TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 08/24 pp400–414
DOI: 10.26599/TST.2023.9010007
Volume 29, Number 2, April 2024

# Federated Learning Security and Privacy-Preserving Algorithm and Experiments Research Under Internet of Things Critical Infrastructure

Nasir Ahmad Jalali and Hongsong Chen*

**Abstract:** The widespread use of the Internet of Things (IoTs) and the rapid development of artificial intelligence technologies have enabled applications to cross commercial and industrial band settings. Within such systems, all participants related to commercial and industrial systems must communicate and generate data. However, due to the small storage capacities of IoT devices, they are required to store and transfer the generated data to third-party entity called "cloud", which creates one single point to store their data. However, as the number of participants increases, the size of generated data also increases. Therefore, such a centralized mechanism for data collection and exchange between participants is likely to face numerous challenges in terms of security, privacy, and performance. To address these challenges, Federated Learning (FL) has been proposed as a reasonable decentralizing approach, in which clients no longer need to transfer and store real data in the central server. Instead, they only share updated training models that are trained over their private datasets. At the same time, FL enables clients in distributed systems to share their machine learning models collaboratively without their training data, thus reducing data privacy and security challeges. However, slow model training and the execution of additional unnecessary communication rounds may hinder FL applications from operating properly in a distributed system. Furthermore, these unnecessary communication rounds make the system vulnerable to security and privacy issues, because irrelevant model updates are sent between clients and servers. Thus, in this work, we propose an algorithm for fully homomorphic encryption called Cheon-Kim-Kim-Song (CKKS) to encrypt model parameters for their local information privacy-preserving function. The proposed solution uses the impetus term to speed up model convergence during the model training process. Furthermore, it establishes a secure communication channel between IoT devices and the server. We also use a lightweight secure transport protocol to mitigate the communication overhead, thereby improving communication security and efficiency with low communication latency between client and server.

**Key words:** Federated Learning (FL); Internet of Things (IoTs); lightweight Transport Layer Security (iTLS); Cheon-Kim-Kim-Song (CKKS)

## 1 Introduction

With the objective of increasing human life quality, Internet of Things (IoTs) enabled smart systems have become increasingly popular in recent years. The smart system concept has been enabled by IoT devices, such as smartphones and Artificial Intelligence (AI), as well as related storage and computation mechanisms, including cloud, edge computing, and big data analytics. However, as the increase of nodes in such smart systems generates massive amounts of data, due to the limited capacity of IoT devices, all

• Nasir Ahmad Jalali and Hongsong Chen are with Department of Computer Science, University of Science and Technology Beijing (USTB), Beijing 100083, China. E-mail: nasir.zehand@gmail.com; chenhs@ustb.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2022-07-20; revised: 2022-12-07; accepted: 2023-02-13

generated data must be transferred and stored in third-party (cloud) servers[1]. Highly mature techniques are required for the computation and processing of such vast amounts of data. Machine Learning (ML) techniques are highly successful techniques whose three main characteristics (big data availability, newer learning model computational power, and the evolution of deep learning model) have contributed significantly to their extensive usage and application[2]. Despite the fact that ML has shown success in big and huge data analysis and processing, most domains are not willing to deploy them in the real world. One study[3] reports that, in today's competitive business environments, data holders are not willing to share their data with a centralized system due to security, privacy, and performance concerns. This is because they create one single point, so if the central server suffers from an attack, all stored data will be exploited. Some other reasons cited by the data holders include the following:

- Concerns about the user's data privacy;
- Confidentiality issues arising from the transfer of data from the clients to the central server for learning;
- Failure to use significant amounts of data and computing resources in the edge network for effective learning improvement.

Other confidentiality and security concerns regarding ML involve the fact that data holders cannot transfer their datasets to a central location for computation and processing while the system is being used by some companies[4, 5]. As shown in Fig. 1, all data are collected and stored in a central server, and the model is trained by the central dataset.

As mentioned above, all users are concerned about their data privacy and confidentiality when transferring data to a third-party cloud server. When clients send their data to the central server, it may not be able to achieve fast computation. As clients need to receive their responses immediately, they are often unable to use those resources effectively for learning improvements. To overcome these challenges, Federated Learning (FL), which is based on the concept of ML, has been proposed as the best and most affordable solution[6]. Although recent studies have shown that FL has numerous advantages, it also faces security, privacy, efficiency, and communication cost issues. For example, private data may be leaked while transferring gradients to the central server in the FL system due to untrustworthy model generation or information about previous data[7]. Thus, in the FL system, encrypting and protecting data from leaking via the Homomorphic Encryption (HE) scheme is a reasonable choice. Thus far, researchers have worked on preserving information privacy efficiently in the FL system based on their expectations. However, they did not consider the improvement of convergence performance in the FL system[8]. Furthermore, the key elements of the FL system, namely, efficient performance in terms of model accuracy, model convergence rate, and communication cost with low latency, must be considered to ensure the security and privacy of the data resources of distributed participants.


**Fig. 1 Machine learning model with centralized dataset.**

## 2 Related Work

The use of FL in distributed learning systems has attracted increasing attention. However, certain issues, such as security and privacy leakage concerns, the requirement for high performance, low communication overhead, and low latency to communicate and respond to clients on time in a secure environment, have yet to be fully resolved. Recent attacks reveal that personal private and sensitive data may be leaked when the model is transferred between clients and the central server. Although differential privacy is used as a security and privacy mechanism, it has also led to increased model training costs and lower model performance quality[9]. Secure communication is very important for IoT devices in FL. However, according to studies in Refs. [10] and [11], the current approaches using Transport Layer Security (TLS) protocol and Datagram TLS (DTLS) protocols are challenged by heavy communication overhead, high latency, low security, and transport protocol exchanges that require many roundtrip messages between a client and server. Different types of HE, such as secure multiparty computation, have shown to be robust to users dropping out; However, these can also lead to large communication and

computational overhead[12]. To resolve this problem, the Cheom-Kim-Kim-Song (CKKS) scheme is proposed, which has an excellent encryption speed than other encryption algorithms, such as RSA. Our scheme uses both the CKKS algorithm and the lightweight Transport Layer Security (iTLS) protocol to encrypt the model's parameters to preserve privacy, speed up model convergence, decrease communication and computation overhead, achieve low latency, and reduce and limit the number of unnecessary roundtrips of communication messages.

### 2.1 Motivation and problem statement

FL is a distributing ML technique that enhances user data privacy by retaining user data locally. However, it is prone to certain security and performance challenges related to distribution ML systems. Although many studies have been conducted regarding privacy and security, they have been unable to completely eliminate FL-related performance, privacy, accuracy, and security challenges arising during real-life applications. Thus, motivated by the above metioned challenges, we aim to provide a secure and high-performance environment for distributed learning machines. The potential contributions of this study are two-fold. First, we propose a full homomorphic algorithm, i.e., CKKS, to encrypt model parameters for their local information privacy preservation and faster model convergence and computation during model training. Second, we use a lightweight protocol to promote efficient and secure communication overhead with low latency.

### 2.2 Paper structure

The remainder of the paper is divided into different sections. Section 2 presents the related works, while Section 3 explores the landscape of the FL model, mainly focusing on security and privacy preservation in the FL system. Section 4 addresses the challenges in FL under the IoT environment, while Section 5 covers the experimental part, including the experimental environment and configuration. Finally, Section 6 presents the conclusion for the entire paper.

