Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 20(2): 2261–2279.
DOI: 10.3934/mbe.2023106
Received: 05 August 2022
Revised: 27 September 2022
Accepted: 07 November 2022
Published: 17 November 2022

Research article

Research on medical data security sharing scheme based on homomorphic encryption

Lihong Guo*, **Weilei Gao**, **Ye Cao** and **Xu Lai**

Department of Information and Communications Engineering, Nanjing Institute of Technology, Nanjing 211167, China

* **Correspondence:** Email: guolihongnj@163.com; Tel: +8613952090181.

Abstract: With the deep integration of "AI + medicine", AI-assisted technology has been of great help to human beings in the medical field, especially in the area of predicting and diagnosing diseases based on big data, because it is faster and more accurate. However, concerns about data security seriously hinder data sharing among medical institutions. To fully exploit the value of medical data and realize data collaborative sharing, we developed a medical data security sharing scheme based on the C/S communication mode and constructed a federated learning architecture that uses homomorphic encryption technology to protect training parameters. Here, we chose the Paillier algorithm to realize the additive homomorphism to protect the training parameters. Clients do not need to share local data, but only upload the trained model parameters to the server. In the process of training, a distributed parameter update mechanism is introduced. The server is mainly responsible for issuing training commands and weights, aggregating the local model parameters from the clients and predicting the joint diagnostic results. The client mainly uses the stochastic gradient descent algorithm for gradient trimming, updating and transmitting the trained model parameters back to the server. In order to test the performance of this scheme, a series of experiments was conducted. From the simulation results, we can know that the model prediction accuracy is related to the global training rounds, learning rate, batch size, privacy budget parameters etc. The results show that this scheme realizes data sharing while protecting data privacy, completes the accurate prediction of diseases and has a good performance.

Keywords: data security sharing; federated learning; homomorphic encryption; model; algorithms

1. Introduction

With the rapid development of information technology, all kinds of data break the restrictions of time and space and accumulate in different fields to form data treasures. It breeds huge commercial value and unlimited potential. Therefore, the intelligent use of data and the maximization of data value have become the focus of competition in various industries. But, in the face of the complex network environment, it must ensure data security while exploiting the value of data. Therefore, the research on data sharing based on privacy protection has become a major challenge [1–3].

At present, there are many data sharing schemes, the most widely used of which are the centralized processing mode and distributed processing mode.

In the centralized processing mode, all participants need to share their data, that is to say, this kind of centralized mode requires the participant to upload their data to the server, and all of the data are applied for centralized learning or training on the server. If the server is malicious or the server is vulnerable to external attacks, then there is a risk of data privacy leakage. So, this kind of sharing mode undoubtedly reduces the possibility of data sharing among different participants.

To solve the above problems of the centralized processing mode and avoid privacy leakage, a new distributed processing mode has emerged. Google proposed a new "distributed training model", i.e., a federal learning model. In this model, the data of the client does not leave the local area and all model training is performed locally. After the local model training, the trained parameters are uploaded to the server. Then, the server receives and combines all of the trained parameters for unified aggregation, and it will redistribute the new results to the local level, where a new model is updated. Essentially, federated learning is a type of distributed machine learning, and its most important feature is that the user's data are stored locally by the client so that the original data of each participant are not leaked during the process of cooperation training. Federated learning organizes the process of model training through distributed mode, and the whole training process only moves the model, not the data, so that the model using the framework of federated learning can ensure that multi-institutional data are jointly modeled under the premise of security without revealing privacy [4,5].

The federated learning model can safely share data, and it achieves the purpose of distributed training based on all data being localized, but it also has some loopholes and is vulnerable to attacks [6–8]. For example, after the client is trained locally, it will upload the training parameters. During communication between the clients and the server, the parameters are transmitted in plain text. If these parameters are obtained by the attacker, the client's data information may also be inferred through the parameters. Studies have shown that a malicious participant can infer sensitive user data backward based on the differences in the federal learning gradient parameters in each round. Therefore, parameters that are not protected by encryption are compromised, and to a certain extent, can be targeted for attack, thus indirectly compromising users' private data [9,10].

In recent years, with the deep integration of "AI + medicine", medical data are playing an increasingly valuable role, with healthcare data sharing taking on greater significance. "Baidu Doctor" was launched in 2014, enabling patients to make appointments with doctors in a short period of time, reducing the cost of medical care and optimizing medical resources. AliCloud established the "Peace of Mind on the Cloud" alliance, which combines AliCloud's massive data and powerful computing power to provide accurate medical services for everyone. The basis for all of this is the security of medical data and the safe use of data on the basis of security. In particular, the development of medical imaging and the combination of big data and medical imaging allows for more accurate prediction of disease.

In order to fully utilize the value of medical data and protect the privacy of individuals while enabling data sharing, medical data were used as the prototype data, and a data sharing scheme based on homomorphic encryption (HE) was designed through the use of federated learning, focusing on feasibility and effectiveness. This scheme can protect the participant's data privacy and realize data security sharing.

In summary, the contributions of this paper are as follows:

1) It proposes a data security sharing scheme based on the C/S communication mode and presents a federated learning architecture that uses HE to protect training parameters.

2) It introduces a distributed parameter update mechanism in the process of training, and the server is responsible for issuing training commands and parameters, aggregating the local model parameters uploaded by the client. On the client side, the client uses the stochastic gradient descent algorithm for gradient trimming and updating.

3) Taking medical data as the test dataset, a series of experiments have been designed to assess the performance of this scheme, and the influence factor has been analyzed from the predicted results.

The rest of the paper is organized as follows. In Section 2, this paper discusses and summarizes related work. The related preliminaries are in Section 3. Section 4 describes the construction, algorithms and execution process. The test results and the performance are shown in Section 5. Finally, Section 6 concludes the paper and discusses future directions. Special note: in this paper, the user, the client or the participant all refer to the same concept.

2. Related work

With the deepening of energy digital transformation, cross-border convergence, the sharing and integration of data and innovative applications are becoming increasingly widespread. According to the Global Internet Trends Report released in 2018, data sharing has become an inevitable trend in the development of the Internet and big data.

At present, in order to ensure data privacy, a lot of cryptography technologies are being used in the process of data sharing. This includes anonymous sharing, ciphertext searches, threshold access, provable security, permission security, etc. [11,12]. Furthermore, attribute encryption, blockchain technology and various data sharing schemes emerged in this period [13]. Among them, federated learning is a hot spot for current research, as it has recently attracted a lot of attention from the academic community. It was first proposed by Google in 2016 and was originally used to solve the problem of updating models locally by Android phone users. The goal of its design is to carry out efficient machine learning among multiple participants under the premise of guaranteeing data security. This technology is a type of distributed cryptographic technology in which all participants can share the underlying data. Its most important feature is to keep one's own data in the local area so that the original data of each participant will not be leaked during the process of cooperation training, and this technology has shown strong vitality and development prospects in more and more scenarios.

