth discovery algorithm.

In our system, the model parameters are exchanged between the clients and the server, instead of the gradients. However, according to SGD algorithm, we can still obtain the global gradient through two global models trained in two adjacent epochs. Supposing that $\mathbf{g}_{global}^{t}$ is the global gradient in the t-th epoch, $\mathbf{w}_{global}^{t}$ and $\mathbf{w}_{global}^{t+1}$ represent the global models in the t-th epoch and the $(t+1)$-th epoch, respectively. Then, $\mathbf{g}_{global}^{t}$ can be deduced through

the formula (12).

$$\mathbf{g}_{global}^{t}=\frac{\mathbf{w}_{global}^{t}-\mathbf{w}_{global}^{t+1}}{\beta} \tag{12}$$

In order to obtain the data quality in the t-th epoch, the client i needs to calculate the distance $\left\|\mathbf{g}_i^t-\mathbf{g}_{global}^t\right\|_2$ between the local gradient $\mathbf{g}_i^t$ and the global gradient $\mathbf{g}_{global}^t$. In our system, the data quality is necessary before the client uploads the local model to the server, while the global gradient in current epoch can only be calculated after the local models are aggregated by the server. For demonstrating this view, we define the normalized distance dis_{global}^t of two neighboring gradients as shown in formula (13), and then perform some model training on the datasets.

$$dis_{global}^{t}=\frac{\left\|\mathbf{g}_{global}^{t}-\mathbf{g}_{global}^{t-1}\right\|_2}{\left\|\mathbf{g}_{global}^{t-1}\right\|_2} \tag{13}$$

Most of the normalized distance dis_{global}^t are near the zero point, which illustrates the approximation between two neighboring gradients. Therefore, the global gradient $\mathbf{g}_{global}^t$ in the t-th epoch can be substituted by the global gradient $\mathbf{g}_{global}^{t-1}$ in the $(t-1)$-th epoch. The data quality Q_i^t for the client i in the t-th epoch is defined as formula (14).

$$Q_i^t=\frac{\delta}{\left\|\mathbf{g}_i^t-\mathbf{g}_{global}^{t-1}\right\|_2}=\frac{\delta}{\sum_{d=1}^{|\mathbf{g}|}(g_{i-d}^t-g_{global-d}^{t-1})^2} \tag{14}$$

Here, $\delta=\mathcal{X}^2_{(1-\tau/2,|\mathbf{g}|)}$ represents the scale factor, which is a public parameter [26]. The symbol $\mathcal{X}$ and τ means the chi-square distribution and the significant level, respectively. $|\mathbf{g}|$ is the dimension of the gradient vector. g_{i-d}^t and $g_{global-d}^{t-1}$ respectively represents the d-th gradient in local gradient vector of the t-th epoch for the client i and the d-th gradient in global gradient vector in the $(t-1)$-th epoch. Therefore, if the signs of g_{i-d}^t and $g_{global-d}^{t-1}$ are opposite, we set the distance to a larger value so as to make the data quality lower. While the smaller distance shows that the local model can follow the optimization direction of the global model better, and the data quality is more suitable for the next iteration. After calculating Q_i^t, the global model for the t-th epoch can be aggregated with a weighted average algorithm based on Q_i^t, which is shown as formula (15).

$$\mathbf{w}_{global}^{t}=\frac{\sum_{i=1}^{N}Q_i^t w_i^t}{\sum_{i=1}^{N}Q_i^t} \tag{15}$$

where N denotes the number of clients participating in federated training, w_i^t represents the local model of the client i in the t-th epoch. It can be seen from formula (15) that better the data quality is, more proportion the local model can take in the global model. In short, better quality means the client can bring greater contribution rate.

4.3 Secure aggregation process

Our proposed privacy-preserving scheme for federated learning is described in detail in this section. In the scheme, homomorphic encryption technology is combined with double-masking mode proposed by Bonawitz et al. [9]. Furthermore, the contribution rate of data quality is also considered, which is more practical for federated learning scenarios with multiple medical institutions. The specific construction is depicted in the following four subsections.

4.3.1 System initialization

Before performing federated training, the server confirms the set of clients participating in training (marked as $\mathcal{P}_1$) and the network model such as DNN, CNN, etc. Simultaneously, it defines the learning rate β, the maximum epochs number T for training convergence, the initialized global model $\mathbf{g}_{global}^0$ and the threshold e in Shamir secret sharing, which determines the maximum number of clients allowed to dropout or maliciously collude during the training process. Subsequently, some secure parameters and pairwise keys need to be generated to guarantee privacy security in the training process.

Firstly, TA selects the security parameter k as input of initializing function, the outputs are public parameters (q,g,G). Then, TA generates T pairs of public-private keys $\{(s_i^{sk-t},s_i^{pk-t})\}_{t\in\{1,2,\ldots,T\}}$ and a pair of keys (c_i^{sk},c_i^{pk}) for each client $i\in\mathcal{P}_1$. The maximal number of epochs equals to the number of pairs of public-private keys, and one pair of public-private key will be used in one epoch only. (s_i^{sk-t},s_i^{pk-t}) is used to mask the local model trained in the t-th epoch, while (c_i^{sk},c_i^{pk}) is for encrypting and decrypting the messages exchanged between the clients. After all keys are generated, TA sends them to the corresponding client i through a secure channel and broadcasts the tuples of these public keys $\{(i,c_i^{pk},\{t,s_i^{pk-t}\}_{t\in\{1,2,\ldots,T\}})\}_{i\in\mathcal{P}_1}$ in the system.

Next, TA selects a random number $s\in Z_q^*$ as the system private key s_{sys}^{sk}, and calculates $g^s\in G$ as the system public key s_{sys}^{pk}. It is noted that s_{sys}^{sk} should be kept confidential strictly and even not leaked to any model training participants, including the server and all clients. On the contrary, s_{sys}^{pk} should be broadcasted in the system. After that, according to the Shamir secret sharing mechanism, TA generates the secret shares (x_i,s_{sys-i}^{sk}) of the system private key s_{sys}^{sk} for each client $i\in\mathcal{P}_1$ by constructing a polynomial with degree (e-1) through the formula (5). Here, x_i is public and s_{sys-i}^{sk} is kept secret.

When all clients are informed of the training network, related parameters, initialized global model, as well as being distributed the pairwise keys and the secret shares of the system private key, the initialization is completed, and the federated learning process with the privacy preservation can be initiated.

4.3.2 Masking of the local model

After obtaining the related messages, each client starts to train the local model w_i^t with SGD algorithm. Then, it calculates the data quality Q_i^t before the local model is submitted to the server. In order to prevent the local model from being snooped by malicious external adversaries, the server and

the clients, some cryptographic information needs to be exchanged among all clients, so as to achieve the purpose of privacy-preservation.

At first, client i selects the private key s_i^{sk-t} corresponding to the current epoch t, and executes secret shares generation algorithm:

$$s_i^{sk-t} \xrightarrow{\text{SSS.share}(s_i^{sk-t},e,\mathcal{P}_1)} \{(s_{i,j}^{sk-t})\}_{j\in\mathcal{P}_1\backslash\{i\}}$$

to divide s_i^{sk-t} into n shares. It also uses the Diffie-Hellman key agreement function (denoted as f_{D-H}) to calculate symmetric key $c_{i,j} = f_{D\text{–}H}(c_i^{sk}, c_j^{pk})$ with each client $j \in \mathcal{P}_1\backslash\{i\}$. After that, client i encrypts the secret share $s_{i,j}^{sk-t}$ with the symmetric key $c_{i,j}$ to generate the ciphertext $ct_{i,j}^t = Enc_{c_{i,j}}(s_{i,j}^{sk-t})$, and sends $(i, j, ct_{i,j}^t)$ to client j.

When client j receives ciphertexts $\{(i, j, ct_{i,j}^t)\}_{i\in\mathcal{P}_2\backslash\{j\}}$ from at least e clients (marked as $\mathcal{P}_2 \subseteq \mathcal{P}_1$), it calculates the symmetric key $c_{j,i} = f_{D\text{–}H}(c_j^{sk}, c_i^{pk})$ with the corresponding client i, and decrypts $ct_{i,j}^t$ with $c_{j,i}$ to obtain the secret share $s_{i,j}^{sk-t} = Dec_{c_{j,i}}(ct_{i,j}^t)$. Here, client j must store all secret shares safely.