## 3 Landscape About FL Model

The research community has developed and proposed the FL framework to overcome the above mentioned limitations and problems related to well-trained central ML, such as data privacy, confidentiality, high-speed communication, and high computation power. FL is widely considered the best and most affordable solution for IoT device communication[13]. FL is also a type of ML technique that can solve the above mentioned challenges compared with the practice of centralizing all data and training the model on a central server. This is because the FL model can train decentralized data without transferring and collecting all data to a central server. As shown in Fig. 2, each client trains the model using local data, after which it sends locally trained models to the central server. When the server receives the local models, it aggregates them, builds one global model, and sends it back to all participants to update the information. The participants circulate these activities without transferring actual local data.

The process described below for FL model updating is iterative, in which each iteration improves the central ML model. The process is organized into three major steps as follows:

**Step 1:** Model selection. The global model pre-trains the model with all initial parameters and shares the output with all clients included in the FL environment.

**Step 2:** Local model training. When a global model is shared with clients, then each node trains the model over its local dataset.

**Step 3:** Aggregate local model. A model is updated when it is trained locally on the client side. Then, the new version of the model is shared with the central point. The server updates and aggregates the global model with new parameters and returns it to the clients to start the iteration that will update the information.

The above steps show that the FL framework trains the ML model on several devices and does not need to transfer real data to the central server (cloud). Furthermore, only the training model will share the data between clients and servers by circulating Steps 2 and 3 to update the model. Figure 3 shows the details of how

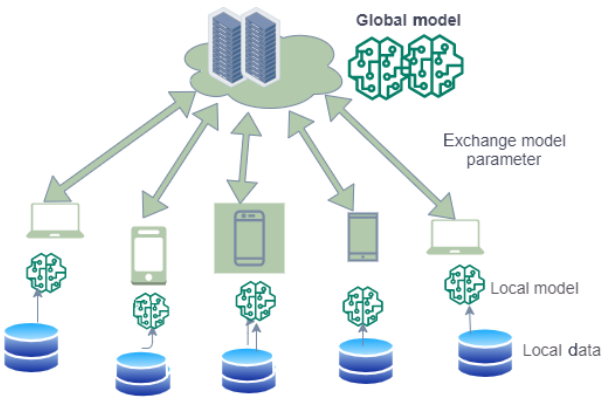


**Fig. 2 Federated learning system.**

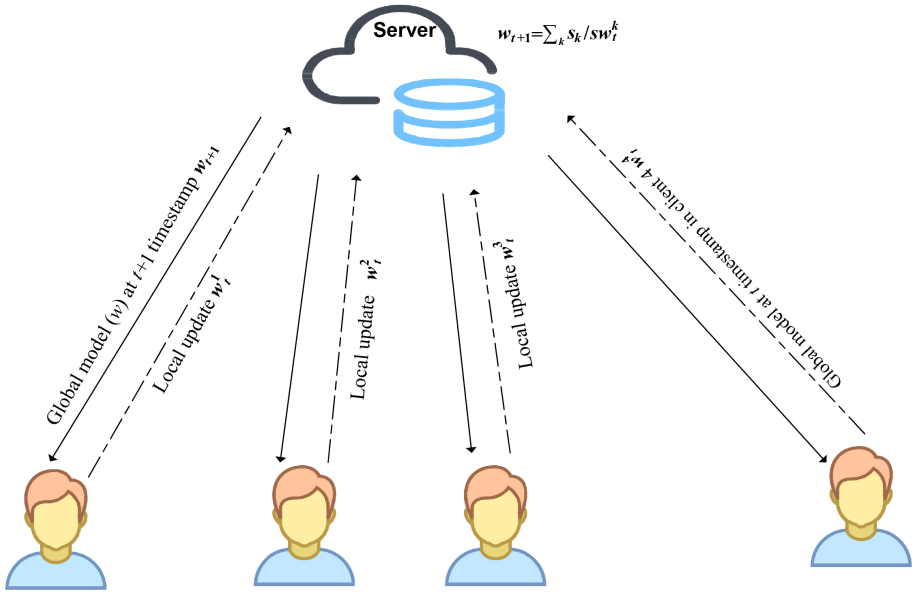


**Fig. 3 Federated learning model updating process.**

the model is updated in the FL system.

As shown in Fig. 3, the process begins with global model $w_t$ initialization, and the model is sent to the clients. Upon receiving the model, each client "$k$" trains the model with its local data, where $s_k$ is the number of trained samples held by clients, and $s$ is the total number of samples trained by clients. Then the updated model $w_t^k$ returns to the server, and the server aggregates all received models to update the global model $w_{t+1}$[14]. In this way, data privacy is maintained locally within the clients' environments. All IoT devices in the FL framework are responsible for performing the computation processes to train the model and preserve privacy efficiently[15].

### 3.1 Security and privacy preservation in FL

In today's data-driven world, most services and applications, such as educational, healthcare, medical, smart city, and financial applications, heavily rely on AI technology, such as IoT devices. Different applications can generate various data types that work on a complex ML model; therefore, users must pay attention to data protection and privacy[16]. However, AI technology has not yet reached its full potential, and applications related to AI and ML continue to face many challenges pertaining to centralized storage and computation[17]. AI technologies generate data, especially personal data, and store them in a data silo due to their limited storage capacity—be they related to an end-user or service provider. While most ML algorithms operate in a centralized manner, they need to collect data into the central server for integrating, aggregating, and processing training data[18]. In a conventional ML algorithm, collecting large amounts of data into a cloud or central server can lead to a single point of failure and result in security risks, such as a data breach. This centralized data processing and management can lead to limitations in transparency and provenance, leading to distrust between end-users and difficult deployment[19]. FL is the best and most affordable solution to overcome such challenges. FL is a technique used by ML algorithms in a decentralized collaborative learning model, where the algorithm is executed on various separated datasets. There is no need to collect all data from IoT devices to a centralized server for operation. Instead, the participants only exchange updated models with a central coordinated server[20, 21]. The major advantage of FL over traditional cloud-centric ML is that the former ensures the privacy of data processed locally on the client side and is required to exchange only the model parameters. Some privacy-preserving and security techniques strengthen this transmission and aggregation between central coordinated servers and clients, resulting in enhanced data privacy and security[22, 23]. ML technology is used to design and build intelligent systems, such as IoT devices, which automatically learn and process their activities based on mathematical algorithms and models (model training process) using a sample set (training data) without direct human instructions[18].

### 3.2 Communication cost and model accuracy in FL

As mentioned in Section 2, FL is a type of ML approach to training a global model with high-quality over distributed datasets coming from numerous IoT clients. These IoT clients might have unreliable and slow network connections; thus, each client may train, update, and compute the current model independently based on its local dataset. Furthermore, local clients update the aggregates to compute a new model and communicate or return to the central server, thus highlighting the importance of communication cost and efficiency[24]. In the FL learning system, all participants maintain iteration rounds with the central server to achieve active connections and obtain the desired accuracy. However, because of unnecessary iterations, model parameters may still be leaked even in favorable environments. In comparison, ML contains and processes a huge number of parameters, resulting in high communication costs and delays while the participants upload the updates[25].