In 2019, FATE (Federated AI Technology Enabler) was developed at the AI department of Webank, and it was open-sourced. It is a high-performance, privacy-secure computing framework that provides a solution platform for common industrial applications [14]. As the concept of federated learning has become popular, the applications of federated learning [15–18] have been gradually developed. In 2022, Ait-Mlouk et al. proposed FEDQAS, a privacy-preserving machine reading system capable of leveraging large-scale private data without the need to pool those datasets in a central location. The proposed approach combines transformer models and federated learning technologies. The system was developed by using the FEDn framework and deployed as a proof-of-concept alliance initiative [19]. In the federated learning scenario, users do not need to share local data, but only upload the model-trained parameters. However, using model parameters as the interaction medium, there may be privacy leakage during the process of learning. Established federal learning works [20] show that private information may still be leaked when the model parameters of distributed users are uploaded piecewise with the model structure. Based on the model parameters uploaded by each participant, the original data owned by the local user can be inferred. Possibly more serious is that, when an attacker who may be an honest but curious server, a malicious client or a malicious third party directly intercepts model parameters uploaded by each client, they can further infer the user's private information, so the model parameters also need protection.

3. Preliminaries

To clarify the scheme proposed in this paper, some relevant theoretical knowledge needs to be introduced here.

3.1. Homomorphic encryption

The concept of HE was first proposed by Rivest et al in 1978 [21]. HE is a method that can process ciphertext information. It is an encryption technology that allows computing operations on ciphertext and generates encryption results. The calculation result obtained in the ciphertext is decrypted and matched with that in plaintext as if the same calculation operation has been performed on plaintext. The processing flow in encrypted and unencrypted states is shown in Figure 1.

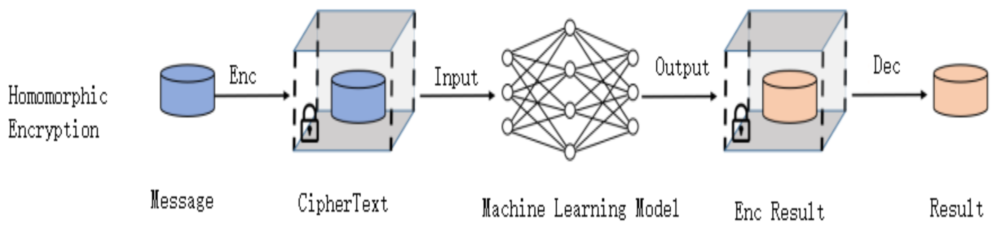


Figure 1. Process of HE.

As a method that can process ciphertext without decrypting ciphertext, HE is the most commonly used privacy protection mechanism nowadays. HE mechanism can compute the ciphertext without decrypting the ciphertext so that the computation party does not need to know the contents of the plaintext, but only needs to obtain the ciphertext, which is a good way to protect sensitive data and information while performing computation operations.

HE can efficiently process cryptographic information and achieve specific algebraic operations on the encrypted content. The HE cryptosystem is composed of quaternions, as shown in Eq (1).

$$H = \{Ho\_Key, Enc, Dec, Eval\} \tag{1}$$

where Ho_Key represents the key generation function, Enc represents the encryption function, Dec

represents the decryption function and $Eval$ represents the evaluation function.

A secure cryptosystem such as Eq (2) can be called a homomorphic operation. Using $Enc_{pub}(\cdot)$ represents the encryption function that uses the public key pub as the encryption key, *M* represents the plaintext space and *C* represents the ciphertext space.

$$\forall m_1, m_2 \in M, Enc_{pub}(m_1 \odot_M m_2) \leftarrow Enc_{pub}(m_1) \odot_C Enc_{pub}(m_2) \tag{2}$$

Here, $\odot_M$ and $\odot_C$ represent the operator on the plaintext space *M* and the ciphertext space *C*, respectively. Eq (2) shows that, for any two elements m_1 and m_2 in the plaintext space *M*, after performing the $\odot_M$ operating on them, the obtained result is encrypted; the result is the same as if m_1 and m_2 were encrypted first and then the operators were executed. The symbol "$\leftarrow$" indicates that the left-hand term is equal to or can be computed directly from the right-hand term without any intermediate decryption operation. To simplify the expression, we can use $[\![v]\!]$ to represent the result of HE for the plaintext *v*. The two basic operations of HE are defined below, namely, addition HE and multiplication HE.

Definition 1: Additive homomorphic operation. For any two elements *u* and *v* in the plaintext space, the encryption results are respectively $[\![u]\!]$ and $[\![v]\!]$, with Dec_{pri} indicating that the private key is used for decryption if Eq (3) is satisfied:

$$Dec_{pri}([\![u]\!] + [\![v]\!]) = Dec_{pri}([\![u+v]\!]) = u + v \tag{3}$$

Definition 2: Multiplicative homomorphic operation. For any two elements *u* and *v* in the plaintext space, the encryption results are respectively $[\![u]\!]$ and $[\![v]\!]$, with Dec_{pri} indicating that the private key is used for decryption if Eq (4) is satisfied:

$$Dec_{pri}([\![u]\!] \times [\![v]\!]) = Dec_{pri}([\![u \times v]\!]) = u \times v \tag{4}$$

3.2. Paillier algorithm

In the Paillier algorithm, the generation steps of the public-private key pair and the principle of encryption and decryption are as follows [22].

Key generation: First, randomly select two large prime numbers *a* and *b* (ensure that *a* and *b* are of equal length). Next, calculate $n = ab$ and $\lambda = lcm(a-1, b-1)$, where lcm is a function to find the least common multiple. Define $L(x) = \frac{x-1}{n}$, and then randomly select a positive integer g less than n^2 to satisfy Eq (5):

$$gcd(L(g^{\lambda} \bmod n^2), n) = 1, u = (L(g^{\lambda} \bmod n^2))^{-1} \bmod n \tag{5}$$

gcd is a function to find the maximum common divisor. By the above operation, we can get the public key (n, g) and private key (λ, u).

Encryption process: For any plaintext message m, choose any random number r satisfying $0 < r < n$; the ciphertext c is calculated by Eq (6):

$$c = g^m r^n \bmod n^2 \tag{6}$$

Decryption process: For the ciphertext c, the plaintext message m is obtained from Eq (7):

$$m = L(c^\lambda \bmod n^2) * u \bmod n \tag{7}$$

The Paillier algorithm is an implementation of an asymmetric encryption algorithm, which can operate on encrypted data under encryption and then decrypt the encrypted result. The obtained result is the same as the result of directly operating on the plaintext. However, the Paillier algorithm does not satisfy the multiplicative homomorphic operation. Although the Paillier algorithm is not fully HE, its computational efficiency is high, so it is widely used in the industry. In this paper, the Paillier algorithm is used as the simulation algorithm of HE.

4. Data security sharing scheme based on HE

4.1. Our construction

In this part, we build a federated learning model based on HE; it not only considers privacy security at the data level, but it also considers the security issues at the client level.