After obtaining the local model expressed by w_i^t and data quality expressed by Q_i^t, client i calculates the symmetric key $s_{i,j}^t = f_{D\text{–}H}(s_i^{sk-t}, s_j^{pk-t})$ with other clients $j \in \mathcal{P}_2\backslash\{i\}$, and takes $s_{i,j}^t$ as the seed of the pseudo-random generator (PRG) to calculate the masked local model according to formula (16).

$$\begin{aligned} y_i^t = w_i^t Q_i^t + \sum_{j\in\mathcal{P}_2\backslash\{i\}, i<j} PRG(s_{i,j}^t) - \\ \sum_{j\in\mathcal{P}_2\backslash\{i\}, i>j} PRG(s_{i,j}^t) + \varphi Q_i^t (\mathrm{mod} R) \end{aligned} \tag{16}$$

In formula (16), φ is a public scale factor which can be preset. $PRG(s_{i,j}^t)$ and φQ_i^t are numbers, while $w_i^t Q_i^t$ is a vector, so $PRG(s_{i,j}^t)$ and φQ_i^t are needed to be repeated to get a vector that matches the dimension of $w_i^t Q_i^t$. Furthermore, it is necessary to quantize Q_i^t to an integer for the subsequent encryption operation. For the floating-point number (Q_i^t), its quantization follows the following two steps: first, the floating-point number is round up to n decimal places, then we expand it to 10^n times to get the quantized integer. That is to say, we ensure that it can only be accurate to n decimal place, and we lost precision in the fractional part. On the contrary, we do an inverse operation on the server side, scaling down the integer value to $1/(10^n)$ of its value, resulting in a floating-point number. The precision parameter n is a tunable parameter, and its value can be adjusted in experiments.

After completing local model masking, client i encrypts the data quality Q_i^t by utilizing the improved ElGamal homomorphic encryption algorithm and the system public key g^s. It randomly selects a number $r_i \in Z_q^*$ to calculate the ciphertext (c_{i1}, c_{i2}) according to formula (17), and then sends the tuple $(i, y_i^t, c_{i1}, c_{i2})$ to the server.

$$Enc_{s_{sys}^{pk}}(Q_i^t) = (c_{i1}, c_{i2}) = (g^{r_i}, 2^{Q_i^t} g^{s r_i}) \tag{17}$$

4.3.3 Aggregation of the local model

After collecting the masked model and the ciphertext $(i, y_i^t, c_{i1}, c_{i2})$ from at least e clients (marked as $\mathcal{P}_3 \subseteq \mathcal{P}_2$), the server starts to aggregate these masked models.

Firstly, it judges whether $\mathcal{P}_3 = \mathcal{P}_2$ is true. If yes, it directly aggregates $\{y_i^t\}_{i\in\mathcal{P}_3}$ and executes the steps described. Otherwise, the server finds out the clients existing in $\mathcal{P}_2$ but not in $\mathcal{P}_3$ (marked as $\mathcal{P}_2\backslash\mathcal{P}_3$). In fact, $\mathcal{P}_2\backslash\mathcal{P}_3$ represents the dropped clients during the uploading process of the masked models. Next, the server broadcasts the identity list in $\mathcal{P}_2\backslash\mathcal{P}_3$ to all clients for enquiring the shares of the dropped clients. After receiving the list, client j submits the shares $\{(s_{i,j}^{sk-t})\}_{i\in\mathcal{P}_2\backslash\mathcal{P}_3}$ to the server voluntarily.

When the server receives the shares $\{(s_{i,j}^{sk-t})\}_{j\in\mathcal{P}_4, i\in\mathcal{P}_2\backslash\mathcal{P}_3}$ from at least e clients (marked as $\mathcal{P}_4$), it executes the secret reconstruction algorithm:

$$\{(s_{i,j}^{sk-t})\}_{j\in\mathcal{P}_4\backslash\{i\}} \xrightarrow{\text{SSS.recons}(e,\{(s_{i,j}^{sk-t})\}_{j\in\mathcal{P}_4\backslash\{i\}})} s_i^{sk-t}$$

to reconstructs the keys $\{(s_i^{sk-t})\}_{i\in\mathcal{P}_2\backslash\mathcal{P}_3}$ for the dropped clients $\mathcal{P}_2\backslash\mathcal{P}_3$. Then, the server calculates symmetric key $s_{i,j}^t = f_{D\text{–}H}(s_i^{sk-t}, s_j^{pk-t})$ between each dropped client i and the other clients $j \in \mathcal{P}_3\backslash\{i\}$ one by one. What we need to pay attention to is that, if there exists a dropout client that recovers in a short time, another private key will be used in the next epoch to ensure security, instead of the reconstructed private key in previous epoch by the server. As we mentioned before, one pair of public-private key will be used in one epoch only, and the purpose is to prevent key leakage in client-dropout situations.

Subsequently, the server aggregates the masked models $\{y_i^t\}_{i\in\mathcal{P}_3}$ from the online clients $\mathcal{P}_3$. In order to offset $\{PRG(s_{i,j}^t)\}_{i\in\mathcal{P}_3, j\in\mathcal{P}_2\backslash\mathcal{P}_3}$ that have been incorporated into masked models of the online clients, the server takes $\{(s_{i,j}^t)\}_{i\in\mathcal{P}_2\backslash\mathcal{P}_3, j\in\mathcal{P}_3}$ as the seeds of the PRGs, and then aggregates all masked local models submitted by clients in $\mathcal{P}_3$ into a value θ, which is shown as formula (18).

$$\begin{aligned} \theta = \sum_{i\in\mathcal{P}_3} y_i^t + \sum_{i\in\mathcal{P}_2\backslash\mathcal{P}_3, j\in\mathcal{P}_3, i<j} PRG(s_{i,j}^t) \\ - \sum_{i\in\mathcal{P}_2\backslash\mathcal{P}_3, j\in\mathcal{P}_3, i>j} PRG(s_{i,j}^t) \\ = \sum_{i\in\mathcal{P}_3} w_i^t Q_i^t + \varphi \sum_{i\in\mathcal{P}_3} Q_i^t \end{aligned} \tag{18}$$

It can be seen from formula (18) that as long as the data quality Q_i^t of each client in $\mathcal{P}_3$ is known, the aggregation result $\sum_{i\in\mathcal{P}_3} w_i^t Q_i^t$ can be easily calculated by $\theta - \varphi \sum_{i\in\mathcal{P}_3} Q_i^t$. Furthermore, the global model is obtained according to formula (15). However, the server cannot acquire Q_i^t directly due to confidentiality of the data quality, but it can still get the encrypted Q_i^t. Through homomorphic decryption algorithm depicted in the following subsection, the global model can be calculated.

4.3.4 Generation of the global model

The decrypting operations on aggregation ciphertext of the data quality $\{Q_i^t\}_{i\in\mathcal{P}_3}$ is performed based on the improved ElGamal homomorphic encryption algorithm.

Firstly, the server aggregates the ciphertexts expressed by$\{(c_{i1}, c_{i2})\}_{i\in\mathcal{P}_3}$ sent by each client $i \in \mathcal{P}_3$ according to

formula (19).

$$c_1 = \prod_{i\in\mathcal{P}_3} c_{i1} = \prod_{i\in\mathcal{P}_3} g^{r_i} = g^{\sum_{i\in\mathcal{P}_3} r_i} = g^{r}$$
$$c_2 = \prod_{i\in\mathcal{P}_3} c_{i2} = \prod_{i\in\mathcal{P}_3} 2^{Q_i^t} \mathrm{g}^{sr_i} = 2^{\sum_{i\in\mathcal{P}_3} Q_i^t} g^{s\sum_{i\in\mathcal{P}_3} r_i} = 2^{\sum_{i\in\mathcal{P}_3} Q_i^t} g^{sr} \tag{19}$$

It can be seen from formula (19) that as long as the system private key s is known, the server can decrypt the aggregation ciphertext (c_1, c_2) and get $\sum_{i\in\mathcal{P}_3} Q_i^t$. However, only TA owns the secret key s, which is not leaked to the server and any clients. Therefore, the server needs to request assistance from e clients to complete the decryption according to the secret reconstruction algorithm.

The server randomly selects e online clients (marked as $\mathcal{P}_5 \subseteq \mathcal{P}_3$) from $\mathcal{P}_3$, and calculates δ_i and $c_1^{\delta_i}$ shown as formula (20), where, x_i is a part of the secret share (x_i, s_{sys-i}^{sk}) which is chosen by TA in the initialization phase and can be accessed by server. Then, the server transmits $c_1^{\delta_i}$ to the corresponding client $i \in \mathcal{P}_5$.

$$\delta_i = \prod_{j\in\mathcal{P}_5, j\neq i} \frac{-x_j}{(x_i - x_j)} \quad , \quad c_1^{\delta_i} = g^{r\delta_i} \tag{20}$$

When client i receives $c_1^{\delta_i}$, it computes λ_i with the secret share s_{sys-i}^{sk} only known by itself, which is shown as formula (21). Then, it sends λ_i back to the server.

$$\lambda_i = (c_1^{\delta_i})^{s_{sys-i}^{sk}} = g^{\prod_{j\in\mathcal{P}_5, j\neq i} \frac{-x_j}{(x_i-x_j)} s_{sys-i}^{sk} \cdot r} \tag{21}$$

After receiving $\{\lambda_i\}_{i\in\mathcal{P}_5}$ from all clients in $\mathcal{P}_5$, the server aggregates them according to formula (22).

