applied sciences

MDPI

Article

Privacy-Preserving Federated Learning Using Homomorphic Encryption

Jaehyoung Park [1] and Hyuk Lim [2,*]

[1] School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, Korea; jaehyoungpark@gist.ac.kr
[2] AI Graduate School, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, Korea
* Correspondence: hlim@gist.ac.kr

Abstract: Federated learning (FL) is a machine learning technique that enables distributed devices to train a learning model collaboratively without sharing their local data. FL-based systems can achieve much stronger privacy preservation since the distributed devices deliver only local model parameters trained with local data to a centralized server. However, there exists a possibility that a centralized server or attackers infer/extract sensitive private information using the structure and parameters of local learning models. We propose employing homomorphic encryption (HE) scheme that can directly perform arithmetic operations on ciphertexts without decryption to protect the model parameters. Using the HE scheme, the proposed privacy-preserving federated learning (PPFL) algorithm enables the centralized server to aggregate encrypted local model parameters without decryption. Furthermore, the proposed algorithm allows each node to use a different HE private key in the same FL-based system using a distributed cryptosystem. The performance analysis and evaluation of the proposed PPFL algorithm are conducted in various cloud computing-based FL service scenarios.

Keywords: privacy preserving; homomorphic encryption; federated learning

Citation: Park, J.; Lim, H. Privacy-Preserving Federated Learning Using Homomorphic Encryption. *Appl. Sci.* **2022**, *12*, 734. https://doi.org/10.3390/app12020734

Academic Editors: Safwan El Assad, René Lozi and William Puech

Received: 14 December 2021
Accepted: 7 January 2022
Published: 12 January 2022

1. Introduction

Artificial intelligence (AI) is a technology that enables machines to realize human learning and reasoning abilities. This technology has been rapidly advancing and playing a significant role in our daily lives. In AI technology, data acquisition is crucial because AI technologies require model training using a certain amount of data for reliable AI-based services, and the performance of AI-based services is considerably affected by the training data quality. However, there are difficulties in data collection because the data may contain sensitive private information. In order to overcome these difficulties, federated learning (FL), in which training is performed without sharing sensitive local data, has been proposed in [1]. In FL, a centralized server sends a global model for AI learning to many distributed devices, which return local model parameters to the centralized server after training the model with local data. The centralized server updates the global model parameters using the locally trained model parameters from the distributed devices and sends the updated global model parameters to the distributed devices. This procedure is repeated until convergence is achieved. FL has the advantage of preventing the leakage of sensitive private information because it does not require local data sharing. However, recent research has shown that the local data of distributed devices can be leaked through the trained local model parameters, and attackers can exploit this loophole to infer sensitive information on the FL participant in [2,3].

Homomorphic encryption (HE) is a technology that enables arithmetic operations on ciphertexts without decryption. Aono et al. utilized an HE scheme to protect local gradients trained with local data in [2]. Using the HE scheme, the centralized server

can update the global model parameters with the encrypted local gradients based on the homomorphic operation. Therefore, the distributed devices participating in FL, which we refer to as clients, do not have to concern about data leakage through local gradients because they deliver encrypted local gradients to the server. However, the clients must share the same private key in the FL-based system because homomorphic operations can only be performed between values encrypted with the same public key. In FL-based systems where many distributed devices, such as smartphones and Internet of Things (IoT) devices participate, the same private key for decryption can be distributed to many clients. Suppose the same private key is shared with many clients. Then, the probability of the private key being leaked or a malicious participant accessing other participants' data increases, which can weaken privacy protections in FL-based systems. As the result, stealing one client's private key can nullify the data privacy protection of all clients participating in FL systems. To overcome this vulnerability, this paper proposes a privacy-preserving federated learning (PPFL) algorithm that allows a cloud server to update global model parameters by aggregating local parameters encrypted by different HE keys in the same FL-based system using homomorphic operations based on a distributed cryptosystem.

This paper is organized as follows. In Section 2, we present related works on FL and the privacy issues of FL. Section 3 describes the preliminaries for understanding the FL algorithm and the cryptosystem for homomorphic operations. In Section 4, we describe the system and attack models for the proposed PPFL algorithm. Next, Section 5 explains the proposed PPFL algorithm using the distributed cryptosystem based on an additive homomorphic encryption (AHE) scheme. Afterwards, Section 6 presents a theoretical analysis of the proposed PPFL algorithm, and Section 7 presents experimental results to verify the performance of the proposed PPFL scheme. Finally, Section 8 concludes the paper.