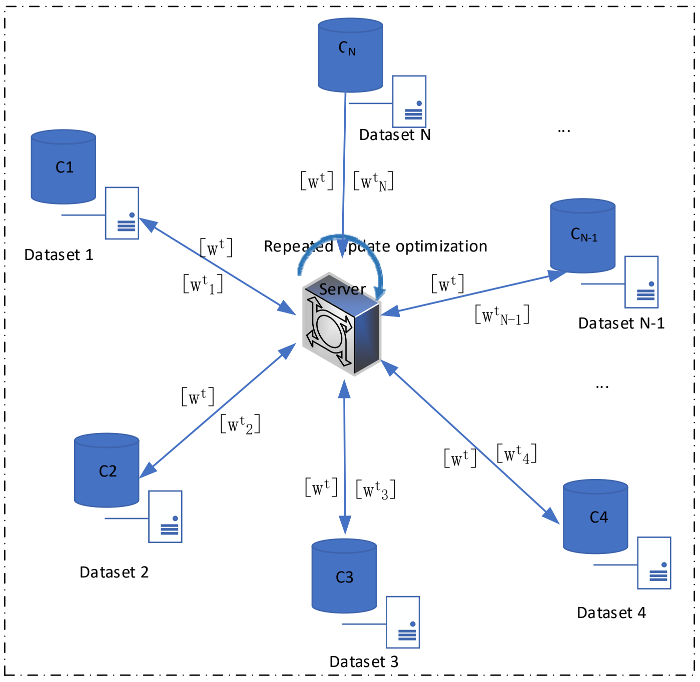


Figure 2. Architecture of data security sharing scheme.

This model not only ensures the privacy security of local data in each client, but it also ensures the information security between clients. The server receives the trained parameters of the local model from the client, but the server cannot determine which client uploaded it. It is also not possible to infer whether a client is participating in the current federated training. The architecture of the model is shown in Figure 2.

4.2. Data pre-processing

Here, we take the medical data as the sharing data. To realize data sharing under the premise of protecting data privacy, we used horizontal federated learning technology, so each participant has the same data features [23–26]. We used an open-source dataset from the Wisconsin Center for Scientific Research [27]. For the convenience of processing, we pre-processed the data features and extracted 30 main features. The information after feature extraction is shown in Table 1, and the 31st column represents the label data (with 1 for benign tumors and 0 for malignant tumors).

Table 1. Main features of medical data.

F1	F2	F3	F4	F5	F6	…	31_label
1.0961	-2.07151	1.268817	0.98351	1.567087	3.280628	…	0
1.828212	-0.35332	1.684473	1.90703	-0.82624	-0.48664	…	0
1.578499	0.455786	1.565126	1.557513	0.941382	1.052	…	0
-0.76823	0.253509	-0.59217	-0.76379	3.280667	3.399917	…	1
1.748758	-1.1508	1.775011	1.824624	0.280125	0.538866	…	0
…	…	…	…	…	…	…	…

Since some feature values are greater than 100 and some feature values are less than 1, the dataset is first normalized and pre-processed to reduce the dimensions and difference of each value. Data standardization mainly scales the value of each dimension according to a certain proportion so that it falls into a specific interval, allowing the feature value of different units or magnitudes to be weighted and compared.

4.3. Algorithms of data sharing scheme

4.3.1. Multi-party computation

Secure multi-party computation (MPC) is a branch of cryptography that involves multiple participants working together to perform collaborative computation [28]. Based on MPC for any function requirements can compute it without revealing information other than the output.

The whole protocol of MPC-based federal learning can be viewed as the process of computing the federal learning function f_{FL}. f_{FL} is defined as the composition of a round of functions $f_n, f_{n-1}, \dots f_1$, where is the jth iteration in the m-ary function.

For each iteration in federal learning, we define an m-ary function:

1) Define the m - 1 parties selected by the federal learning server (FLS) as MPC participants. There are m participants, including FLS itself.

2) Define the model weights and training parameters as the system parameters (*sysm*) of MPC.

3) Define the initial *view*$_i$. Initialize the view of the participant P_i by inputting the random numbers D_i, r_i and *sysm*.

4) Define , where $sout_i(i = 1,...,m - 1)$ denotes the output of P_i and the output of FLS is $sout_m$.

5) Define function m-ary: , where . In the case where *f* is a deterministic m-ary function, a secure MPC protocol Π is secure if there exists a probabilistic polynomial-time algorithm denoted as *S* for every $I \subseteq [m]$ if $S(I,(x_{i1},...,x_{it}),f_I(\bar{x}))$ is computationally indistinguishable from $view_I\Pi(\bar{x})$. In this scheme, we use MPC to protect the participants; it can ensure the security of the client level.

4.3.2. Calculation of the encryption loss function

In the federation learning model that uses the Paillier algorithm to protect the training parameters, the public and private keys are generally generated randomly on the server side. The public key is mainly for encrypting data, and the private key is for decrypting data. In machine learning models, a loss function is usually first defined, and then an optimization algorithm such as a stochastic gradient descent is used to find the minimum value of $L(\theta;x)$. The parameter θ^* that minimizes the value $L(\theta;x)$ is optimal. Taking logistic regression as an example, let the current set of *n* sample data points be $T = (x_1, y_1),(x_2, y_2),...,(x_n, y_n)$, and use the logarithmic loss function as its target loss function, as shown in Eq (8):

$$L = \frac{1}{n}\sum_{i=1}^{n} log(1 + e^{-y_i\theta^T x_i}) \tag{8}$$

The model parameters are updated by taking the partial derivative θ in the above Eq (8) and bringing the obtained gradient values into the gradient descent equation as shown in Eq (9):

$$\theta = \theta - lr * \frac{\partial L}{\partial \theta} \tag{9}$$

The above computation process is repeated until the value of the loss function $L(\theta;x)$ is no longer decreasing or the maximum number of iterations is reached; then, the iteration is stopped. The above computation process, including the parameters and data information, is computed in the explicit state, and there is a risk of data leakage in the federal learning scenario.

Federal learning based on HE requires that the parameters are solved in the encrypted state, i.e., the transmitted parameter θ is usually an encrypted value $[[\theta]]$; the loss function is shown in Eq (10).

$$L = \frac{1}{n}\sum_{i=1}^{n} log(1 + e^{-y_i[[\theta]]^T x_i}) \tag{10}$$

The calculation of the loss function involves exponential and logarithmic operations on the encrypted data, but the Paillier algorithm only supports addition homomorphism and scalar multiplication homomorphism; it does not support multiplication homomorphism and complex exponential and logarithmic operations. Therefore, it is not possible to solve the above Eq (10) in the encrypted state. Here, we use the Taylor loss function to approximate the original logarithmic loss function instead, i.e., by Taylor expansion of the original logarithmic loss function, the logarithmic loss function is approximated by polynomials, and after Taylor expansion, the loss function is transformed into only scalar multiplication and addition operations so that Paillier can be applied directly to the cryptographic solution. When $f(z)$ is the logarithmic loss function, the Taylor expansion expression for $f(z) = \log(1+e^{-z})$ at z is as shown in Eq (11).

$$log(1+e^{-z}) \approx log2 - \frac{1}{2}z + \frac{1}{8}z^2 + o(z^2) \tag{11}$$

Using the second-order polynomials to approximate the logarithmic loss function and substituting $z = y\theta^T x$ into Eq (11), we can get Eq (12):

$$log(1+e^{-y\theta^T x}) \approx log2 - \frac{1}{2}y\theta^T x + \frac{1}{8}(y\theta^T x)^2 \tag{12}$$

The encrypted gradient equation can be obtained from Eq (13):

$$\left[\left[\frac{\partial L}{\partial \theta}\right]\right] = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{4}[[\theta]]^T x_i - \frac{1}{2}y_i)x_i \tag{13}$$

4.3.3. Design of re-encryption algorithm

When using the Paillier algorithm for encryption and decryption operations, a large number of large prime power operations are involved, so intermediate results may be out of bounds and usually result in overflow errors. Therefore, we design the re-encryption algorithm to re-encrypt the data using the server-side key when the number of local training iterations reaches a certain number of rounds.