$$\begin{aligned} \prod_{i\in\mathcal{P}_5} \lambda_i &= \prod_{i\in\mathcal{P}_5} g^{\prod_{j\in\mathcal{P}_5, j\neq i} \frac{-x_j}{(x_i-x_j)} s_{sys-i}^{sk} \cdot r} \\ &= g^{\left[\sum_{i\in\mathcal{P}_5} \prod_{j\in\mathcal{P}_5, j\neq i} \frac{-x_j}{(x_i-x_j)} s_{sys-i}^{sk}\right] \cdot r} = g^{sr} \end{aligned} \tag{22}$$

So far, the server can calculate the sum of data quality through formula (23).

$$\sum_{i\in\mathcal{P}_3} Q_i^t = \log_2 \frac{c_2}{\prod_{i\in\mathcal{P}_5} \lambda_i} \tag{23}$$

Finally, known the value θ and the scale factor φ, the server can generate the global model by aggregating the local model of the online clients, which is shown as formula (24).

$$\mathbf{w}_{global}^t = \frac{\theta - \varphi \sum_{i\in\mathcal{P}_3} Q_i^t}{\sum_{i\in\mathcal{P}_3} Q_i^t} = \frac{\sum_{i\in\mathcal{P}_3} w_i^t Q_i^t}{\sum_{i\in\mathcal{P}_3} Q_i^t} \tag{24}$$

5 Security Analysis

In our system, the server and all clients are considered to be semi-honest. Therefore, they can execute the secure aggregation process authentically. They neither tamper with the messages transmitted in channel, nor forge public-private keys and secret shares. Therefore, some cryptographic primitives such as public key infrastructure (PKI) and digital signatures are not involved in our system. Given the semi-honest model, we analyze the security of our scheme, including the security of the masking mode and the security of the homomorphic encryption.

5.1 Security of the masking mode

First of all, we take $au_i^t = w_i^t Q_i^t + \varphi Q_i^t$ as the real input. Our scheme aims to mask the real input au_i^t by superimposing a random number on it. According to the properties of the random number, we can see that the masked result is random. Then, in our aggregation model, as long as the sum of the masked results from all clients is equal to the sum of all real inputs, the adversary cannot distinguish the real input from the masked result. Because the masked result hides the real input of each client, the view of the adversary is only the sum of the real inputs instead of any individual real input.

Lemma 1 Given $R, \mathcal{P}, \forall i, j \in \mathcal{P}, au_i \in Z_R$, then

$$\begin{aligned} &\{\{PRG_{i,j} \leftarrow Z_R\}_{i<j}, \ \ \forall i > j, \\ &PRG_{i,j} \leftarrow -PRG_{j,i} : \{au_i + \sum_{j\in\mathcal{P}\setminus\{i\}} PRG_{i,j}\}_{i\in\mathcal{P}}\} \\ \equiv &\{\{\xi_i \leftarrow Z_R\}_{i\in\mathcal{P}}, \ \ s.t. \ \ \sum_{i\in\mathcal{P}} \xi_{i,j} = \sum_{i\in\mathcal{P}} au_i : \{\xi_i\}_{i\in\mathcal{P}}\} \end{aligned}$$

Here, the symbol "$\equiv$" denotes two groups of random variables that are computationally indistinguishable.

Next, we analyze two scenarios, one is the collusion of clients only, and the other is the collusion between some clients and the server. All clients are divided into five categories, namely, $\mathcal{P}_5 \subseteq \mathcal{P}_4 \subseteq \mathcal{P}_3 \subseteq \mathcal{P}_2 \subseteq \mathcal{P}_1$. $\mathcal{P}_1$ represents all clients in the federated learning system; $\mathcal{P}_2$ represents the clients participating in the model masking, which means their pairwise keys are incorporated into the mask; $\mathcal{P}_3$ represents the online clients that successfully uploads the masked models; $\mathcal{P}_4$ represents the clients contributing the secret shares of dropped clients to the server, and $\mathcal{P}_5$ is the clients joining in the decryption of data quality. For simplifying the description, we set the server as S and the colluding parties as $\mathcal{A}$. $view_{\mathcal{A}}^{e,k}(au_{\mathcal{P}}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4, \mathcal{P}_5)$ is a random variable, which represents all view the colluding parties $\mathcal{A}$ can obtain in the entire federated learning process, given threshold e in Shamir secret sharing algorithm and system security parameter k.

Theorem 1. (Honest but curious security, collusion by clients only). Assuming that there is a PPT simulator $\mathcal{C}$ who attempts to break the semantic security of our masking mode with the help of $\mathcal{A}$. If given $e, k, \mathcal{P}, au_{\mathcal{P}}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4, \mathcal{P}_5$ with $|\mathcal{P}| \geq e$, $\mathcal{A} \subseteq \mathcal{P}$, $|\mathcal{A}| < e$, $\mathcal{P}_5 \subseteq \mathcal{P}_4 \subseteq \mathcal{P}_3 \subseteq \mathcal{P}_2 \subseteq \mathcal{P}_1$, the view acquired by the simulator $\mathcal{C}$ is indistinguishable from the view of $\mathcal{A}$.

$$\begin{aligned} &view_{\mathcal{A}}^{e,k}(au_{\mathcal{P}}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4, \mathcal{P}_5) \\ \equiv &\, view_{\mathcal{C}}^{e,k}(au_{\mathcal{A}}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4, \mathcal{P}_5) \end{aligned}$$

Proof. In the first honest but curious threat model, we only refer to the collusion of the clients, while the server is considered to be absolutely honest, which means all clients do not actively tamper with the data transmitted in the channel. The server only transmits the data that each client deserves. In our system, the honest clients only communicate with the server, and they do not share the masked

results with $\mathcal{A}$. Furthermore, even intercepting the masked results, $\mathcal{A}$ can still not eliminate the random numbers from the masked results. Therefore, the joint view of colluding parties $\mathcal{A}$ does not depend on the view of the other honest parties $\mathcal{P}_2\backslash\mathcal{A}$. Although the simulator $\mathcal{C}$ can obtain the real input au_i^t of $\mathcal{A}$ in the simulation, it can only forge the real inputs of the honest user, which means the view of the simulator $\mathcal{C}$ is equivalent to the joint view of $\mathcal{A}$.

Theorem 2. (Honest but curious security, collusion by clients and server). Assuming that there is a PPT simulator $\mathcal{C}$ who attempts to break the semantic security of our masking mode with the help of $\mathcal{A}$. If given e, k, $\mathcal{P}$, $au_{\mathcal{P}}$, $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, $\mathcal{P}_4$, $\mathcal{P}_5$ with $|\mathcal{P}| \geq e$, $\mathcal{A} \subseteq \mathcal{P} \cup \{S\}$, $|\mathcal{A}\backslash\{S\}| < e$, $\mathcal{P}_5 \subseteq \mathcal{P}_4 \subseteq \mathcal{P}_3 \subseteq \mathcal{P}_2 \subseteq \mathcal{P}_1$, the view of the simulator $\mathcal{C}$ is indistinguishable from the view of $\mathcal{A}$.

$$\begin{aligned} &view_{\mathcal{A}}^{e,k}(au_{\mathcal{P}}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4, \mathcal{P}_5) \\ \equiv\ &view_{\mathcal{C}}^{e,k}(au_{\mathcal{A}}, \sigma, \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4, \mathcal{P}_5) \\ &\text{where } \sigma = \sum_{i\in\mathcal{P}_3\backslash\mathcal{A}} au_i \end{aligned}$$

In the second honest but curious threat model, given the real inputs of the colluding party $\mathcal{A}$ and the sum of the real inputs of all honest online clients $\mathcal{P}_3\backslash\mathcal{A}$; then, the simulator $\mathcal{C}$ can still not learn the real input of any individual honest client, except the given view and the view of $\mathcal{A}$. In particular, we emphasize that only the collusion joined by less than e clients cannot break the security of our system. The following proof process abides by this premise.

Proof. (1) Firstly, the simulator $\mathcal{C}$ makes use of $\mathcal{A}$ to get the same view with $\mathcal{A}$.

(2) Next, we simulate each honest client $i \in \mathcal{P}_2\backslash\mathcal{A}$, and replace its symmetric key $c_{i,j} = f_{D-H}(c_i^{sk}, c_j^{pk})$ with a randomly selected key $c_{i,j}^{'} \in G$. Then, we encrypt the secret shares $s_{i,j}^{sk-t}$ with random key $c_{i,j}^{'}$, and transmits the ciphertext to other clients $j \in \mathcal{P}_2\backslash\mathcal{A}$. According to the Decisional Diffie-Hellman assumption, in the case of unknowing private key, $c_{i,j}^{'}$ and $c_{i,j}$ are indistinguishable, so the view of the simulator $\mathcal{C}$ is no more than the colluding parties $\mathcal{A}$.