## 4 Addressing Challenges in FL Application Under IoT Environment

A previous study[26] predicted that the number of IoT devices would reach 75 billion worldwide. Such unbelievable development will cause enormous growth in the distributed data generated by these IoT devices. Such data are relayed on centralized storage where all data are transferred from clients and collectively stored on the server. Therefore, having a central point can lead to vulnerabilities, such as compromising data security and privacy. In recent years, Google has introduced FL as a reasonable solution to reduce security and privacy challenges. FL enables all distributed clients to collaboratively learn a shared ML model without transferring a client's local data to the central server[27]. However, FL still faces security and privacy risks because the leaking of information related to locally trained datasets from the model's parameters is still possible. Thus, it is important to implement specific security and privacy algorithms, such as HE algorithms, to enhance security and reduce privacy risks.

### 4.1 HE

This technique is used in ML algorithms to achieve privacy and preservation. HE enables an application to perform a computation on encrypted data without decrypting the cipher text. This type of encryption allows the delivery of a cipher text $C(M)$ of a plaintext message $M$ and performs the computation on cipher text $C(f(M))$ function on a plaintext message $M$ without decrypting the message $M$[18, 28]. This encryption process proceeds in three steps:

**Step 1:** Key generation. This step generates both the secret key (sk) and public key (pk).

**Step 2:** Encryption. The plaintext message performs encryption with the public key to produce ciphertext.

**Step 3:** Decryption. The encrypted message is decrypted using a secret key to yield a plain text message.

The steps of the encryption process between the client and server in the cloud are shown in Fig. 4. In Step 1, the client generates a plaintext message $M$ with the encryption key sent to the cloud, and then stores the encrypted data along with the key in the central database.

Whenever the client needs to perform an operation, it must send a request to the service provider (cloud), after which the service provider transfers the client's request to the processing server. Then, the server performs the operation $C(f(M))$ as per request, and returns the result to the client as a response step. Finally, the client decrypts the result from the service provider (cloud) with a secret key. HE is classified into three major categories based on its operation on data.

(1) Partial HE: It allows either addition or multiplication operations but only on encrypted data.

(2) Somewhat HE: This will perform more than one operation on encrypted data, but not with a limited scale.

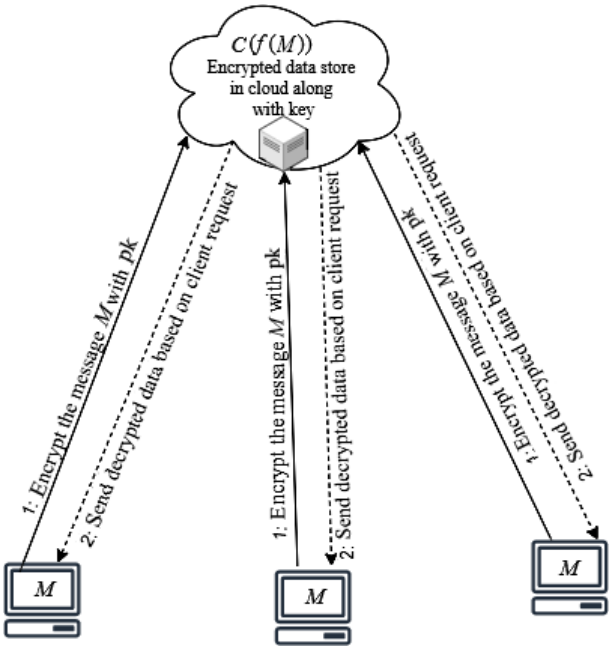


**Fig. 4 Basic structure of the homomorphic encryption method.**

(3) Full HE (FHE): This will allow many addition and multiplication to perform the operation on encrypted data.

#### 4.1.1 FHE

FHE is a key technological algorithm for secure computation that allows a third party to perform arbitrary functions over encrypted data without knowing the input and the results of the computation[29]. In the FHE method, the composition polynomial is used to split high-degree polynomials into different low-degree polynomials. The concept of this procedure is to encrypt a plaintext message with a large-ring-packed cipher text. Then, this message will recover as a simple linear function on the plaintexts encrypted in the smaller-ring cipher text. As such, transferring a smaller cipher text instead of a large cipher text will improve process efficiency[30]. We use the CKKS, which is an updated and new type of HE for data security and privacy.

#### 4.1.2 CKKS

As mentioned in Section 3.3.2, FHE allows computation over encrypted data. Researchers have argued that the data encryption/decryption process with FHE is computationally expensive[31]. The CKKS scheme is a newly proposed FHE encryption scheme that can rapidly perform encryption and computation compared with other HE schemes, such as Rivest-Shamir-Adleman (RSA)[32], which is depicted in Fig. 5. The reasonable solution to preserve data privacy and improve the model training efficiency of FL is the CKKS scheme or HE for Arithmetic Approximate Numbers (HEAAN)[33].

When data are exchanged between two parties in a communication system, data security (confidentiality and integrity) should be guaranteed between them. This is because security assurance in IoT devices is more important in achieving privacy and quality of services[34]. The full homomorphic CKKS scheme operation features several steps. First, the computer/client-generated message $M$ is encoded into plain text $p(X)$, after which the message is encrypted with public keys $c_0$ and $c_1$, which show various encrypted messages, for example, we have two messages and they are encrypted into ciphertext $C = c_0 + c_1$, so it changed from plain text to cipher. Then, the server stores the message to perform a homomorphic operation $f()$ over encrypted data. Based on the client's request, the server returns the encrypted data to the client and performs additional steps (reverse steps of encrypts) to decrypt the message or data. Next, the client decrypts the data with its secret key to plain text $p = f()$, after which it decodes and converts it to message $f(M)$ or data. The main concept behind HE is to have HE properties, such as encoder, encryptor, decryptor, and decoder, enabling participants to decrypt and decode operations correctly on cipher text and provide the output after the operations are implemented. The proposed CKKS scheme works with polynomials because it has good intercommunication between security and efficiency than other standard computations. There are many other protocols used to secure communication channels while they exchange data; however, they are unable to decrease communication overhead with low latency and cannot control iteration rounds between a client and server. Lightweight security protocol and symmetric payload encryption are used to minimize communication overhead with low latency and perform computation, and save communication capacity[35].