4.3.4. FedAvg and FedProx algorithms

Gradient descent is one of the most important optimization algorithms in machine learning. Since machine learning involves a large number of optimization problems, it is often difficult to directly use partial derivatives to obtain the optimal solution; then, the gradient descent method and its derivative models are needed to obtain the optimal solution iteratively. The computational process of gradient descent is to solve for the minimal value along the direction of gradient descent (or the maximal value along the direction of gradient ascent); FedAVg and FedProx gradient descent algorithms are used here.

The FedAvg algorithm is the most fundamental gradient aggregation method that only computes the gradient on the client side, the FedAvg method expects the client side to do more operations to get a better descent direction than the gradient [29,30]. The essence of the FedAvg idea is that the client uses a random gradient descent algorithm to get the weight parameters, and the server integrates each user's trained weights for averaging, as shown in Eq (14).

$$w_{t+1} \leftarrow \sum_{k=1}^{K}\frac{n_k}{n}w_{t+1}^{k} \tag{14}$$

The FedProx algorithm focuses on improving FedAvg performance from two directions: system heterogeneity and statistical heterogeneity. The local iterations of epochs performed by each client may not be guaranteed, so adding a proximal term to the optimization objective function of the client makes the optimization algorithm more stable and, ultimately, makes FedProx converge faster, even under statistical heterogeneity, as shown in Eq (15).

$$Q^{t+1} = Q^t + \lambda \sum_{i=1}^{m} (L_i^{t+1} - Q_i^t) \tag{15}$$

where Q^t denotes the global model parameters of the t^{th} round of aggregation, Q_i^{t+1} denotes the model of the i^{th} client after the $t+1$ round of local update and Q^{t+1} denotes the global model after the $t+1$ round of aggregation.

4.3.5. Logistic regression algorithm

Logistic regression is the most commonly used binary classification algorithm; it belongs to the family of generalized linear models and is widely used because of its simplicity and good results. A differentiable nonlinear function *f* is found to relate to the discrete label value *y*, and the predicted continuous value of linear regression is shown in Eq (16).

$$y = f(W^T X + b) \tag{16}$$

In logistic regression, the logistic function is generally used to act as this nonlinear mapping, and the logistic function is expressed in the form of Eq (17).

$$f(z) = \frac{1}{1+e^{-z}} \tag{17}$$

When using logistic regression for classification prediction, if the prediction value of linear regression $W^T X + \mathrm{b} \geq 0$, then it is judged to be a positive case and the output is 1; otherwise, it is judged to be a negative case and the output is 0.

4.4. Execution process

In this part, the server and the clients are based on the C/S communication mode. First, use the MPC protocol to ensure the participants and the server. Then, the server randomly generates a public-private key pair Key_{pub} and Key_{pri} based on the Paillier algorithm to encrypt the initialized model parameters W by using the public key to obtain $[w^t]$. Next, it sends the encrypted model parameters $[w^t]$ to the participating clients.

The clients begin the local training after getting the model parameters $[w^t]$, and then they send the new trained parameters $[w_i^t]$ back to the server. The server decrypts them by using the private key and uses a logistic regression algorithm to evaluate the model on the test set; it then generates the next round of new encrypted model parameters w^{t+1} according to the model aggregation algorithm. If it has not met the training requirements, repeat the process, i.e., encrypt, send, train, upload, decrypt, etc., until it meets the requirements of training.

5. Test results and discussion

In order to assess the performance of this scheme, a series of experiments was conducted. Here, we ignore communication time and assume that all participants are normally involved in the collaborative computation. The test was conducted using a Windows 10 operating system with an Intel (R) Core$^{\text{(TM)}}$ i5-6500 CPU 3.20 GHz and 16 GB of RAM. We used PyCharm as the integrated development environment. Python was used as the programming language, and many third-party libraries were installed, such as pytorch, socket, numpy, etc. The test dataset was from the Wisconsin Center for Scientific Research [28].

In the experiments, we considered the effects of the global training round, batch size and learning rate; at the same time, we compared the convergence speed of the FedAvg and FedProx algorithms. In this part, we mainly compared and tested two scenarios:

(i) Directly pass the parameters. (Here, we named it DP). In this federated learning scenario, it is just like the scheme FedQAS [19], without the need to pool those datasets in a central location. It just combines transformer models and federated learning technologies and allows intuitive participation and execution of local model training. It presents the architecture and implementation of the system, as well as provides a reference evaluation based on the SQUAD dataset, to showcase how it overcomes data privacy issues and enables knowledge sharing between alliance members in a federated learning setting.

(ii) Indirectly pass the parameters under HEmode (Here, it is named HEM). In this federated learning scenario, users do not need to share local data, but only upload the model-trained parameters. However, using model parameters as the interaction medium, there may be privacy leakage during the process of learning, so we chose the Paillier algorithm to realize the addition homomorphism to protect the training parameters.

5.1. Effect of global training round

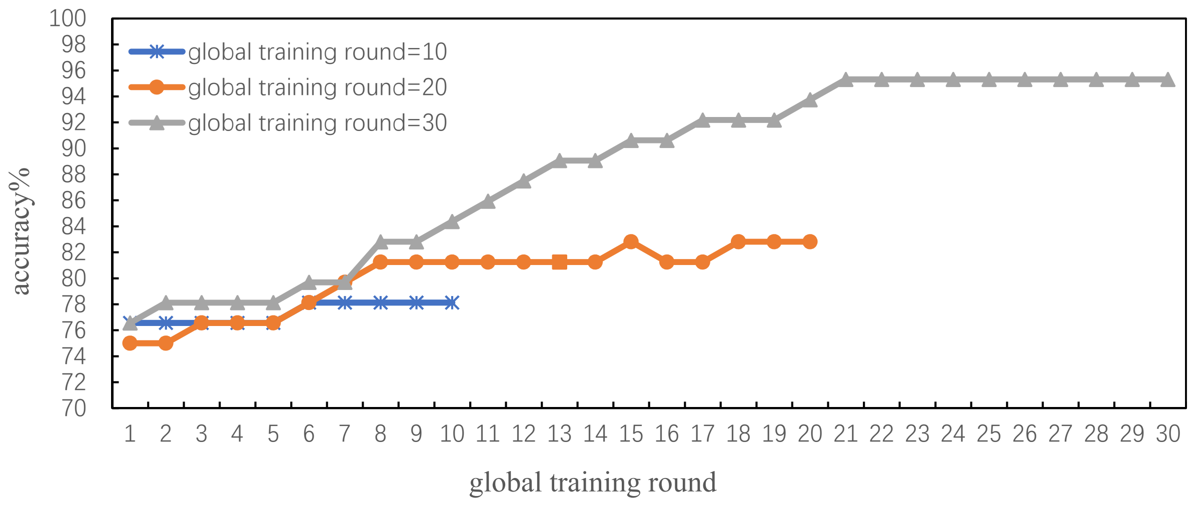
We set the learning rate to 0.01, the privacy budget to 0.5 and the batch size to 64. We compared the performance of the model when the global training round was 10, 20 and 30, respectively. The test condition is shown in Table 2.