(3) In this step, we simulate an honest client $i \in \mathcal{P}_2\backslash\mathcal{A}$, and replace its real secret share $s_{i,j}^{sk-t}$ with a randomly selected share $s_{i,j}^{'} \in Z_q^*$, Then send the ciphertext of the random share $s_{i,j}^{'}$ to other clients $j \in \mathcal{P}_2\backslash\mathcal{A}$. When calculating the model mask, we still use the real pairwise keys to generate the random number. If the server wants to restore the real input from the masked result of the honest client, it can only ask clients to upload the secret share $s_{i,j}^{'}$. Since the ciphertext of the real share $s_{i,j}^{sk-t}$ and the ciphertext of the random share $s_{i,j}^{'}$ have indistinguishability under chosen-plaintext attack (IND-CPA), the server cannot distinguish $s_{i,j}^{sk-t}$ from dummy $s_{i,j}^{'}$, therefore it cannot reconstruct the true private key of the honest client. In other words, even with the help of the compromised server, the view of the simulator $\mathcal{C}$ is no more than the previous step.

(4) We simulate an honest client $i \in \mathcal{P}_2\backslash\mathcal{A}$, and replace the real secret share $s_{i,j}^{sk-t}$ of its private key s_i^{sk-t} with a randomly selected share $s_{i,j}^{'} \in Z_q^*$, Then send the random share $s_{i,j}^{'}$ to the colluding client $j \in \mathcal{A}$. When calculating the model mask, we still use the real pairwise keys to generate the random number. According to the Shamir secret sharing algorithm, if $|\mathcal{A}| < e$, the private key cannot be reconstructed. Therefore, as long as the random shares are identical in distribution with the real secret shares, the simulator $\mathcal{C}$ cannot obtain additional information except the view of $\mathcal{A}$.

(5) We generate a random key $s_{v,j}^{'} \in G$ for a special selected client $v \in \mathcal{P}_3\backslash\mathcal{A}$ to substituting its real symmetric key $s_{v,j}^t = f_{D-H}(s_v^{sk-t}, s_j^{pk-t})$ agreed with other clients $j \in \mathcal{P}_2$. Then take $s_{v,j}^{'}$ as the seed of PRG to calculate forged masked results for client v and the other clients $i \in \mathcal{P}_3\backslash\mathcal{A}\backslash\{v\}$, which are respectively shown as formula (25) and formula (26). Finally, the forged masked results y_v^t and y_i^t are sent to the server.

$$\begin{aligned} y_v^{'} = au_v^t + &\sum_{j\in\mathcal{P}_2\backslash\{v\}, v<j} PRG(s_{v,j}^{'}) - \\ &\sum_{j\in\mathcal{P}_2\backslash\{v\}, v>j} PRG(s_{v,j}^{'}) \end{aligned} \tag{25}$$

$$\begin{aligned} y_i^{'} = au_i^t + &\sum_{j\in\mathcal{P}_2\backslash\{v,i\}, i<j} PRG(s_{i,j}^t) - \\ &\sum_{j\in\mathcal{P}_2\backslash\{v,i\}, i>j} PRG(s_{i,j}^t) + PRG(s_{i,v}^{'}) \\ &\text{where } i > v, PRG(s_{i,v}^{'}) := -PRG(s_{i,v}^{'}) \end{aligned} \tag{26}$$

According to the Decisional Diffie-Hellman assumption, $s_{v,j}^t$ and $s_{v,j}^{'}$ are indistinguishable, so $PRG(s_{v,j}^t)$ and $PRG(s_{v,j}^{'})$ are identically distributed, the view of the simulator $\mathcal{C}$ is indistinguishable from the colluding party $\mathcal{A}$.

(6) We substitute $PRG(s_{v,i}^{'})$ computed by the selected client v and other clients $i \in \mathcal{P}_3\backslash\mathcal{A}\backslash\{v\}$ in the previous step with randomly selected number $\gamma_{v,i}^{'}$. As long as $\gamma_{v,i}^{'}$ and $PRG(s_{v,i}^{'})$ are distributed identically, the simulator $\mathcal{C}$ cannot distinguish $\gamma_{v,i}^{'}$ from $PRG(s_{v,i}^{'})$ due to the randomness of PRG. Thus, in this step, the view of the simulator $\mathcal{C}$ is no more than the previous step.

(7) When calculating the masked result for each client $i \in \mathcal{P}_3\backslash\mathcal{A}$, we replace the real input au_i^t with a random number ξ_i^t, but still superimpose the correct mask $PRG(s_{i,j}^t)$. After that, the fake masked result $y_i^{'}$ is sent to the server, instead of the real masked result y_i^t. The calculations of y_i^t and $y_i^{'}$ are respectively shown in formula (27) and formula (28).

$$\begin{aligned} y_i^t = au_i^t + &\sum_{j\in\mathcal{P}_2\backslash\mathcal{A}\backslash\{i\}, i<j} PRG(s_{i,j}^t) - \\ &\sum_{j\in\mathcal{P}_2\backslash\mathcal{A}\backslash\{i\}, i>j} PRG(s_{i,j}^t) \end{aligned} \tag{27}$$

$$\begin{aligned} y_i^{'} = \xi_i^t + &\sum_{j\in\mathcal{P}_2\backslash\mathcal{P}_3\backslash\mathcal{A}\backslash\{i\}, i<j} PRG(s_{i,j}^t) - \\ &\sum_{j\in\mathcal{P}_2\backslash\mathcal{P}_3\backslash\mathcal{A}\backslash\{i\}, i>j} PRG(s_{i,j}^t) \end{aligned} \tag{28}$$

Here, ξ_i^t subjects to the conditional distribution such that $\sum_{i\in\mathcal{P}_3\backslash\mathcal{A}} \xi_{i,j} = \sum_{i\in\mathcal{P}_3\backslash\mathcal{A}} au_i$. According to the Lemma 1, the simulator $\mathcal{C}$ cannot obtain the real input of each client $i \in \mathcal{P}_3\backslash\mathcal{A}$. The view of the simulator is only the sum of their real inputs.

From the simulation process mentioned above, it can be seen that the view of the simulator $\mathcal{C}$ is computationally indistinguishable from the colluding parties $\mathcal{A}$. Therefore, the view of the simulator is no more than the colluding parties $\mathcal{A}$, even in the colluding scenario between some clients and the server.

5.2 Security of the homomorphic encryption

The security analysis in the previous subsection has proved that the masking mechanism can protect the real inputs au_i^t. In this section, if given the real inputs $au_i^t = w_i^t Q_i^t + \varphi Q_i^t$, we only discuss whether the colluding parties $\mathcal{A}$ can decrypt data quality and restore the local model w_i^t through breaking the semantic security of homomorphic encryption. It is worth emphasizing that our threat model still needs to follow the premise that fewer than e clients participate in collusion.

In the encryption process, client i uses the system public key g^s to encrypt data quality and send the ciphertext $(g^{r_i}, 2^{Q_i^t} g^{sr_i})$ to the server. If the server wants to decrypt the data quality of client i, and furtherly snoop its local model w_i^t by equation $w_i^t = au_i^t/Q_i^t - \varphi$, it firstly needs to restore $2^{Q_i^t}$. However, the system private key s is kept secret for the server and all clients. Therefore, if the server expects to restore $2^{Q_i^t}$ without s, it either solves the discrete logarithm problem to deduce r_i from g^{r_i} and then calculates $(g^s)^{r_i}$, or solves the Decisional Diffie-Hellman problem to calculate g^{sr_i} according to g^{r_i} and g^s. So far, the two types of problems are recognized as NP-hard problems, and the server cannot solve them.

When the server cannot restore $s^{Q_i^t}$ by itself, it may collude with other clients to request their assistances for reconstructing the private key s. In our system, although the server also needs to cooperate with online clients to obtain the aggregated result $\sum_{i \in \wp_3} Q_i^t$ of data quality, it just needs to compute g^{sr} instead of reconstructing the private key s. However, the threat model of collusion is different from normal cooperation, and the colluding parties $\mathcal{A}$ aim to get the quality data Q_i^t of an individual honest client. Without the private key s, they can only obtain the sum of data qualities of all honest clients through computing $\sum_{i \in \mathcal{P}_3 \backslash \mathcal{A}} Q_i^t = \sum_{i \in \mathcal{P}_3} Q_i^t - \sum_{i \in \mathcal{A}} Q_i^t$. Then, it seems that reconstructing s is the unique effective way to achieve their purpose. Unfortunately, the number of colluding clients $\mathcal{A} \backslash \{S\}$ is less than threshold e. According to the Shamir secret sharing algorithm, the private key s cannot be reconstructed from less than e shares. Therefore, the server and the client have no ability in decrypting a single Q_i^t, and the local model w_i^t is protected.

In summary, as long as less than e clients participate in collusion, our homomorphic encryption is semantically secure.

6 Performance Analysis

In this part, we suppose that our system consists of a server and n clients, the dimension of the local model is m, and the threshold of the Shamir secret sharing algorithm is e. The theoretical analysis on computational cost and communication cost caused during an epoch of federated learning is performed. All costs brought by the system initialization are neglected, because initialization is executed only once, which would not affect the performance during the subsequent training process. We elaborate on the specific analysis process from the perspective of an individual client and the server respectively.