### 4.2 Lightweight secure transport protocol

The IoT is a growing technology that connects cyberworlds with physical devices. We can find IoT technologies and applications in different areas of our daily life, such as health, education, transportation, and so on. All IoT-based systems have benefits, but they also face some communication and performance challenges related to communication cost, power capability, computation, and storage capacity of IoT devices. Thus, high speed with low latency secure communication is

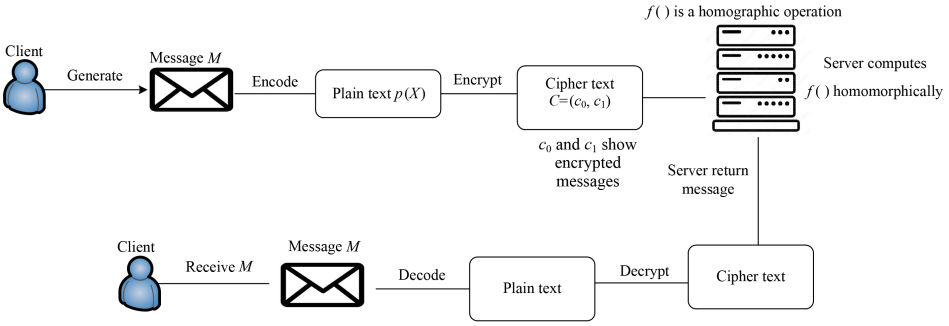


**Fig. 5 Full homomorphic CKKS operation diagram.**

important for many IoT applications to achieve the timely exchange of information security with other participants. Related to this, various computationally lightweight protocols, such as TLS and DTLS, have been proposed and implemented. However, the current TLS authentication method may suffer from heavy overhead, high communication costs, latency, and other security issues in constrained IoT environments. Instead of these protocols, we propose an iTLS protocol to deliver protected data with low communication latency and less overhead[36, 37]. A lightweight protocol is a type of protocol that has a minor and leaner payload, and has been used and transmitted over a network connection. The characteristics of the lightweight protocol are that it is faster, simpler, and easier to use and manage than other communication protocols on client/server-based network systems. Lightweight protocols leave out unessential communication steps and data, or may use data compression techniques that affect a network connection. Thus, iTLS has the same characteristics when it is used for client-server connections, because it is also considered an iTLS protocol. The iTLS protocol reduces communication overhead because it can identify irrelevant updates dynamically, as provided by the clients, and forbids unnecessary data from transferring. The aforementioned protocol is also equipped with inside techniques to dynamically detect clients at first connection and implement security filtering steps on participants compared with transport protocols, while also integrating some connection steps due to communication cost and latency[38, 39]. iTLS uses identity-based cryptography, which is known as asymmetric encryption or a public key cryptography mechanism, to establish an inherent binding between the public key and the entity that presents the public key. The IBAKA schema authenticates communication among different parties while establishing a shared key; in the process, it no longer needs to use a communication and verification certificate with iTLS while establishing the connection. iTLS uses the identity of both parties to generate the first key dynamically in order to protect the data in the first communication[10]. The authentication flowchart is shown in Fig. 6. As can be seen, different steps are followed to establish a communication channel. They are divided into two main stages.

• Initialization: This stage initializes or searches for the connection request to exchange channel-searching signals.

• Handshake: When the initialization stage is completed, the iTLS handshake protocol generates the encryption key and exchanges encryption values. Then, iTLS establishes a secure connection between participants.

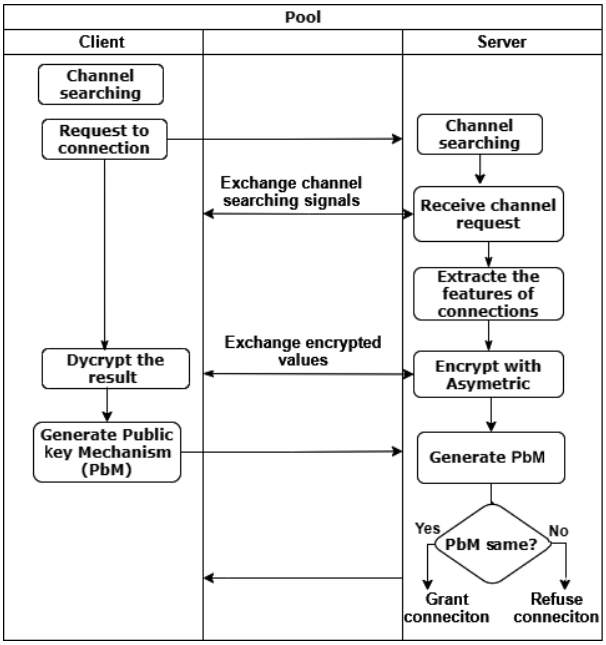


**Fig. 6 Flowchart of the lightweight iTLS transport protocol.**

## 5 Experiment

In this section, we evaluate the security and privacy algorithm under an IoT environment with one of the FL frameworks as that in the real world. Doing so ensures the integrity of the exchanged information between clients and coordinated servers. We propose a non-IID distribution (that is Independent Identically Distribution) type to train the model on a standard dataset named "MNIST" with the implementation of full homomorphic algorithm to encrypt model parameters for their local information privacy preservation. Moreover, we use iTLS protocol to speed up model performance during model training, reduce communication overhead, and decrease latency for improved communication efficiency.

### 5.1 Experimental environment

The purpose of our experiments is to define a Python-based FL with the support of lightweight protocols and FHE algorithms to enable clients and servers to have secure communication and decrease communication overhead and latency, while providing privacy preservation for model parameters in their local information in FL under an IoT environment. The logical diagram is shown in Fig. 7, which presents client-server communication based on CKKS and

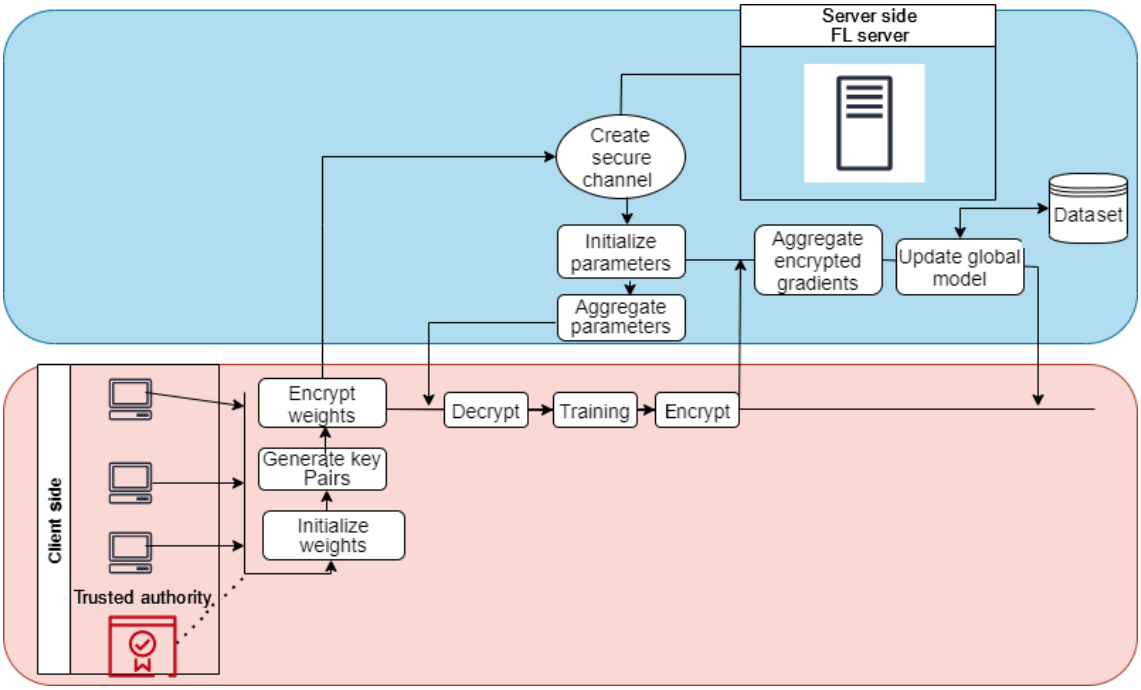


**Fig. 7 Logical diagram of client-server communication under CKKS and iTLS protocol.**

iTLS. This communication consists of various parts, namely, initialization, creating secure communication, generating encryption keys, and model training on both sides (client/local training and server/global training). We explain each part briefly as follows.