Table 2. Test condition.

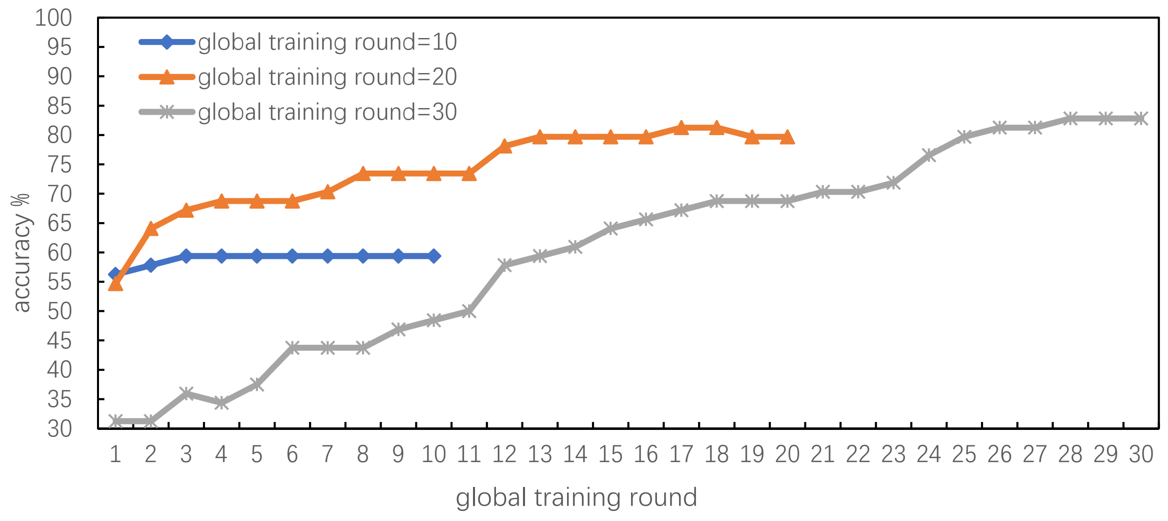
learning rate	privacy budget	batch size	global training round		
0.01	0.5	64	10	20	30

Based on the above test condition, we can get the test results shown in Figure 3.

Figure 3(a) is the result of directly passing the parameters (named in DP), and Figure 3(b) is the execution result of the parameters under HE protection (named in HE). From the result, we know that, when the number of global training rounds is small (no overfitting), the prediction accuracy increases with the number of training iterations, i.e., the model performance becomes better and better. And, when the number of training rounds is large, the change rate of the prediction accuracy slowly becomes smaller. In DP mode, the accuracy varies quickly with the number of training increasing, but slowly changes in HE mode.



(a) Effect of global training rounds in DP



(b) Effect of global training rounds in HEM

Figure 3. Effects of global training rounds in DP and HEM.

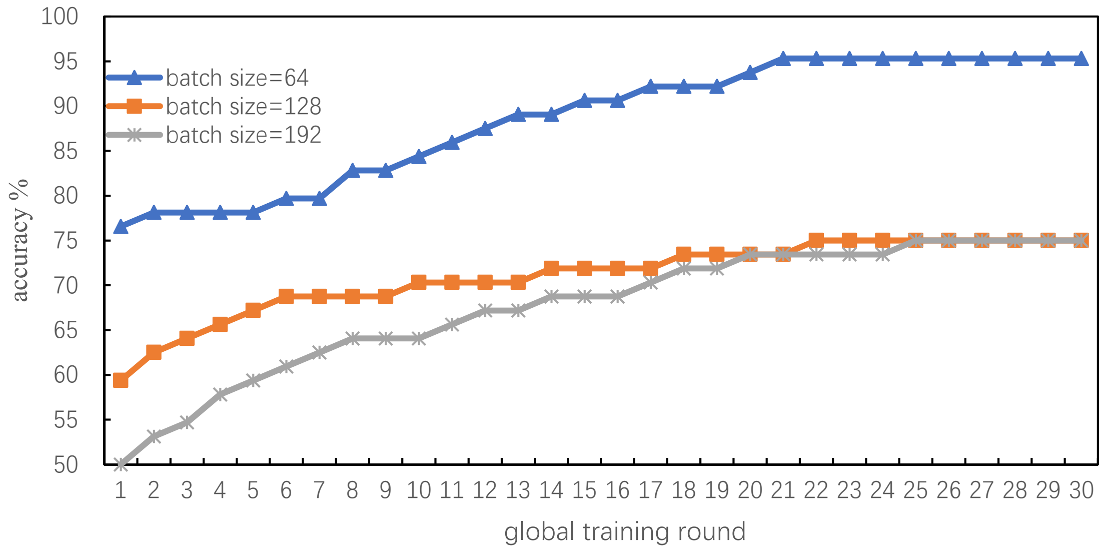
5.2. Effect of batch size

In this experiment, we set the test condition shown in Table 3.

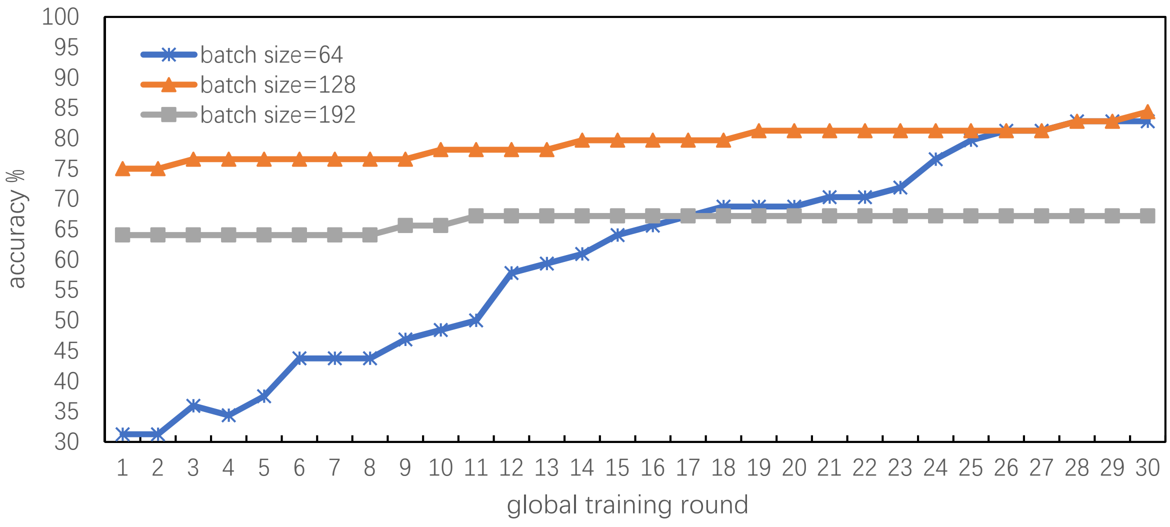
Table 3. Test condition.

learning rate	privacy budget	global training round	batch size		
0.01	0.5	30	64	128	192

The experimental results are shown in Figure 4. From the results, we can know that, when the global training number is small, the training effect of a smaller batch size is not as effective as that of a larger batch size, because the disadvantage of a smaller batch size is that the model is not guaranteed to converge to the global optimum. However, when the batch size is set too large, it leads to problems such as poor generalization ability. Usually, the larger the batch size, the more accurate the direction of gradient descent and the smaller the oscillation. But, when the batch size is too large, the local optimum will be generated. A smaller batch size brings more random factors and it is difficult to achieve convergence; in rare cases, it can obtain a better result.