6.1 Cost analysis of client

6.1.1 Computation cost

The computation cost of each client i in a training epoch is mainly generated by the following five operations:

(1) When the local model is masked, in order to generate the seed of PRG, client i needs to calculate symmetric keys with other $(n-1)$ clients, that means the key agreement function $f_{D\text{–}H}(s_i^{sk-t}, s_j^{pk-t})$ is performed $(n-1)$ times. In the same way, when the client encrypts or decrypts the secret shares $s_{i,j}^{sk-t}$ related with other $(n-1)$ clients, the function $f_{D\text{–}H}(c_i^{sk-t}, c_j^{pk-t})$ is also executed $(n-1)$ times for generating symmetric key. Therefore, key agreement function is executed $2(n-1)$ times in total, and the computation complexity is approximately $O(n)$.

(2) Client i needs to perform $(n-1)$ times encryption operations $Enc_{c_{i,j}}(s_{i,j}^{sk-t})$ on the secret shares distributed to others and $(n-1)$ times decryption operation $Dec_{c_{i,j}}(ct_{j,i}^t)$ on the secret shares received from others. It requires $2(n-1)$ encryption/decryption operations, then, the complexity is approximately $O(n)$.

(3) Client i needs to create $(n-1)$ secret shares of its private key, and the secret share generation algorithm is run $(n-1)$ times, and the complexity is approximately $O(n)$.

(4) Since the output of PRG is a number, while the local model is an m-dimensional vector, in order to facilitate calculation and satisfy better privacy, a PRG needs to be expanded into m dimensions. Furthermore, a masking tuple includes $(n-1)$ numbers of PRGs. Therefore, the total number of expansions is $m(n-1)$ times, then the complexity is $O(mn)$.

(5) If client i is selected by the server to assist decryption, it needs to perform a homomorphic encryption operation on data quality, which is similar to the key agreement function in computation complexity. Therefore, the complexity is approximately $O(1)$.

From the analysis mentioned above, it concludes that the total computation complexity of a client in a training epoch is $O(n+n+n+nm+1) \approx O(nm)$.

6.1.2 Communication cost

The communication cost of each client i in a training epoch mostly caused by four parts:

(1) Client i sends $(n-1)$ numbers of encrypted secret shares to others, and receives $(n-1)$ encrypted secret shares from others.

(2) It sends the masked model and the ciphertext of data quality $(i, y_i^t, \mathrm{c}_{i1}, \mathrm{c}_{i2})$ to the server.

(3) Assuming that there are d clients dropping out during the uploading of the masked model, and the client i is willing to contribute its shares, then it sends the secret shares of d dropped clients to the server. In the worst case,

if $(n-e)$ clients drop out, client i needs to submit $(n-e)$ shares.

(4) If the client i is selected to assist the server in homomorphic decryption, it receives $c_1^{\delta_i}$ sent by the server and responses λ_i to the server.

We set that the length of the public key and private key are a_k and a_s, respectively. The range of the elements in the masked model is R, and the length of ciphertext of data quality is a_c. According to the formula (20) and (21), the length of $c_1^{\delta_i}$ and λ_i are equivalent to the length of the public key. Then, the total cost caused by the four steps mentioned above is $[2(n-1)+d]a_s + 2a_k + a_c + m\lceil \log_2 R \rceil$. In real scenarios, the threshold e is usually much smaller than the number n of clients. The probability that the client i participates in homomorphic decryption is low. If it is unwilling to upload shares, the last two steps would not be executed. At this time, the total communication cost is minimum $2(n-1)a_s + a_c + m\lceil \log_2 R \rceil$.

6.2 Cost analysis of server

6.2.1 Computation cost

The computation cost of server in a training epoch is mainly divided into four categories:

(1) If there are d clients dropping out, the server needs to recover the keys for them. Therefore, it needs to perform d times secret reconstruction operations, and the complexity is $O(d)$.

(2) In order to offset the PRG that is related to the dropped clients and has been incorporated into the masked model of online clients, it is necessary to calculate $(n-d)$ numbers of symmetric keys for each dropped client. Then, the server needs to perform $(n-d)d$ times of key agreement function $s_{i,j}^t = f_{D\text{–}H}(s_i^{sk-t}, s_j^{pk-t})$, the complexity is $O(nd)$.

(3) Each symmetric key generated in the previous step is taken as the seed of PRG. Then, the server needs to calculate $(n-d)d$ times $PRG(s_{i,j}^t)$; furthermore, each $PRG(s_{i,j}^t)$ needs to be expanded into m dimensions. So, the complexity is $O((n-d)dm) \approx O(nmd)$.

(4) The server performs homomorphic decryption operation on the data quality of $(n-d)$ online clients including $2(n-d)$ times multiplications of ciphertexts, e times calculation of δ_i and $g^{r\delta_i}$, e times multiplications of λ_i and 2 times division. Usually, d and e are much smaller than n, therefore, the total computation complexity is $O(2(n-d)+e+e+2) \approx O(n)$.

According to the analysis mentioned above, the total computation complexity of the server in a training epoch is $O(d + nd + nmd + n) \approx O(nmd)$. When the number of dropped clients reaches the maximum allowed in our system, i.e., $(n-e)$, the maximum cost of server is $O(nm(n-e)) \approx O(mn^2)$. If no client drops out, Step (1)-(3) can be ignored, and the computation complexity is the lowest $O(n)$.

6.2.2 Communication cost

The communication cost of server in a training epoch is composed of four aspects:

(1) Assuming that there exist d dropped clients, the server receives the masked models and the ciphertext of data quality $(i, y_i^t, c_{i1}, c_{i2})$ from $(n-d)$ online clients.

(2) The server broadcasts an identities list of dropped clients to all online clients.

(3) For reconstructing private keys of dropped clients, the server receives secret shares sent by e online clients, and each client sends d shares, then the server receives ed secret shares in total.

(4) In order to complete the homomorphic decryption of data quality, the server requests assistance from e online clients. It sends $c_1^{\delta_i}$ e times and receives λ_i e times in this process.

If the cost caused by broadcasting the identities list in Step (2) can be ignored, the total communication cost of the server is $eda_s + 2ea_k + (n-d)a_c + (n-d)m\lceil \log_2 R \rceil$. In the case of no dropped clients, Step (3) will not be executed, and the communication cost is $2ea_k + na_c + nm\lceil \log_2 R \rceil$.

7 Performance Evaluation

7.1 Datasets

For the effectiveness evaluation, the dataset we used is the HAM10000 ("Human Against Machine with 10,000 training images") dataset [47], consists of 10,015 dermatoscopic images of common pigmented skin lesions. It has 7 different class of skin cancer, namely: 1) Nevus, 2) Melanoma, 3) Pigmented Bowen's, 4) Basal Cell Carcinoma, 5) Pigmented Benign Keratoses, 6) Vascular, 7) Dermatofibroma. We use only the provided images without any usage of meta-data or external datasets.

Due to the imbalance of categories in the dataset, in order to improve the accuracy of the classification model and avoid overfitting, we expanded the original HAM10000 dataset and applied some data augmentation techniques. We modified the training data with small transformations to reproduce the variations. Several data augmentation techniques were applied during training, such as randomly rotating some training images by 5 degrees, randomly vertical and horizontal flip some training images by 10%. By applying the above transformations to our training data, we can expand the number of training images to 90,135, and the size of each image is 28*28 pixels.

Then, we split the expanded dataset into training set (80,000 images) and testing set (10,135 images) with some randomness. Due to the needs of federated learning, we split the training set into multiple subsets of approximately the same size and distributed them to each client to train a local model. The training dataset owned by each client

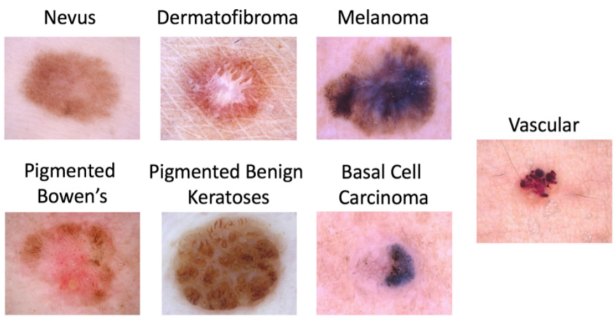


Fig. 2. Samples from the HAM10000 dataset.