**(1) Initialization:** The iTLS protocols are initialized to determine whether it needs to send a message for the model update. If it is necessary, then the iTLS protocols create a secure channel between communication participants. In addition, a trusted authority generates a pair of keys (public key and secret key) based on the CKKS encryption scheme. Each client initializes the weight parameters of its local model (local weights) and encrypts it with the public key pk before sending it to a central or cloud server for aggregation. When the parameters are received, the server starts computation on the global model by aggregating all received parameters. Thus, the server sends the global model, enc (global, weights), to clients.

**(2) Local model training:** Once a client receives encrypted global model parameters enc (global weights), it performs decryption with a secret key (sk) to obtain the global model. Then, the client loads the global model into its local model and computes local model gradients through its private data resources. Once the computation of the gradients is completed, the model is encrypted again with the public key and returned to the cloud server for further processing.

**(3) Global model training:** As soon as the central or cloud server receives the encrypted gradients enc (local weights) from all distributed clients, it then computes all encrypted gradients, updates the global model, and then sends it back to all distributed clients.

### 5.2 Experimental configuration and result

The logical diagram shows the communication between FL clients and servers under the IoT environment shown in Fig. 7. This is validated by experiments with the implementation of CKKS and iTLS to decrease communication overhead and achieve low latency and data privacy (integrity and confidentiality). Many phases are included in performing the simulation. For a better explanation, we design the whole program flowchart to show the main step and the substeps in the system structure, which is shown in Fig. 8. Specifically, Fig. 8 shows the steps of the whole program structure and functions related to the logical design. The flowchart shows two processes related to logical design. The first one is the establishment of the client-server connection using iTLS. Once the connection is created, then the CKKS encryption schema starts the function to encrypt data and training model for privacy preservation and security.

The flowchart is designed based on some critical attributes in the logical diagram, Some program-related parameters with their descriptions are presented in

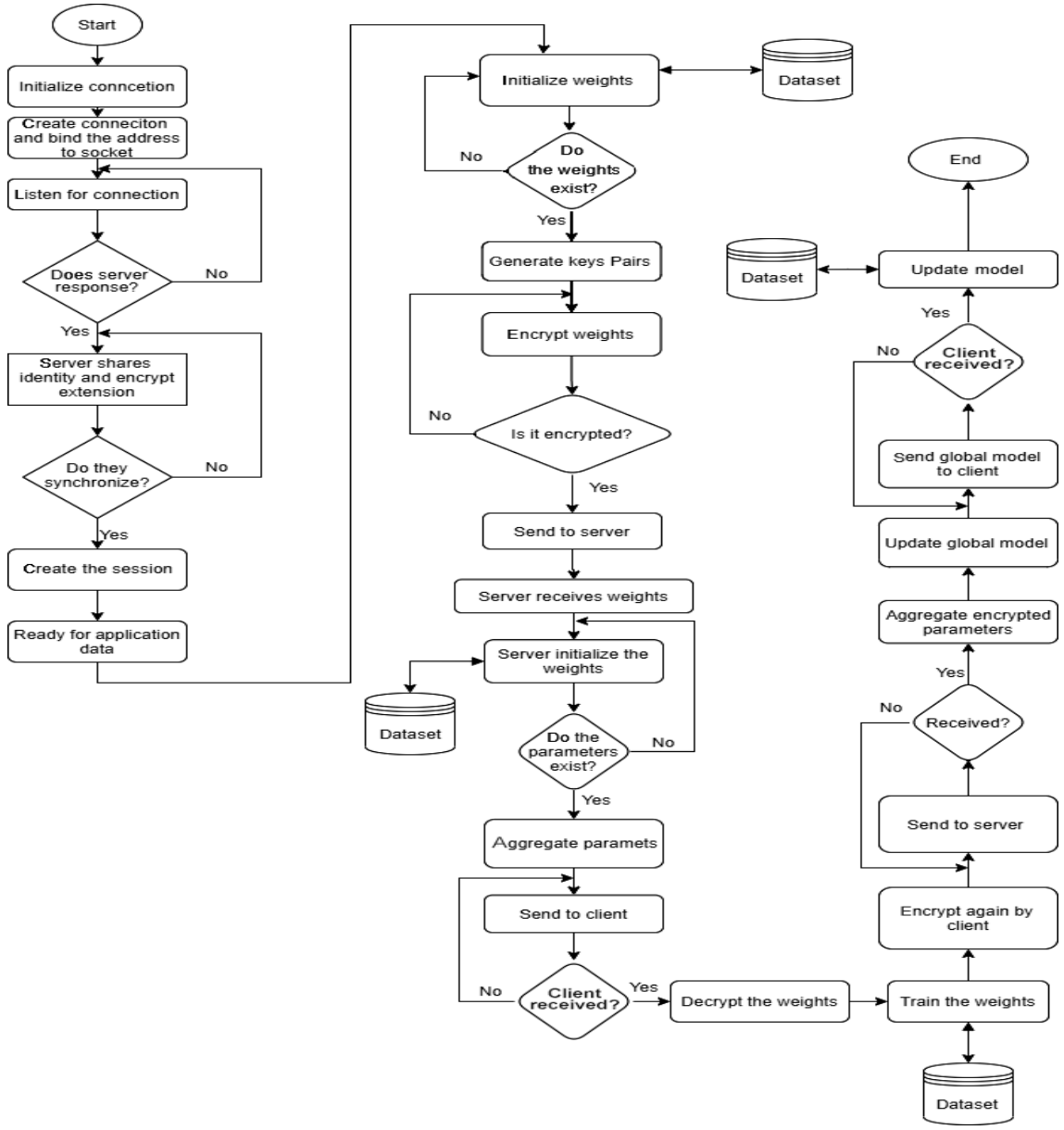


**Fig. 8 Client-server communication flowchart with the implementation of CKKS and iTLS protocol in the FL system.**

Table 1.

**(1) Initializer attribute:** First, both clients and server agents are initialized for the iteration of the training model, after which evaluation is performed to determine whether a model update is required. If it is needed, then the iTLS exchanges the flag message to create a channel between participants and exchange the keys with each other. This phase also distributes the correct and required amount of dataset for each client. Once the initializer phase is created, then simulation ( ) is invoked to start the simulation.