2. Related Work

FL is one possible solution for preserving privacy in the machine learning field because the clients participating in the training process deliver only local model parameters trained with local data to a centralized server. McMahan et al. demonstrated the feasibility of FL by conducting empirical evaluation in various FL scenarios in [1]. Since then, many studies have been conducted to improve FL performance for learning accuracy, fairness, robustness, security, and privacy in various environments, such as IoT, edge, and cloud computing in [4–6]. In [7], a lightweight federated multi-task learning framework was proposed to provide fairness among participants and robustness against a poisoning attack that reduces learning accuracy. In [8], an FL framework using device-to-device communication was proposed to overcome the degradation in energy and learning efficiency due to frequent uplink transmissions between participants and physically distant central servers.

The studies on FL can be classified according to how they collect and process data for FL. In a case where the data have the same feature space and a different sample space, it is classified as horizontal FL, and in a case where the data have a different feature space and the same sample space, it is classified as vertical FL in [9]. In vertical FL, data alignment must be performed for vertical data utilization by sharing several different feature spaces. In this process, privacy is not protected because row data exchange may be required. For preserving data privacy, HE-based vertical FL algorithms were implemented by utilizing a trusted third party [4,9–11]. An approach to collaboratively train a high-quality tree boosting model was proposed to simplify FL-based systems by omitting third parties and showed that the performance of the proposed scheme was as accurate as the performance of centralized learning techniques in [12]. Horizontal FL is an algorithm in which multiple devices train a learning model using local data with the same feature space and share the trained model data to train a global model, and the scheme presented in [1] was a representative horizontal FL. The horizontal FL can be implemented without a data alignment process because it has the same feature space. Although many studies have been conducted for the development of FL, privacy threats still exist in FL. It was shown that

sensitive private data could be leaked through the local gradients in [2,13], and participants' data can be inferred through a generative adversarial network using the global and the local model parameters in [3].

Several studies have been conducted to solve the privacy issues associated with the local model parameters in an FL-based system in [2,13–15]. Shokri and Shmatikov proposed a privacy-preserving deep learning (PPDL) algorithm where several distributed participants collaborate to train a deep learning model using local data; they established a trade-off relationship between practicality and security for the number of clients participating in the training process in [13]. Moreover, Aono et al. suggested a PPDL algorithm that encrypts local model parameters using HE schemes to protect the local and global model parameters in [2]. In the algorithm proposed in [2], strict key management is required and reliable channels for conveying ciphertexts must be established because all participants use the same private key for HE. In [16], HE-based federated learning was proposed, and its overhead was analyzed. However, all clients participating in training still use the same key in the system. In [14], based on Shamir's t-out-of-n Secret Sharing in [17], they presented an algorithm that allows the server to perform updates using local model parameters containing noise that can be canceled out through the cooperation of the participants in an FL-based system. The scheme proposed in [14] can prevent leakage of local model parameters due to the noise contained in the local parameter but can be vulnerable to insider attacks because participants must actively cooperate. Recently, Xu et al. offered a technique in which participants verify the integrity of the updated results in the system that updates the global model parameters using the Secret Sharing scheme in [15].

The algorithms using the HE scheme in [2] and Secret Sharing in [14] have shown that neural networks can be safely trained without personal information leakage in FL scenarios. In [2], all training process participants owned the same private key, although the distributed deep learning system using the HE scheme was designed to protect shared data. For this reason, all channels between participants and servers must be protected using transport layer security or secure socket layer. However, as the number of participants increases, the cost to establish secure channels becomes very high. In addition, since the probability of one participant's private key being leaked is the same as the probability of all participant's private keys being leaked in this system, the risk of private key leakage increases as the number of participants grows. In the proposed system, each participant can own different private keys on the same FL-based system. In [14], at least half of the participants must guarantee their honesty for privacy preservation. If the number of participants in FL is large, this assumption is reasonable, but there can be a variety of FL scenarios. The proposed system allows participants to preserve data privacy regardless of the number of honest participants and can be utilized as a solution to build a flexible PPFL-based system.

This paper presents a PPFL algorithm based on a distributed cryptosystem using the AHE scheme to protect the local and global model parameters. The participant uses the HE scheme to encrypt the local model parameters with its private key, and the cloud server updates the global model parameters with the local model parameter encrypted with different keys based on the distributed cryptosystem. The proposed PPFL-based system can achieve robust privacy protection because the proposed algorithm can allow each node to use a different private key for the HE scheme in the same FL-based system. Furthermore, a highly flexible FL-based system can be built using our algorithm because clients only need to encrypt and decrypt model parameters to protect them.