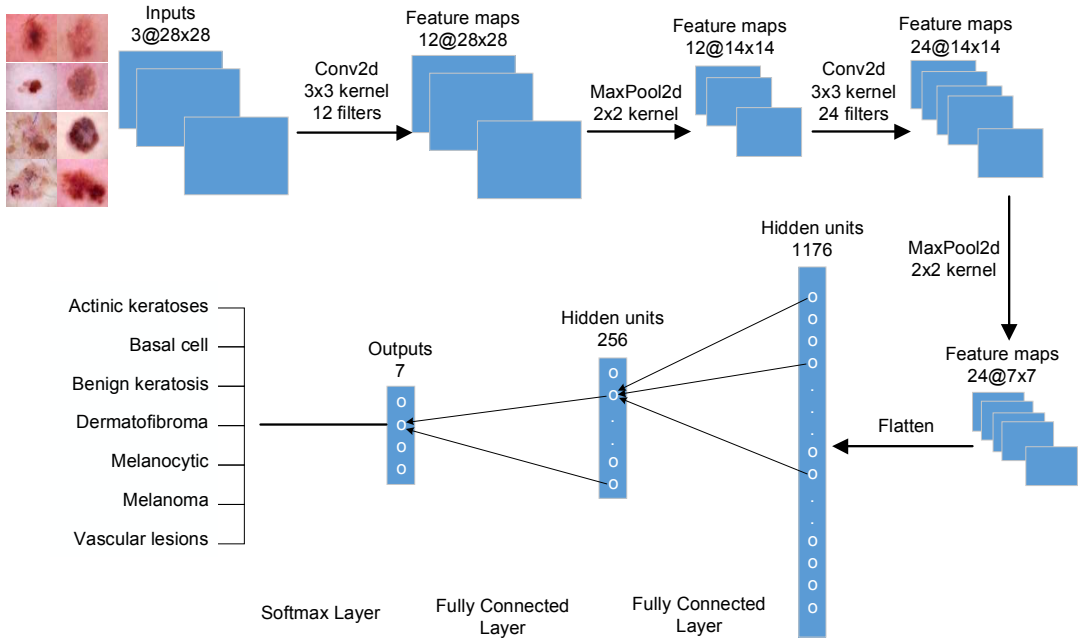


Fig. 3. Deep convolution neural networks for skin cancer detection in each client.

is independent and cannot be leaked, ensuring the security and privacy of the image. Furthermore, we split the train set owned by each client into two parts: 90% is used to train the model, and 10% is the validation set using which the model is evaluated. We encode labels which are 7 different classes of skin cancer (expressed by $0-6$), and we need to encode these labels to one-hot vectors.

In this section, we introduce the steps to detect skin cancer types using deep convolution neural networks. Because this paper focuses on the privacy protection scheme, the network structure of the DNN used is just the conventional multiple convolutional layers, pooling layers, and fully connected layers. It does not pay too much attention to optimize the model, which can be fine-tuned to produce more accurate scores in the future. This paper only considers the simple optimization, such as data augmentations, and the data quality given to different clients when performing parameter updates.

The designed DNN network architecture is shown in Fig. 3, which is composed of two convolution layers, two MaxPooling layers, two full-connected layers, one Softmax layer. A total of 36 kernel filters are designed for convolution layer. The total number of parameters in the local model is 306,063. Model's parameters are initialized randomly, and then distributed to all clients. Each local model is trained by using local training dataset. The accuracy of the models is evaluated on the test dataset by the server.

7.2 Comparison of schemes

We compared the following five schemes, and the functionality comparison in terms of dropout support and homomorphic encryption is listed in Table I.

- Scheme I: the federated learning scheme named FedAvg proposed by McMahan et al. [5], which utilizes weighted averaging for parameter aggregation and updates without any additional privacy protection methods, such as homomorphic encryption.
- Scheme II: the federated learning scheme using average-based parameter aggregation and updates, and it could tolerate a certain degree of client dropout, which was proposed by Bonawitz et al. [9].
- Scheme III: the federated learning scheme named FedOpt proposed by Asad et al. [10], which designed a sparse compression algorithm for efficient communication and also integrated the homomorphic encryption.
- Scheme IV: the federated learning scheme based on homomorphic encryption for all parameters, proposed by Fang et al. [11], but could not tolerate a certain degree of client offline.
- Scheme V: the federated learning scheme proposed in this paper, which supports homomorphic encryption-based privacy protection and could also tolerate a certain degree of client dropout.

TABLE 1
Functionality comparison of five schemes.

Scheme	dropout support	homomorphic encryption
Scheme I	$\surd$	$\times$
Scheme II	$\surd$	$\times$
Scheme III	$\times$	$\surd$
Scheme IV	$\times$	$\surd$
Scheme V	$\surd$	$\surd$

7.3 Comparison result and analysis

In this section, we present the effectiveness and efficiency of the proposed federated learning scheme through comparison of performance evaluation results. We implemented the prototype system of the proposed federated learning

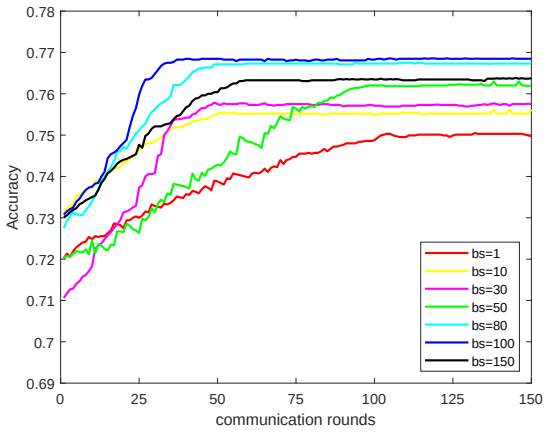


Fig. 4. Accuracy as the increase of communication rounds with different mini-batch size.

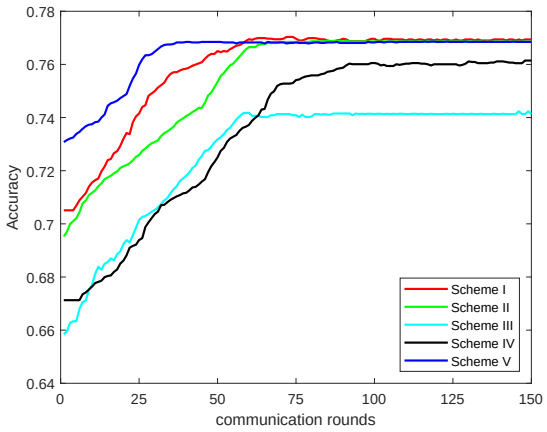


Fig. 5. Accuracy as the increase of communication rounds for five schemes.

scheme in Java language, including three main components, parameter learning in local, homomorphic encryption and parameter updates. Using ElGamal encryption algorithm, a homomorphic encryption strategy is designed, which changes ElGamal cryptosystem from satisfying multiplication homomorphism to satisfying additive homomorphism, implemented by encrypting the exponential form of the plaintexts instead of original plaintexts. For the efficiency evaluation analysis, we run the federated learning on a Linux server equipped with Intel Xeon E5-1650 CPU (6 cores, 12 threads, 3.5GHz) and 32 GB RAM, and the operating system is Ubuntu 18.04. We run the server program to control the learning and run multi-client programs in the same Linux server (the number of client is varied from 10 to 30). Since all client runs in the same server with only one IP address, we assign a unique port for each client. Communications is implemented by Java Remote Procedure Call (RPC) framework. The services encapsulated by clients is called by the server which controls key distribution, model parameter distribution, model training, and parameter aggregation. In order to ensure the fairness of the comparison, performance evaluation of the five schemes (Scheme I, II, III, IV and V) are performed using the same hardware and software environment, the same datasets, preprocessing and distribution methods, and the same library and multi-client communication environment. The homomorphic encryption and DNN involved in the five schemes use the same programming code. The value of parameter for quantization in our experiment is $n = 2$, which means it can only be accurate to 2 decimal places.

7.3.1 Accuracy

In this evaluation, during each training epoch, all clients communicate with the server for one time. The mini-batch size is denoted as bs, and the value of bs is varied from 1, 10, 30, 50, 80, 100 to 150. The learning rate is 0.01, and the number of clients is 10. As shown in Fig. 4, we use this comparative experiment to show that the accuracy of model classification increases with the increase of communication rounds when we set different bs. This figure also indicates the convergence rate. In addition, the accuracy of the model does not increase monotonically as the value of bs increases. When the mini-batch size is set to 100, we obtained the best accuracy. Thus, in the following evaluation, we also set the value of mini-batch size to 100.

Then, we compared the accuracy for the five schemes, as shown in Fig. 5. For Scheme V, we obtained a maximal accuracy of around 76.9% in the test set. However, due to the requirements of homomorphic calculations, the model parameters (data quality in the proposed scheme) will be quantified by integers, so the accuracy of the model will be lost to a certain extent for all homomorphic encryption-based schemes (Scheme III, IV, V). From the overall performance, Scheme III and IV are worse than others. The effect of the proposed scheme is similar to Scheme I and II.

7.3.2 Computation cost

We evaluated the training time of different schemes. As you can see in Fig. 6, as the number of communication rounds increases, the training time becomes significantly longer. We use this comparative experiment to illustrate the time cost of homomorphic encryption and decryption. A tentative conclusions has been drawn that homomorphisms will bring time cost. Compared with Bonawitz et al.'s scheme which doesn't rely on homomorphic encryption, the time for the proposed scheme has been increased by 15.03% when the communication round reaches 300. However, it can ensure the safety of model parameters, and the server cannot derive model parameters, which is also supported by Bonawitz et al.'s scheme.

7.3.3 Communication cost

We evaluated the communication cost of five schemes. In order to compare them, the result of Scheme V is regarded as a benchmark for comparison. In order to avoid the curves overlapping and present more clearly, we calculate the difference between Scheme I, II, III, IV and Scheme V by subtraction, which is shown in Fig. 7. Since each client obtains a subset of the original data set in the federation learning, the communication cost mainly caused by parameter aggregations and key agreements. Scheme III obtained the worse result in terms of communication cost, and our scheme is a little better than Scheme IV. Scheme I obtained the lowest communication cost, due to its simplicity.