**(2) Simulation.py and config.py:** After the initialization phase, simulation ( ) invokes both the client and server agent. Here, a client creates a map for client agent names and shares it with other clients while the server is invoked by request_value ( ) from all participating clients. Each client tries to generate weights and trains the learning model throughout its own dataset $D$ for that iteration. Then, it sends a copy of the trained model to the server with a security offset. When the server receives the trained model, it proceeds to aggregate all models and returns them to the clients. Once the clients receive the aggregate model weights for each iteration, they then compute the accuracy of the federated model in contrast to its local model. This activity will be iterated to $i$ up to a certain number of

**Table 1 Parameters configuration.**

| Parameter | Explanation |
|---|---|
| iTLS protocol | This is a secure transport communication. If its state is "True", then new sockets and client-server connections will be established to exchange primary flag messages to be prepared for real data transmission. Otherwise, the communication circuit will not be created. |
| Security key | If it is "True", then the client performs key exchanges with other participants in the offline portion of the simulation to establish common keys for the encryption method. |
| Add Differential Privacy (DP) | If it is "True", then the CKKS allows adding noise following the important bit containing the focal message. In addition, this encryption noise is considered an error while executing approximate computation, thus ensuring the security hardness assumption. |
| Remove DP | If it is "True", the DP will be removed while data are received by the client in the FL system. Otherwise, the client will use the model computed by the server. |
| Client dropped-out | If it is "True", the client drops out after receiving the weights from each iteration in the FL model, and the simulation continues without clients. |
| Simulate and latency | If it is "True", the system simulates how long each step of the protocol of user-defined communication latency in the config.py file will take. Otherwise, the information will not be displayed. |

iterations.

**(3) Message. py:** All participants must communicate, invoke each other, and send the message for communication. The message contains all metadata about the communication and body attributes.

**(4) Client-agent.py:** This is the main instance of the simulation that trains the learning model. The agent has both the agent name and number to be determined during the simulation.

When the server invokes the agent, it will use the following two main methods:

• Generate weights: In this stage, the client transmits the learning model on its own dataset for each iteration using generate-weights (self, message).

• Receive weights: Receive-weights (self, message) is used when the server wants to return aggregated federated weights to the client. This message includes the iteration number, weight, and simulated time of the received message. If the client's weights converge to the federated weights, the method is considered true; otherwise, it is considered false. The dropped client method is used at the end of each iteration in the simulation, and this message contains a list of dropped clients and simulated time.

**(5) Server agent:** As mentioned previously, the server agent is an instance of the third party, and only coordinates all participating clients to exchange the training model and information. The server agent uses request_value ( ) to perform this activity.

**(6) CKKS encryption:** This step is done after the initializing step when the clients want to exchange data or training models with other participants. CKKS encryption is the FHE schema used for ensuring confidentiality and integrity (security and privacy). It is an encrypting training model that exchanges between various participants in the FL system, especially between client and server. To perform this function, the CKKS schema generates two types of keys (secret and public keys). The client encrypts the message with the public key and sends it to the server, after which the server aggregates all encrypted messages received from different clients. Such operations are done over encrypted data. Based on a client's request, the server returns the aggregated encrypted data to the client. Using the secret key, the client performs data decryption and proceeds to learn the model. Some program-related parameters with their descriptions are presented in Table 1.

### 5.3 Results

When the simulation runs, the iTLS and CKKS algorithms start their function according to the steps explained in the configuration environment of the simulation above. At this point, the iTLS protocol provides the communication channel between participants and gives clients a chance to share their identity with others, thus decreasing unnecessary communication steps between participants. As shown in the iTLS communication sequence diagram in Fig. 9, the iTLS protocol employs a new extension used by the client and server under a Client Hello and server message. The aim is to exchange the identity details and cryptographic parameters used for shared secrets between them. In addition, all components belonging to certificates and certificate verification messages are ignored because those need

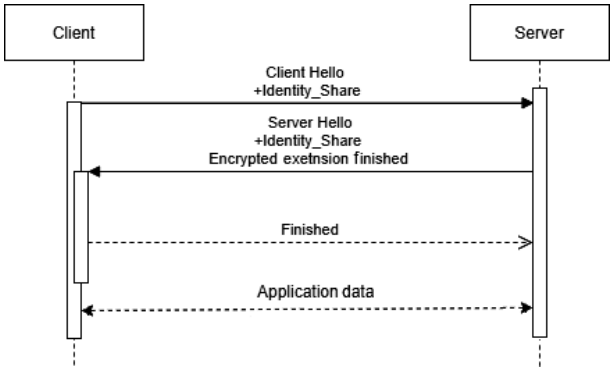


**Fig. 9 iTLS protocol communication steps.**

more communication steps and increase communication overhead when exchanged by other protocols, such as TLS. Therefore, this decrement in communication steps improves performance and decreases communication overhead and latency.

Comparisons of latency and unnecessary round trip messages between iTLS and TLS based on the packet loss percentage are presented in Figs. 10 and 11. In particular, Fig. 10 shows that TLS increases the number of messages and has increased the latency rapidly, thus affecting the connection between client and server promptly. In comparison, Fig. 11 shows iTLS with the CKKS scheme, which prevents unnecessary messages and decreases the latency with packet losses, which

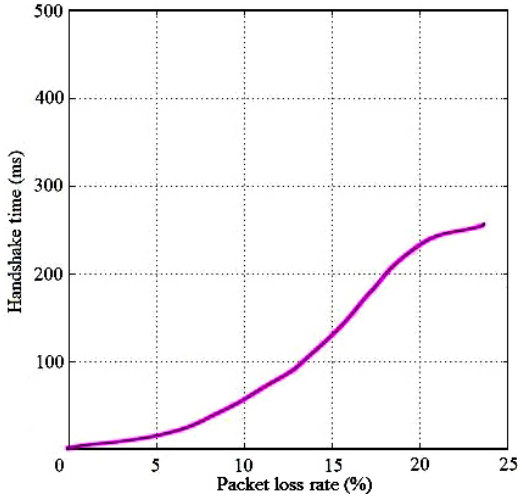


**Fig. 10 Handshake latency (packet loss) in TLS.**

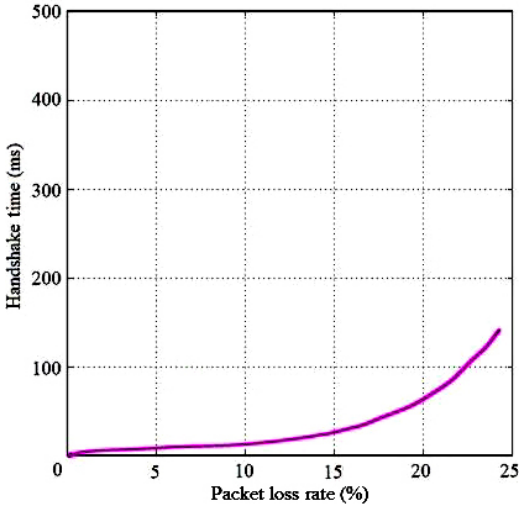


**Fig. 11 Handshake latency (packet loss) in iTLS.**

means the client-server connection is less affected in contrast to TLS. Furthermore, Fig. 11 shows that the proposed scheme speeds up the computation process. Table 2 shows the comparison of various schemes and algorithms used for FL security and privacy, and computation process. Due to their computation process and roundtrip messages, they have a different measure of accuracy. As indicated by the results, our proposed scheme limits unnecessary roundtrip messages, speeds up the computation process, and increases accuracy.