(a) Effect of batch size in DP



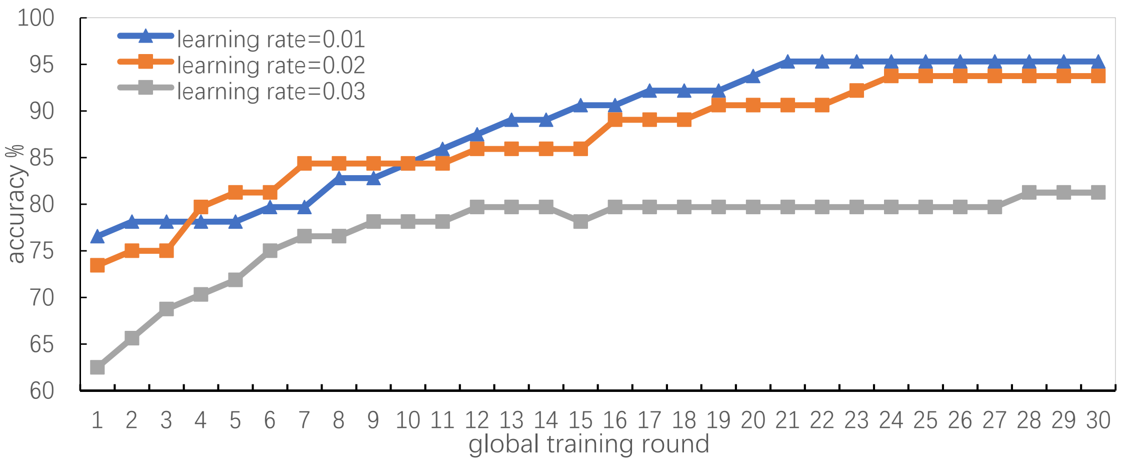
(b) Effect of batch size in HEM

Figure 4. Effects of batch size in DP and HEM.

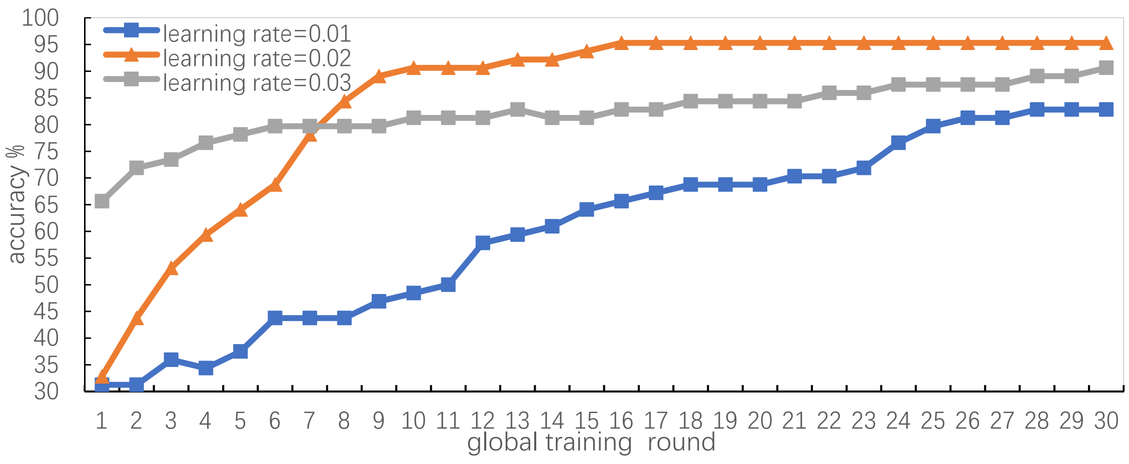
5.3. Effect of learning rate

For this experiment, the test condition is shown in Table 4. The learning rate represents the scale (step size) of the weight parameters in each update, which directly affects the update of the model weight parameters. When the learning rate is large, the gradient value is updated faster and it is easy to reach the convergence value and overfit the training model; the test results are shown in Figure 5. In Figure 5(a), when the learning rate is 0.03 and the global training round is 15, the prediction accuracy decreases instead of increasing. Likewise, in Figure 5(b), when the learning rate is 0.03 and the global training round is 14, the prediction accuracy is decreasing.

In most cases, when the learning rate is low, the gradient parameters are updated a little more slowly, which makes it easier to capture the optimal solution, just like when the learning rate is 0.01.



(a) Effect of learning rate in DP



(b) Effect of learning rate in HEM

Figure 5. Effects of learning rate in DP and HEM.

Table 4. Test condition.

batch size	privacy budget	the global training round	learning rate		
64	0.5	30	0.01	0.02	0.03

5.4. Comparison of convergence speed

In this experiment, we set the test parameters as below in Table 5. We compared the performance of two aggregation algorithms: FedAvg and FedProx. The test results are shown in Figure 6; from the results, we can know that the FedProx algorithm converges faster and achieves an accuracy of 67.5% after 10 rounds of global training. The FedAvg algorithm tends to deviate from the global optimum because of the client's data heterogeneity and local iterations, which affects the convergence. It leads to its model performance being inferior to the FedProx aggregation algorithm. The test results are shown in Figure 6.

Table 5. Test condition.

learning rate	privacy budget	the global training round	algorithm	
0.01	0.5	10	FedAvg	FedProx

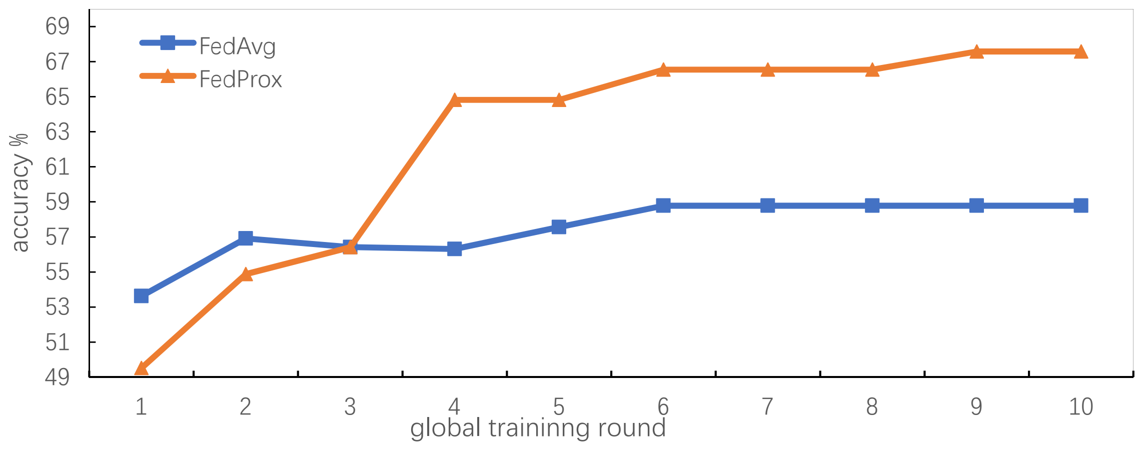


Figure 6. Comparison of convergence speed.