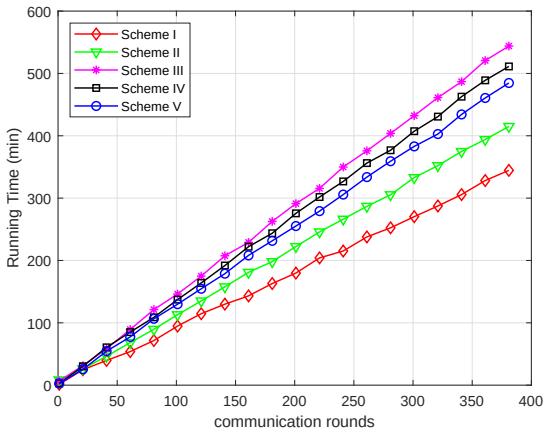


Fig. 6. Server running time as the increase of communication rounds.

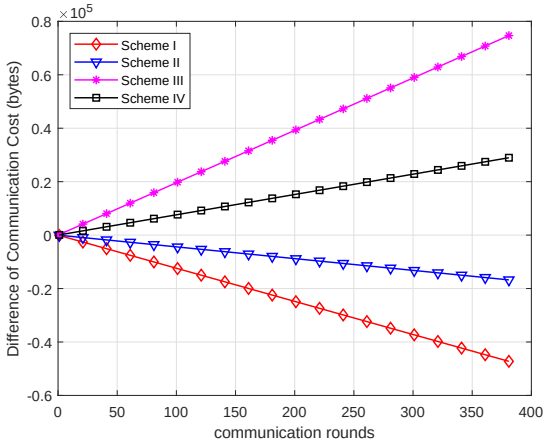


Fig. 7. Difference of communication cost as the increase of communication rounds.

7.3.4 *Client dropout*

Finally, we varied the fraction of dropout clients. Fig. 8 presents the server running time as the increase of dropout rates. Similar to Bonawitz et al.'s scheme, for each dropped client, the server must remove that client's masks, through the secret reconstruction algorithm to reconstructs the keys for the dropped clients, which has been presented in Section 4. It can be observed clearly from this figure that higher dropout rate brings higher cost of dealing with dropped clients. In the case of 30 clients, even if the dropout rate reaches 30%, the time is only increased by 8.29% compared with the situation of no dropout client.

8 Conclusion

In this paper, we have proposed a novel federated learning scheme for privacy-perseveration in IoT-based healthcare applications. A weighted average algorithm based on data quality is proposed to replace the traditional weight calculation method based on the amount of data. A novel masking scheme based on homomorphic encryption and the secure multi-party computation is proposed for federated learning. Moreover, the homomorphic encryption scheme does not encrypt every model parameter because the model in deep learning usually has high dimensions, and the

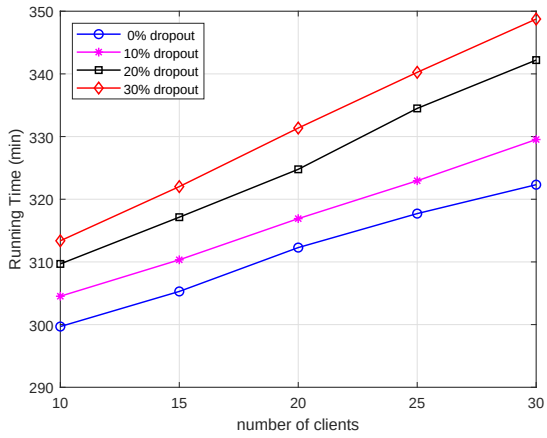


Fig. 8. Server running time under different dropout rates as the increase of client number.

homomorphic encryption on such high-dimensional data would bring huge computation overhead. There is only a variable called data quality to be encrypted for each client in each training epoch in our scheme. Therefore, the scheme proposed in the paper would not cause the computation overhead to increase sharply. An appropriate adjustment to the ElGamal encryption algorithm is made so as to change the characteristics of the algorithm from multiplicative homomorphism to additive homomorphism, which is suitable for the scheme proposed in the paper. Moreover, a flexible dropout-tolerable and participants collusion-resistible solution is provided in our scheme by employing Diffie-Hellman key exchange and Shamir secret sharing algorithm.

Moreover, this paper focuses on privacy protection schemes and classification accuracy. In our experiments, we are able to detect lesion cell type with an accuracy of more than 76.9%. The model can also be fine-tuned to perform and get a better accuracy in the future. However, we have not considered the situation of heterogenous clients with resource-constrained hardware and asynchronous federated learning. The proposed scheme still needs to be tuned in heterogenous environment to obtain a high efficiency. Furthermore, malicious server is out of the research scope of our proposed aggregation in federated learning, and we have not considered that the aggregated model is tampered or forged by the server. Verifiable aggregation in federated learning will also be one of our future work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant no. 61872138, and the Natural Science Foundation of Hunan Province under grant no. 2021JJ30278.

References

[1] M. Arshad, M. A. Khan, U. Tariq, A. Armghan, F. Alenezi, M. Y. Javed, S. M. Aslam, and S. Kadry, "A computer-aided diagnosis system using deep learning for multiclass skin lesion classification," *Comput. Intell. Neurosci.*, vol. 2021, pp. 9 619 079:1–9 619 079:15, 2021.

[2] J. Shen, H. Yang, P. Vijayakumar, and N. Kumar, "A privacy-preserving and untraceable group data sharing scheme in cloud computing," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[3] H. Zanddizari, N. Nguyen, B. Zeinali, and J. M. Chang, "A new preprocessing approach to improve the performance of cnn-based skin lesion classification," *Medical Biol. Eng. Comput.*, vol. 59, no. 5, pp. 1123–1131, 2021.

[4] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, ser. Proceedings of Machine Learning Research, vol. 54. PMLR, 2017, pp. 1273–1282.

[6] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, and S. Bakas, "Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data," *Scientific Reports*, vol. 10, 2020.

[7] B. Liu, B. Yan, Y. Zhou, Y. Yang, and Y. Zhang, "Experiments of federated learning for COVID-19 chest x-ray images," *CoRR*, vol. abs/2007.05592, 2020.

[8] R. Wang, J. Lai, Z. Zhang, X. Li, P. Vijayakumar, and M. Karuppiah, "Privacy-preserving federated learning for internet of medical things under edge computing," *IEEE Journal of Biomedical and Health Informatics*, 2022.

[9] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA*. ACM, 2017, pp. 1175–1191.

[10] M. Asad, A. Moustafa, and T. Ito, "Fedopt: Towards communication efficiency and privacy preservation in federated learning," *Applied Sciences*, vol. 10, no. 2020.

[11] C. Fang, Y. Guo, N. Wang, and A. Ju, "Highly efficient federated learning with strong privacy preservation in cloud computing," *Comput. Secur.*, vol. 96, p. 101889, 2020.

[12] X. Li, J. He, P. Vijayakumar, X. Zhang, and V. Chang, "A verifiable privacy-preserving machine learning prediction scheme for edge-enhanced hcpss," *IEEE Transactions on Industrial Informatics*, 2021.

[13] P. Vijayakumar, V. I. Chang, L. J. Deborah, B. Balusamy, and S. G. Padinjappurathu, "Computationally efficient privacy preserving anonymous mutual and batch authentication schemes for vehicular ad hoc networks," *Future Gener. Comput. Syst.*, vol. 78, pp. 943–955, 2018.

[14] H. Yang, J. Shen, T. Zhou, S. Ji, and P. Vijayakumar, "A flexible and privacy-preserving collaborative filtering scheme in cloud computing for vanets," *ACM Trans. Internet Technol.*, vol. 22, no. 2, pp. 1–19, 2022.

[15] Z. Xu, W. Liang, K.-C. Li, J. Xu, A. Y. Zomaya, and J. Zhang, "A time-sensitive token-based anonymous authentication and dynamic group key agreement scheme for industry 5.0," *IEEE Transactions on Industrial Informatics*, 2021.

[16] A. Bhowmick, J. C. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," *CoRR*, vol. abs/1812.00984, 2018.

[17] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019, pp. 739–753.

[18] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 3–18.

[19] M. Fredrikson, E. Lantz, S. Jha, S. M. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*. USENIX Association, 2014, pp. 17–32.

[20] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019, pp. 691–706.

[21] L. Zhu and S. Han, "Deep leakage from gradients," in *Federated Learning - Privacy and Incentive*, ser. Lecture Notes in Computer Science. Springer, 2020, vol. 12500, pp. 17–31.

[22] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *2020 USENIX Annual Technical Conference, USENIX ATC 2020, July 15-17, 2020*. USENIX Association, 2020, pp. 493–506.

[23] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *CoRR*, vol. abs/1711.10677, 2017.