3. Preliminary

3.1. Federated Learning

In FL, multiple distributed servers or devices with local data train a machine learning model without exchanging local data. Distributed servers or devices share only local model parameters obtained by training a global learning model delivered from a centralized server with local data, allowing them to participate in the training process without concern about data leakage. The centralized server aggregates locally trained model parameters to update

the global model and delivers the updated global parameters to distributed servers or devices to perform the training process again. This procedure is repeated until convergence is achieved.

According to the data distribution characteristics, FL can be categorized into horizontal federated learning, vertical federated learning, and federated transfer learning in [9]. Horizontal and vertical FL algorithms are applied when local datasets have the same feature space and different sample spaces and when local datasets have different feature spaces and the same sample space, respectively. Federated transfer learning is applied to a scenario where the local datasets have varying features and minimal overlapping samples. In this case, federated transfer learning utilizes the transfer learning techniques in [18] for FL-based systems. We consider horizontal FL in this paper. In other words, we assume that the local datasets have the same feature space and different sample spaces, and we consider an FL scenario in which many clients, including smartphones and IoT devices, participate in the training process.

3.2. Homomorphic Encryption

HE is a form of encryption that allows third parties to perform arithmetic operations directly with ciphertexts, and the HE scheme can be utilized to develop PPML in fields where data privacy is important. First, partial homomorphic encryption (PHE) capable of only addition or multiplication was developed. For example, the property of the AHE scheme, which can only perform addition operations, is represented as follows in [19]:

$$D_{sk_i}(E_{pk_i}(m_1) \cdot E_{pk_i}(m_2)) = m_1 + m_2, \tag{1}$$

where $D_{sk_i}(\cdot)$ is a decryption function using a private key sk_i, $E_{pk_i}(\cdot)$ is an encryption function using a public key pk_i, and m_i is a plaintext. The cloud server can perform homomorphic addition operations without decryption using (1). Subsequently, fully homomorphic encryption (FHE) capable of both addition and multiplication operations was established in [20] to overcome the limitations of PHE, which is challenging to implement various homomorphic operations in [21]. FHE enables a variety of operations to be implemented using addition and multiplication operations. These HE technologies have led to the development of PPML algorithms in the cloud and machine learning fields.

3.3. Distributed Homomorphic Cryptosystem

Distributed homomorphic cryptosystem (DHC) is a cryptosystem that can perform various homomorphic operations using secure multiparty computation (SMC) for implementing various homomorphic operations in a distributed manner. Figure 1 illustrates the decryption process of DHC. In typical public-key cryptography, parties with public and private keys perform encryption and decryption for secure communication, respectively. On the other hand, a private key is divided into several partial private keys in the DHC, and the partial private keys are distributed to multiple distributed servers. Distributed servers with partial private keys perform partial decryption using values encrypted with the public key. The other distributed server can obtain the plaintext by collecting the partially decrypted ciphertexts. This decryption process enables a variety of homomorphic operations based on multilateral cooperation.

The functions for the DHC are described as follows:

- **Key generation**: Function that generates a public-private key pair (pk_i, sk_i), $i \in \{1, \cdots, N_c\}$ of a user for given two large prime numbers p and q, where N_c is the number of clients participating in the local training.The public key is calculated by $p \cdot q$, and the private key corresponding to the public key is calculated by $lcm(p-1, q-1)/2$, where $lcm(x, y)$ denotes the least common multiple (LCM) of x and y. Note that the key size K is $p \cdot q$. Then, as the selected prime numbers increase, the computational complexity of cryptosystem increases because the complexity for the exponentiation operation of encryption and decryption increases [22]. Then, partially private keys

$[psk_i^{(1)}, psk_i^{(2)}, \cdots, psk_i^{(N_s)}]$ for distributed servers can be obtained by splitting the private key sk_i, where N_s is the number of distributed servers [23]. We select δ that satisfies $\delta \equiv 0 \bmod sk_i$ and $\delta \equiv 1 \bmod K^2$ at the same time and select y random numbers $\{a_1, a_2, \ldots, a_y\}$ from $\mathbb{Z}^*_{sk_iK^2}$. Then, we use these values to define the polynomial $p(x) = \delta + \sum_{i=1}^{y} a_i x^i$. The partial private key $psk_i^{(j)}$ is obtained by calculating the polynomial $p(x_j)$ using a non-zero value x_j from $\mathbb{Z}^*_{sk_iK^2}$.