Two model aggregation algorithms, FedAvg and FedProx, are used in this scheme. Although the FedProx has improved the prediction effect compared with the FedAvg, the change in learning rate and batch size settings can easily lead to the occurrence of overfitting. Therefore, FedProx is more suitable for heterogeneous networks. In addition, since FedProx, HE algorithm needs to pass the gradient difference of model parameters, which is approximately equal to 0, so FedProx algorithm is not suitable for the federal learning model based on homomorphic encryption.

5.5. Analysis and conclusions

Summarizing the above experimental results and considering all the factors, the ideal parameter configuration is as follows: privacy budget = 0.5, learning rate = 0.02 and batch size = 128, and it is an optimal configuration.

For this experiment, the parameter configuration is shown in Table 6. This test compared two schemes, where one is based on the idea of FedQAS [19], and the other is the scheme of this paper (our scheme is named HE). The test results are shown in Figure 7.

Table 6. Test condition.

learning rate	batch size	privacy budget	global training round			algorithm
0.02	128	0.5	10	20	30	FedProx

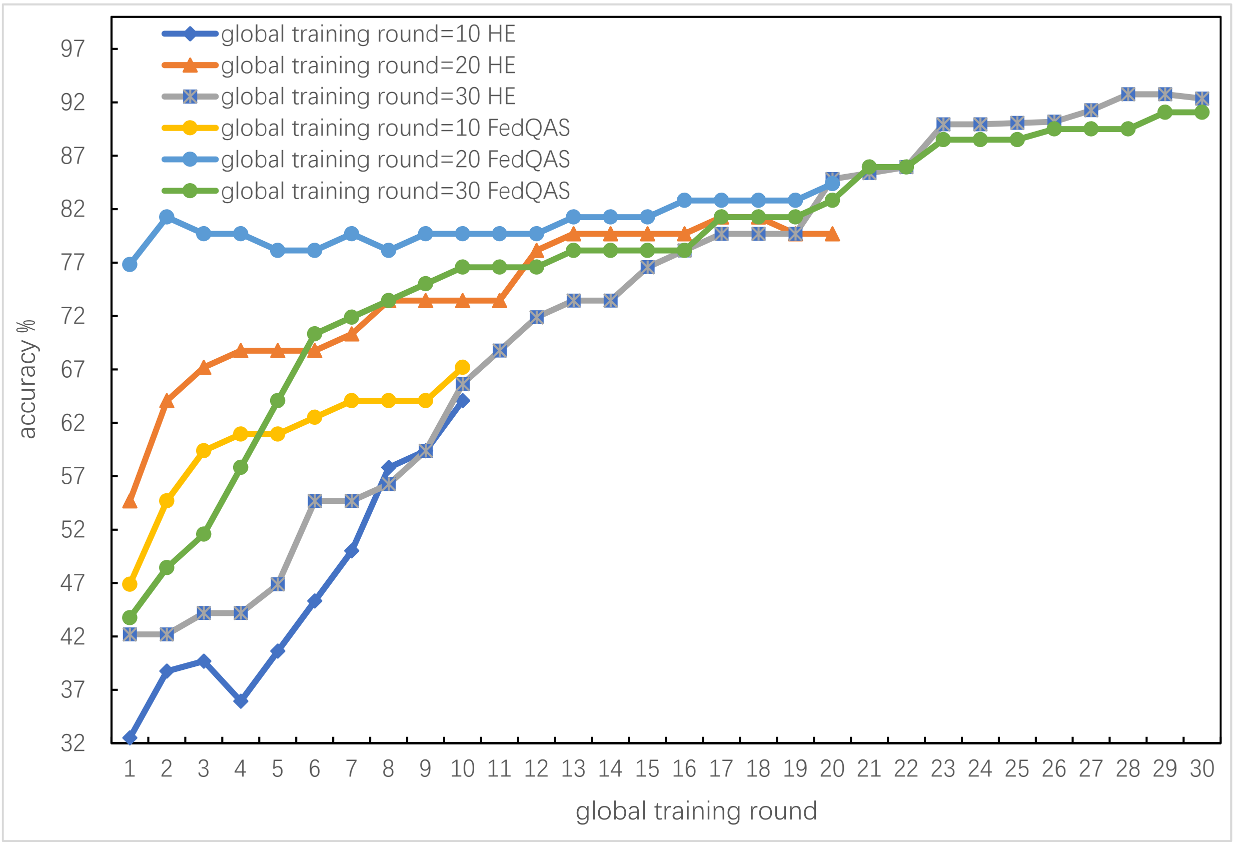


Figure 7. Comparison of accuracy of two schemes.

After many rounds of testing, for this specific test dataset, when the global training round is 30, in the scheme of HE, the lowest prediction accuracy rate is 89.528% and the highest accuracy rate is 92.352%. And, in the case of the FedQAS scheme, the lowest prediction accuracy rate is 88.938% and the highest accuracy rate is 91.063. In the case of the HE scheme, when the global training round is low, its accuracy has a slight fluctuation, but, with the increase of global training rounds, it becomes gradually steady. From the test results, we know that this kind of data security sharing scheme based on HE has good performance.

The introduction of HE increases the intensity of privacy protection, and the experimental results fully demonstrate the validity of the scheme. Therefore, the introduction of HE to federated learning not only improves the prediction accuracy, but it also effectively prevents privacy leakage.

6. Conclusions

In this study, to improve the accuracy of the prediction and diagnosis of diseases in the medical field, we designed and implemented a data security sharing model based on HE with a distributed federated learning architecture. It not only considers privacy security at the data level, but it also considers the security issues at the client level. At the data level, the original data are kept locally, which significantly reduces the risk of data leakage, to prevent parameter leakage in the federated learning process; the Paillier algorithm was introduced to protect the model parameters in communication. At the client level, the server uses an MPC protocol to select clients who will participate in the training and has no way of knowing which one is involved in the training between clients, which protects the privacy of the clients.

The system is based on the C/S communication mode; the server is mainly responsible for issuing training parameters, aggregating the local model parameters from the clients and predicting the joint diagnostic results. The client mainly uses the stochastic gradient descent algorithm for gradient trimming, updating and transmitting the trained model parameters back to the server.

From the simulation test results, we can know that, when the global training round is 30, the lowest prediction accuracy rate is 89.528% and the highest accuracy rate is up to 92.352% for this specific test dataset. Therefore, the introduction of HE to federated learning can effectively prevent privacy leakage. This scheme realizes data security sharing, completes the accurate prediction of diseases and has a good effect.

Acknowledgments

This work was supported by funding from the Nanjing Institute of Technology (No. KE21-451) and Innovation and Entrepreneurship Training Program for College Students in Jiangsu Province (No. 202211276101Y).