[24] C. Fang, Y. Guo, Y. Hu, B. Ma, L. Feng, and A. Yin, "Privacy-preserving and communication-efficient federated learning in internet of things," *Comput. Secur.*, vol. 103, p. 102199, 2021.

[25] H. Zhu, R. Wang, Y. Jin, K. Liang, and J. Ning, "Distributed additive encryption and quantization for privacy preserving federated deep learning," *CoRR*, vol. abs/2011.12623, 2020.

[26] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. Deng, "Privacy-preserving federated deep learning with irregular users," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020.

[27] B. Choi, J. Sohn, D. Han, and J. Moon, "Communication-computation efficient secure aggregation for federated learning," *CoRR*, vol. abs/2012.05433, 2020.

[28] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, "Calibrating noise to sensitivity in private data analysis," *J. Priv. Confidentiality*, vol. 7, no. 3, pp. 17–51, 2016.

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

[30] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong, "Personalized federated learning with differential privacy," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9530–9539, 2020.

[31] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *CoRR*, vol. abs/1905.06731, 2019.

[32] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Int. J. Medical Informatics*, vol. 112, pp. 59–67, 2018.

[33] J. Lee, J. Sun, F. Wang, S. Wang, C.-H. Jun, and X. Jiang, "Privacy-preserving patient similarity learning in a federated environment: Development and analysis," *JMIR Med Inform.*, vol. 6, no. 2, p. e20, 2018.

[34] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das, "Differential privacy-enabled federated learning for sensitive health data," *CoRR*, vol. abs/1910.02578, 2019.

[35] N. Rieke, J. Hancox, W. Li, F. Milletari, H. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. H. Maier-Hein, S. Ourselin, M. J. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *CoRR*, vol. abs/2003.08119, 2020.

[36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[38] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 12:1–12:19, 2019.

[39] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*. mlsys.org, 2019.

[40] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[41] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[42] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, vol. 4, no. 11, pp. 169–180, 1978.

[43] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, 1985.

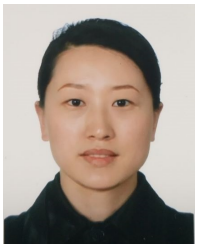
[44] C. Miao, W. Jiang, L. Su, Y. Li, S. Guo, Z. Qin, H. Xiao, J. Gao, and K. Ren, "Cloud-enabled privacy-preserving truth discovery in crowd sensing systems," in *Proceedings of the 13th ACM Conference*

on Embedded Networked Sensor Systems, SenSys 2015, Seoul, South Korea, November 1-4, 2015. ACM, 2015, pp. 183–196.

[45] G. Xu, H. Li, C. Tan, D. Liu, Y. Dai, and K. Yang, "Achieving efficient and privacy-preserving truth discovery in crowd sensing systems," *Comput. Secur.*, vol. 69, pp. 114–126, 2017.

[46] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*. USENIX Association, 2017, pp. 629–647.

[47] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset: A large collection of multi-source dermatoscopic images of common pigmented skin lesions," *CoRR*, vol. abs/1803.10417, 2018.



Li Zhang received the M.S. degree in communication and information systems from Wuhan University of Technology, China, in 2007. She is currently pursuing the Ph.D. degree in software engineering at Hunan University of Science and Technology, China. Her research interest includes network security for Internet of Things and edge computing.

Jianbo Xu received the M.S. degree from the Department of Computer Science and Technology, National University of Defense Technology, China, in 1994, and the Ph.D. degree from the College of Computer Science and Electronic Engineering, Hunan University, China, in 2003. Since 2003, he has been a Professor with the School of Computer Science and Engineering, Hunan University of Science and Technology, China. His research interests include network security and distributed computing.

Pandi Vijayakumar (Senior Member, IEEE) received the B.E. degree in Computer Science and Engineering from Madurai Kamaraj University, Madurai, India, in 2002, the M.E. degree in Computer Science and Engineering from the Karunya Institute of Technology, Coimbatore, India, in 2005, and the Ph.D. degree in Computer Science and Engineering from Anna University, Chennai, India, in 2013. He is the former Dean and currently an Assistant Professor with the Department of Computer Science and Engineering, University College of Engineering Tindivanam, Melpakkam, India, which is a constituent college of Anna University Chennai, India. He has seventeen years of teaching experience and he has produced four Ph.D. candidates successfully. He has also authored and co-authored more than 100 quality papers in various IEEE transactions/journals, ACM transactions, Elsevier, IET, Springer, Wiley and IGI Global journals. He is serving as Associate Editor in many SCI indexed journals namely International Journal of Communication Systems (Wiley), PLOS One, International Journal of Semantic Web and Information Systems (IGI Global), and Security and Communication Networks (Wiley—Hindawi). Moreover, He is serving as an Academic Editor in the International Journal of Organizational and Collective Intelligence (IGI Global), International Journal of Software Science and Computational Intelligence (IGI Global), International Journal of Cloud Applications and Computing (IGI Global), International Journal of Digital Strategy, Governance, and Business Transformation (IGI Global) and Security and Privacy (Wiley). He is also serving as a Technical Committee member in the journal Computer Communications (Elsevier). Recently, He was elevated to Editor-in-Chief Position in the journal Cyber Security and Applications (KeAi—Elsevier). Till now he has authored four books for various subjects that belong to the Department of Computer Science and Engineering. He is a senior member of IEEE. He is also listed in the world's Top 2% Scientists for citation impact during the calendar year 2020 by Stanford University.

Pradip Kumar Sharma (M'18, SM'21) is an Assistant Professor in Cybersecurity in the Department of Computing Science at the University of Aberdeen, UK. He received his Ph.D. in CSE (August 2019) from the Seoul National University of Science and Technology, South Korea. He also worked as a Postdoctoral Research Fellow in the Department of Multimedia Engineering at the Dongguk University, South Korea. He was a Software Engineer and involved on variety of projects, proficient in building largescale complex data warehouses, OLAP models, and reporting solutions that meet business objectives and align IT with business. He has published many technical research papers in leading journals from IEEE, Elsevier, Springer, Wiley, MDPI, etc. Some of his research findings are published in the most cited journals. He has been an expert reviewer for IEEE Transactions, Elsevier, Springer, and MDPI journals and magazines. He is listed in the world's Top 2% Scientists for citation impact during the calendar year 2019 by Stanford University. Also, he received a top 1% reviewer in computer science by Publons Peer Review Awards 2018 and 2019, Clarivate Analytics. He has also been invited to serve as the technical program committee member and chair in several reputed international conferences such as IEEE CNCC 2021, ACM ICDCN 2021, CSA 2021, CSA 2020, IEEE ICC 2019, IEEE MENACOMM'19, 3ICT 2019, etc. Currently, he is Associate Editor of Peer-to-Peer Networking and Application (PPNA), Human-centric Computing and Information Sciences (HCIS), Electronics (MDPI), and Journal of Information Processing Systems (JIPS) journals. He has been serving as a Guest Editor for international journals of certain publishers such as IEEE, Elsevier, Springer, MDPI, JIPS, etc. His current research interests are focused on the areas of Cybersecurity, Blockchain, Edge computing, SDN, and IoT security.

Uttam Ghosh (Senior Member, IEEE) received the Ph.D. degree in electronics and electrical engineering from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 2013. He is working as an Assistant Professor of the Practice with the Department of Computer Science, Vanderbilt University, Nashville, TN, USA. He has postdoctoral experience with the University of Illinois at UrbanaChampaign, Champaign, IL, USA; Fordham University, New York, NY, USA; and Tennessee State University, Nashville, TN, USA. He has been awarded the 2018-2019 Junior Faculty Teaching Fellow and has been promoted to a Graduate Faculty position with Vanderbilt University. He has published over 80 papers in reputed international journals, including IEEE Transactions, Elsevier, Springer, IET, Wiley, InderScience, and IETE, and also in top international conferences sponsored by IEEE, ACM, and Springer. He is co-editing ten books on Smart IoT, Security, and Data Analysis with CRC Press and Springer. His main research interests include cybersecurity, wireless 5G networks, SDN, energy delivery systems, and cloud computing. Dr. Ghosh has served as a technical program committee member at renowned international conferences. He is serving as an Associate Editor for the International Journal of Computers and Applications (Taylor & Francis) and also a reviewer for international journals, including IEEE Transactions, Elsevier, Springer, and Wiley. He serves as a Guest Editor for special issues with IEEE Sensors, IEEE Transaction on Industrial Informatics, IEEE Journal of Biomedical and Health Informatics, IEEE Transaction on Network Science and Engineering, ACM Transactions on Internet Technology, Computer Communications (Elsevier), Multimedia Tools and Applications (Springer), Internet Technology Letters (Wiley), Sensors (MDPI), and Future Internet (MDPI). He has conducted several sessions and workshops related to cyberphysical systems (CPS), SDN, IoT, and smart cities as the Co-Chair at top international conferences, including IEEE Globecom 2020-2021, IEEE MASS 2020, SECON 20192020, CPSCOM 2019, and ICDCS 2017. He is a member of ACM and Sigma-Xi.