- **Encryption**: Function that generates a ciphertext $E_{pk_i}(m) \in \mathbb{Z}^*_{K^2}$ for a plaintext $m \in \mathbb{Z}_K$, using a public key pk_i, where the key size K is $p \cdot q$. For simplicity, the ciphertext $E_{pk_i}(m)$ can be represented by $[m]_i$;
- **Decryption**: Function that decrypts a ciphertext $[m]_i$ using a private key sk_i and returns m;
- **Partial decryption**: Function that generates a partially decrypted ciphertext by partially decrypting $[m]_i$ using a partial private key $psk_i^{(k)}, k \in \{1, \cdots, N_s - 1\}$, as shown in Figure 1. For simplicity, the partially decrypted ciphertext $PD_{psk_i^{(k)}}([m]_i)$ can be denoted by $PD^k([m]_i)$;
- **Combined decryption**: Function that obtains and returns m using $(N_s - 1)$ partially decrypted ciphertexts $PD^k([m]_i)$ for $\forall k \in \{1, \cdots, N_s - 1\}$ and the partially private key $psk_i^{(N_s)}$. Note that a vector $\boldsymbol{PD([m]_i)}$ signifies $[PD^1([m]_i), PD^2([m]_i), \cdots, PD^{N_s-1}([m]_i)]$ in Figure 1.

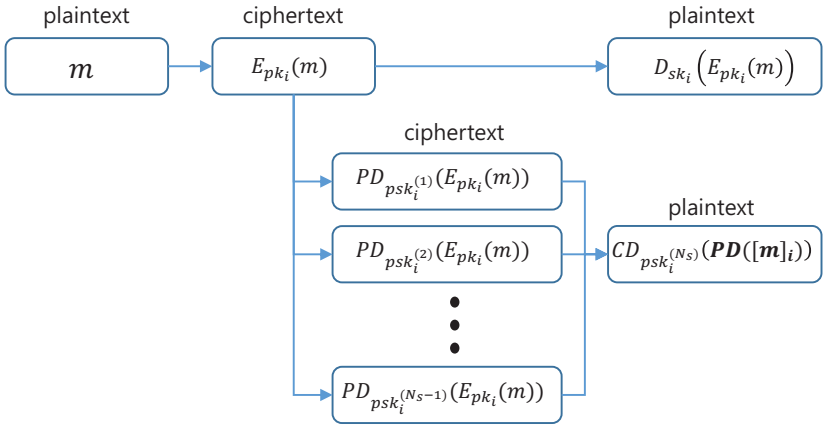

Figure 1. Diagram for encryption, decryption, and partial decryption.

Using the DHC, we have established a PPFL-based system in which the parties can jointly perform a global model update based on homomorphic operations to preserve the data privacy of the participants in the FL training process. The proposed PPFL algorithm is explained in detail in Section 5.

4. System and Attack Models

4.1. System Model

We consider a horizontal FL scenario that operates in a cloud system using SMC. A large amount of local model parameters is exchanged between the cloud server and many clients. Our proposed system encrypts the local model parameters using the DHC as described in Section 3.3 to protect data privacy. When a client first participates in FL, the certified key generation center generates a private and public key pair for the client. The private is sent to the client, and the public key is distributed to the client and servers in the system through a secure channel. In addition, after generating the private key, the certified key generation center splits the key into as many partial private keys as the number of the authenticated cloud server and computation provider, and distributes them to the servers one by one through a secure channel. If the private keys are stolen during the private key delivery process, the entire system may collapse. In the proposed system, secure channels

for the private key delivery are built using secure sockets layer or transport layer security protocol. Since the private key delivery is performed intermittently, the possibility of an attacker stealing the private key is extremely low in the system. The cloud server and computation provider collaborate with each other to update the global model using the model parameters encrypted with different private keys from clients. Once the cloud server receives a set of model parameters from a client, it adds a random noise encrypted with the client's public key to the set of model parameters, partially decrypts it with the client's partial private key, and delivers it to the computation provider. The computation provider server obtains the partial decrypted sets of model parameters for the clients and decrypts them using the other partial private keys of the clients. Finally, the computation provider performs the model aggregation and encrypts it with the public key for each client, and returns it to the cloud server. The cloud server removes the random noise from the encrypted global model and sends it to each client. The detailed update process is described in Section 5.3.