The comparison of communication overhead between TLS and iTLS with the CKKS scheme protocol is shown in Fig. 12. As shown in the chart, for both


**Fig. 12 Communication overhead comparison between iTLS and TLS.**

**Table 2 Accuracy comparison between current and previous scheme.**

| Scheme | Accuracy (%) | Reason | Dropout |
|---|---|---|---|
| MPC | 82.0 | Decreased due to the conversion from floating-point operations to modular computations | Yes |
| DP | 77.0 | Decreased due to noise and additional communication rounds | Yes |
| FedAvg/SecAgg | 73.0 | Less accurate due to additional computation and communication processes | – |
| Our scheme | 95.3 | Increased due to the prevention of additional communication rounds; faster computation process | No |

protocols during the handshake, the TLS generated 1840 bytes. This is because using iTLS means that the client and server need not exchange any additional certificate components and verify messages. Then, they generate 850 bytes in size, because iTLS reduces communication overhead by about 65 bytes. In addition, the CKKS algorithm starts its own mechanism to encrypt the gradients of the model, which are supposed to be sent to the server for model updating, thus saving information against potential attackers and reducing activities to exploit information integrity and confidentiality. This process also helps mitigate risks that are harmful to model training security. As mentioned previously, to improve the FL system's model training efficiency and security, the proposed method uses a CKKS encryption scheme to preserve the data resource of the distributed client and encrypt model parameters, as shown in Fig. 5. The CKKS encryption method consists of the following three functions:

**(1) Key generation:** Security parameters and ring degree are defined by $n$, the cipher text modulus is represented by $P$. In this process, the message is encoded and converted to plaintext. Thus, the trust authority generates pair of keys (pk and sk) based on the CKKS-defined cryptosystem standard.

**(2) Model encryption:** This process encrypts the model parameters using the public key generated by a trusted authority.

**(3) Model aggregation in the cloud server:** In this step, the cloud server receives encrypted messages or gradients from all distributed clients, after which it aggregates all gradients and creates/updates a global model.

### 5.4 State-of-art

FL is a distributed and collaborative AI method, in which a global model is trained over distributed datasets, and clients only need to train their local models over their private datasets before sending them to a central server without real data to improve data privacy. However, as the number of connected devices grows, the FL systems need high-speed communication, along with low communication overhead and latency, as well as privacy, accuracy, and best computational power. Numerous studies have investigated FL privacy and communication overhead, but none have worked effectively as required by the FL system. Thus, the current paper compares previous research with our research scheme, as shown in Table 3.

## 6 Conclusion

This section presents the conclusion according to the research conducted in this paper. Here, we use the iTLS transport protocol and the CKKS FHE schema, both of which play important roles in data and model security and privacy. These also enhance the performance of data processing, decrease communication overhead, and reduce latency, while data are exchanged between various participants in FL systems. The required implementation of the Confidentiality Integrity Availability (CIA) triangular is also completed in this research to achieve data security. Whenever a client wants to send model weights and gradients for the model update to the server, it must first determine whether the model update is necessary and when a client is able to send gradients to the server. Thus, this effort allows participants to confirm the availability of transferred data between client and server. Furthermore, the data will not be transferred periodically if not needed, thus reducing the number of communication steps while creating a client-server channel to enhance the communication performance and decrease communication overhead. The CKKS encryption schema starts its mechanism to encrypt the model, thus protecting information against potential attackers. This also prevents hacking activities that can harm the integrity and confidentiality of the information, which are harmful to model training security. This security and privacy fact and communication performance allow users to use cloud structure to perform processing in different domains of human life, such as medical systems, education, banking, and other structures, which need data security

**Table 3 Comparison of the present scheme with referenced scheme.**

| Scheme | Accuracy and computation process | Communication overhead and latency | Operation type | More training and communication iteration | Dropping in model performance |
|---|---|---|---|---|---|
| Ours | High and faster | Less | Complex (adding and multiplying) | Reduced | No |
| Others | Less and slower[40] | More[10, 41] | Simple (either adding or multiplying)[42, 43] | Not-reduced[36, 44] | Yes[9, 43] |

and consistency.

To enhance data security in cloud-based structures, future research may consider the MultiKey HE (MKHE) schema, as it can perform the mathematical processes on cipher text encrypted under various keys.

## Acknowledgment

This work was supported by the National Key Research and Development Program of China (No. 2018YFB0803403) and the Fundamental Research Funds for the Central Universities (Nos. FRF-AT-20-11 and FRF-AT-19-009Z) from the Ministry of Education of China.

## References

[1] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, Privacy-preserving blockchain-based federated learning for IoT devices, *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, 2021.

[2] D. Chen, V. Tan, Z. Lu, and J. Hu, OpenFed: A comprehensive and versatile open-source federated learning framework, arXiv preprint arXiv: 2109.07852, 2023.

[3] L. Zhang, Z. Zhang, and C. Guan, Accelerating privacy-preserving momentum federated learning for industrial cyber-physical systems, *Complex Intell. Syst.*, vol. 7, no. 6, pp. 3289–3301, 2021.

[4] B. Jeon, S. M. Ferdous, M. R. Rahman, and A. Walid, Privacy-preserving decentralized aggregation for federated learning, in *Proc. IEEE INFOCOM 2021–IEEE Conf. Computer Communications Workshops*, Vancouver, Canada, 2021, pp. 1–6.

[5] M. Asad, A. Moustafa, and C. Yu, A critical evaluation of privacy and security threats in federated learning, *Sensors*, vol. 20, no. 24, p. 7182, 2020.

[6] M. Alazab, S. Priya, M. Parimala, P. K. R. Maddikunta, T. R. Gadekallu, and Q. V. Pham, Federated learning for cybersecurity: Concepts, challenges, and future directions, *IEEE Trans. Ind. Inform.*, vol. 18, no. 5, pp. 3501–3509, 2022.

[7] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, Beyond inferring class representatives: User-level privacy leakage from federated Learning, in *Proc. IEEE INFOCOM 2019–IEEE Conf. Computer Communications*, Paris, France, 2019, pp. 2512–2520.

[8] Y. Aono, T. Hayashi, L. T. Phong, and L. Wang, Privacy-preserving logistic regression with distributed data sources via homomorphic encryption, *IEICE Trans. Inf. Syst.*, vol. E99-D, no. 8, pp. 2079–2089, 2016.

[9] D. Stripelis, H. Saleem, T. Ghai, N. Dhinagar, U. Gupta, C. Anastasiou, G. V. Steeg, S. Ravi, M. Naveed, P. M. Thompson, et al., Secure neuroimaging analysis using federated learning with homomorphic encryption, arXiv preprint arXiv: 2108.03437, 2021.

[10] P. Li, J. Su, and X. Wang, iTLS: Lightweight transport-layer security protocol for IoT with minimal latency and perfect forward secrecy, *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6828–6841, 2020.

[11] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle, Towards viable certificate-based authentication for the internet of things, in *Proc. 2nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy*, Budapest, Hungary, 2013, pp. 37–42.

[12] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, Towards efficient and privacy-preserving federated deep learning, in *Proc. IEEE Int. Conf. Communications* (*ICC*), Shanghai, China, 2019, pp. 1–6.

[13] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, Federated learning: Challenges, methods, and future directions, *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, 2020.

[14] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtep, H. Kim, and S. Nepal, End-to-end evaluation of federated learning and split learning for internet of things, in *Proc. 2020 Int. Symp. Reliable Distributed Systems* (*SRDS*), Shanghai, China, 2020, pp. 91–100.

[15] C. Shen and W. Xue, An experiment study on federated learning testbed, in *Proc. SmartCom 2021*, Singapore, 2021, pp. 209–217.