Conflict of interest

The authors declare that there is no conflict of interest.

References

1. J. Scheibner, J. L. Raisaro, J. R. Troncoso-Pastoriza, M. Ienca, J. Fellay, E. Vayena, et al., Revolutionizing medical data sharing using advanced privacy-enhancing technologies: technical, legal, and ethical synthesis, *J. Med. Internet Res.*, **23** (2021), e25120. https://doi.org/10.2196/25120
2. S. R. Oh, Y. D. Seo, E. Lee, Y. G. Kim, A comprehensive survey on security and privacy for electronic health data, *Int. J. Environ. Res. Public Health*, **18** (2021), 9668. https://doi.org/10.3390/ijerph18189668
3. C. Thapa, S. Camtepe, Precision health data: Requirements, challenges and existing techniques for data security and privacy, *Comput. Biol. Med.*, **12** (2021), 104130. https://doi.org/10.1016/j.compbiomed.2020.104130

4. D. Froelicher, J. R. Troncoso-Pastoriza, J. L. Raisaro, M. A. Cuendet, J. S. Sousa, H. Cho, et al., Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption, *Nat. Commun.*, **12** (2021), 5910. https://doi.org/10.1038/s41467-021-25972-y
5. N. Peng, H. Wang, *Federated Learning Technology and Practice*, Electronic Industry Press, Beijing, 2021.
6. E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, preprint, arXiv:1807.00459. https://doi.org/10.48550/arXiv.1807.00459
7. A. N. Bhagoji, S. Chakraborty, P. Mittal, S. Calo, Analyzing federated learning through an adversarial lens, in *Proceedings of the 36th International Conference on Machine Learning*, (2019), 634–643.
8. L. Chen, H. Wang, Z. Charles, D. Papailiopoulos, DRACO: byzantine-resilient distributed training via redundant gradients, preprint, arXiv:1803.09877. https://doi.org/10.48550/arXiv.1803.09877
9. C. Fung, C. J. M. Yoon, I. Beschastnikh, Mitigating sybils in federated learning poisoning, preprint, arXiv:1808.04866. https://doi.org/10.48550/arXiv.1808.04866
10. L. T. Phong, Y. Aono, T. Hayashi, L. Wang, S. Moriai, Privacy-preserving deep learning via additively homomorphic encryption, *IEEE Trans. Inf. Forensics Secur.*, **13** (2017), 1333–1345. https://doi.org/10.1109/TIFS.2017.2787987
11. B. Qiu, H. Xiao, A. Chronopoulos, D. Zhou, S. Ouyang, Optimal access scheme for security provisioning of C-V2X computation offloading network with imperfect CSI, *IEEE Access*, **8** (2020), 9680–9691. https://doi.org/10.1109/ACCESS.2020.2964795
12. J. Yu, R. Hao, Comments on SEPDP: secure and efficient privacy preserving provable data possession in cloud storage, *IEEE Trans. Serv. Comput.*, **14** (2021), 2090–2092. https://doi.org/10.1109/TSC.2019.2912379
13. C. Lin, D. He, X. Huang, X. Xie, K. R. Choo, PPChain: A privacy-preserving permissioned blockchain architecture for cryptocurrency and other regulated applications, *IEEE Syst. J.*, **15** (2021), 4367–4378. https://doi.org/10.1109/JSYST.2020.3019923
14. Q. Yang, A. Huang, Y. Liu, T. Chen, *Practicing Federated Learning*, Electronic Industry Press, Beijing, (2021), 26–30.
15. A. Hard, K. Rao, R. Mathews, F. Beaufays, D. Ramage, Federated learning for mobile keyboard prediction, preprint, arXiv:1811.03604. https://doi.org/10.48550/arXiv.1811.03604
16. D. Shi, L. Li, R. Chen, P. Prakash, M. Pan, Y. Fang, Towards energy efficient federated learning over 5G+ mobile devices, preprint, arXiv: 2101.04866. https://doi.org/10.48550/arXiv.2101.04866
17. D. Jiang, Y. Tong, Y. Song, X. Wu, W. Zhao, J. Peng, et al., Industrial federated topic modeling, *ACM Trans. Intell. Syst. Technol.*, **12** (2021), 1–22. https://doi.org/10.1145/3418283
18. Y. Luo, H. Zhou, W. Tu, Y. Chen, W. Dai, Q. Yang, Network on network for tabular data classification in real-world applications, in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, (2020), 2317–2326. https://doi.org/10.1145/3397271.3401437
19. A. AitMlouk, S. Alawadi, S. Toor, A. Hellander, FedQAS: privacy-aware machine reading comprehension with federated learning, *Appl. Sci.*, **12** (2022), 3130. https://doi.org/10.3390/app12063130

20. W. Zhang, X. Li, Federated transfer learning for intelligent fault diagnostics using deep adversarial networks with data privacy, *IEEE/ASME Trans. Mechatron.*, **27** (2022), 430–439. https://doi.org/10.1109/TMECH.2021.3065522
21. R. L. Rivest, L. Adleman, M. L. Dertouzos, On data banks and privacy homeomorphisms, *Found. Secure Comput.*, (1978), 169–179.
22. L. M. Surhone, M. T. Timpledon, S. F. Marseken, *Paillier Cryptosystem*, Betascript Publishing, 2010.
23. A. Alsirhani, M. Ezz, A. M. Mostafa, Advanced authentication mechanisms for identity and access management in cloud computing, *Comput. Syst. Sci. Eng.*, **43** (2022), 967–984. https://doi.org/10.32604/csse.2022.024854
24. M. Ragab, H. A. Abdushkour, A. F. Nahhas, W. H. Aljedaibi, Deer hunting optimization with deep learning model for lung cancer classification, *CMC-Comput. Mater. Continua*, **73** (2022), 533–546. https://doi.org/10.32604/cmc.2022.028856
25. X. Zhang, W. Zhang, W. Sun, X. Sun. S. K. Jha, A robust 3-D medical watermarking based on wavelet transform for data protection, *Comput. Syst. Sci. Eng.*, **41** (2022), 1043–1056. https://doi.org/10.32604/csse.2022.022305
26. Y. Y. Ghadi, I. Akhter, S. A. Alsuhibany, T. A. Shloul, A. Jalal, K. Kim, Multiple events detection using context-intelligence features, *Intell. Autom. Soft Comput.*, **34** (2022), 1455–1471. https://doi.org/10.32604/iasc.2022.025013
27. *UCI*, Breast cancer wisconsin (original) data set, 2021. Available from: http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29.
28. C. Zhao, S. Zhao, M. Zhao, Z. Chen, C. Gao, H. Li, et al., Secure multi-party computation: theory, practice and applications, *Inf. Sci.*, **476** (2019), 357–372. https://doi.org/10.1016/j.ins.2018.10.024
29. C. Luo, X. Chen, C. Ma, S. Zhang, Improved federated average algorithm based on tomographic analysis, *Comput. Sci.*, **48** (2021), 32–40. https://doi.org/10.11896/jsjkx.201000093
30. X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of FedAvg on Non-IID data, preprint, arXiv:1907.02189. https://doi.org/10.48550/arXiv.1907.02189