[16] M. Yang, Y. He, and J. Qiao, Federated learning-based privacy-preserving and security: Survey, in *Proc. Computing, Communications and IoT Applications* (*ComComAP*), Shenzhen, China, 2021, pp. 312–317.

[17] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J. P. Bossuat, J. S. Sousa, and J. P. Hubaux, POSEIDON: Privacy-preserving federated neural network learning, arXiv preprint arXiv: 2009.00349, 2021.

[18] N. Truong, K. Sun, S. Wang, F. Guitton, and Y. Guo, Privacy preservation in federated learning: Insights from the GDPR perspective, arXiv preprint arXiv: 2011.05411, 2021.

[19] N. B. Truong, K. Sun, G. M. Lee, and Y. Guo, GDPR-Compliant personal data management: A blockchain-based solution, *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1746–1761, 2020.

[20] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, Federated optimization: Distributed machine learning for on-device intelligence, arXiv preprint arXiv: 1610.02527v1, 2016.

[21] 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 *Proc. 2017 ACM SIGSAC Conf. Computer and Communications Security*, Dallas, TX, USA, 2017, pp. 1175–1191.

[22] 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 Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[23] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, Federated learning: Strategies for improving communication efficiency, arXiv preprint arXiv: 1610.05492, 2017.

[24] M. Asad, A. Moustafa, T. Ito, and M. Aslam, Evaluating the

communication efficiency in federated learning algorithms, in *Proc. IEEE 24th Int. Conf. Computer Supported Cooperative Work in Design*, Dalian, China, 2021, pp. 552–557.

[25] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, FedAT: A high-performance and communication-efficient federated learning system with asynchronous tiers, in *Proc. SC21: Int. Conf. High Performance Computing, Networking, Storage and Analysis*, St. Louis, MO, USA, 2021, pp. 1–17.

[26] J. Ma, S. A. Naas, S. Sigg, and X. Lyu, Privacy-preserving federated learning based on multi-key homomorphic encryption, arXiv preprint arXiv: 2104.06824v1, 2021.

[27] V. Mugunthan, A. Peraire-Beuno, and L. Kagal, PrivacyFL: A simulator for privacy-preserving and secure federated learning, in *Proc. 29th ACM Int. Conf. Information and Knowledge Management*, Virtual Event, 2020, pp. 3085–3092.

[28] K. Rangasami and S. Vagdevi, Comparative study of homomorphic encryption methods for secured data operations in cloud computing, in *Proc. 2017 Int. Conf. Electrical, Electronics, Communication, Computer, and Optimization Techniques* (*ICEECCOT*), Mysuru, India, 2017, pp. 1–6.

[29] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, A survey on homomorphic encryption schemes: Theory and implementation, *ACM Comput. Surv*., vol. 51, no. 4, p. 79, 2018.

[30] M. Ogburn, C. Turner, and P. Dahal, Homomorphic encryption, *Procedia Comput. Sci*., vol. 20, pp. 502–509, 2013.

[31] J. He, B Gong, and J. Yang, ASCFL: Accurate and speedy semi-supervised clustering federated learning, *Tsinghua Science and Technology*, vol. 28, no. 5, pp. 823–837, 2023.

[32] H. Chen, W. Dai, M. Kim, and Y. Song, Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference, in *Proc. 2019 ACM SIGSAC Conf. Computer Communication Security*, London, UK, 2019, pp. 395–412.

[33] J. H. Cheon, A. Kim, M. Kim, and Y. Song, Homomorphic encryption for arithmetic of approximate numbers, in *Proc. 23rd Int. Conf. Theory and Applications of Cryptology and Information Security*, Hong Kong, China, 2017, pp. 409–437.

[34] D. Shehada, A. Gawanmeh, C. Fachkha, and H. A. Damis, Performance evaluation of a lightweight IoT authentication protocol, in *Proc. 3rd Int. Conf. Signal Processing and Information Security* (*ICSPIS*), DUBAI, United Arab Emirates, 2020, pp. 1–4.

[35] A. Diro, H. Reda, N. Chilamkurti, A. Mahmood, N. Zaman, and Y. Nam, Lightweight authenticated-encryption scheme for internet of things based on publish-subscribe communication, *IEEE Access*, vol. 8, pp. 60539–60551, 2020.

[36] M. N. Khan, A. Rao, and S. Camtepe, Lightweight cryptographic protocols for IoT-Constrained devices: A survey, *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4132–4156, 2021

[37] P. Liu, X. Xu, and W. Wang, Threats, attacks and defenses to federated learning: Issues, taxonomy and perspectives, *Cybersecurity*, vol. 5, no. 1, p. 4, 2022.

[38] O. Shahid, S. Pouriyeh, R. M. Parizi, Q. Z. Sheng, G. Srivastava, and L. Zhao, Communication efficiency in federated learning: Achievements and challenges, arXiv preprint arXiv: 2107.10996v1, 2021.

[39] L. Wang, W. Wang, and B. Li, CMFL: Mitigating communication overhead for federated learning, in *Proc. IEEE 39th Int. Conf. Distributed Computing System* (*ICDCS*), Dallas, TX, USA, 2019, pp. 954–964.

[40] J. Loya and T. Bana, Privacy-preserving keystroke analysis using fully homomorphic encryption & differential privacy, in *Proc. Int. Conf. Cyberworlds* (*CW*), Caen, France, 2021, pp. 291–294.

[41] E. Rescorla, The Transport Layer Security (TLS) Protocol version 1.3, Internet Engineering Task Force (IETF), RFC 8446, https://www.rfc-editor.org/info/rfc8446, 2018.

[42] K. Lauter, M. Naehrig, and V. Viakuntanathan, Can homomorphic encryption be practical? in *Proc. Association for Computing Machinery* (*ACM*) CCSW, 2011, pp.113–124.

[43] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, Deep learning with differential privacy, in *Proc. ACM SIGSAC Conf. Computer and Communications Security*, Vienna, Austria, 2016, pp. 308–318.

[44] U. Gupta, D. Srtipelis, P. K. Lam, P. M. Thompson, J. L. Ambite, and G. Ver Steeg, Membership inference attacks on deep regression models for neuroimaging, *Mach. Learn. Res.*, vol. 143, pp. 228–251, 2021.

**Nasir Ahmad Jalali** received the MEng degree in computer science (information technology) from Kabul University, Afghanistan. He is a PhD candidate at University of Science and Technology Beijing (USTB), China. He has been an assistant professor at Ghazni University, Afghanistan since 2012. His research area is network security, information security, wired and wireless network, machine learning, and big data. He has published five academic papers. He got the outstanding student award at USTB in 2022. Also, he is the author of two books entitled *MPLS-VPN Impacts on VoIP-QoS* and *Framework Development for Higher Education Information Security in Afghanistan*.

**Hongsong Chen** received the PhD degree in computer science from Harbin Institute of Technology, China in 2006. He has been a professor at Department of Computer Science, University of Science and Technology Beijing (USTB), China since 2008. He was a visiting scholar at Department of Computer Science, Purdue University, USA during 2013–2014. He is a high-level member of the China Computer Federation. He is an IEEE member now. His research interests include artificial intelligence and information security, wireless network and pervasive computing, and trust computing. He received the Excellent Young Academic Paper Award at USTB in 2009. He has published more than 60 academic papers and 6 books.