Figure 2 depicts a simple system model comprising a key generation center (KGC), cloud server (CS), computation provider (CP), and multiple clients. The KGC is a trusted organization that performs authentication procedures for clients and servers and generates key pairs. The CS is responsible for securely combining the trained parameters on the clients and can select clients at every iteration for FL. The CP communicates directly with the CS and provides computational resources for requests of the CS. A single CP or multiple CPs can exist in the system, and the CPs and the CS perform cooperative encryption described in Section 3.3. Clients own each private key for decryption and perform local training with local datasets. In this system model, we make the following assumptions:

- The CS, CP, and the clients may attempt to abuse each others' data;
- Both the CS and CP are not simultaneously compromised by attackers;
- The CS and the CP do not cooperate to access client information.

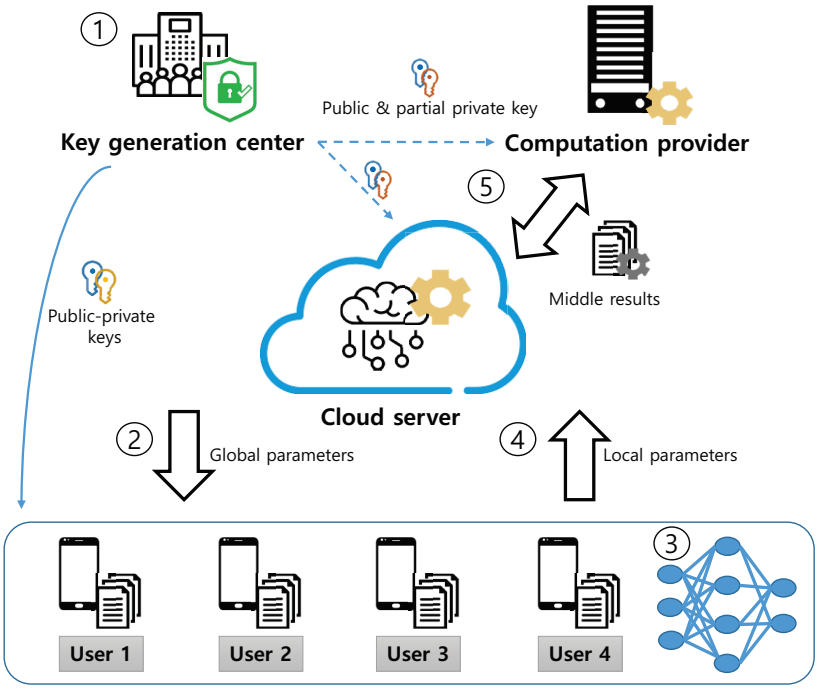

Figure 2. System model for privacy-preserving federated learning.

4.2. Attack Model

We consider several attack scenarios in the proposed system in terms of data privacy.

4.2.1. Single Malicious Entity Attack

- **Malicious clients**: Clients protect data privacy by encrypting shared parameters using the HE scheme. In the proposed system where all clients have different private keys, even if a malicious client can eavesdrop on all channels between the cloud and the clients, the malicious client cannot access the data because it cannot decrypt the ciphertexts without the corresponding private key. In addition, even if multiple malicious clients cooperate, they cannot access other clients' data without the corresponding private key;
- **Single malicious server**: The CS cannot access decrypted values in the proposed system because it only receives and handles encrypted values. For the CP, local model parameters can be accessed through combined decryption when partially decrypted values are delivered from the CS. However, the CS can prevent the CP from accessing the local model parameters by adding random noise to the encrypted local model parameters.

4.2.2. Cooperative Attacks by Multiple Malicious Entities

- **Malicious clients and CS**: If the CS cooperates with malicious clients, it can access the local model parameters of other clients because the CS can decide which clients participate in every iteration. For example, the CS can determine the list of participants with one client and the other malicious clients and calculate the average of encrypted local model parameters based on the HE scheme to obtain encrypted global model parameters. Then, the malicious clients can access the local parameters of the honest client by offsetting the local model parameter of the malicious clients by sharing their parameters because malicious clients can decrypt the global model parameter from the CS. This threat can be eliminated by ensuring more than one honest client at each iteration. In the proposed system, the threat can be eliminated by delivering the sum of local parameters through the cooperation of two or more honest clients. In addition, the privacy threat can be kept very low by ensuring the randomness of client selection through the KGC. When one honest client participates in the learning process, the conditional probability of all remaining participants being malicious clients can be expressed as $P_m^{N_c-1}$, where P_m is the ratio of the number of malicious clients to the total number of clients, and N_c is the number of clients participating in the local training. Thus, the probability of having access to local parameters of honest clients becomes very small, despite multiple malicious clients and CS collaborating. For example, even if half of all clients are malicious and N_c is 20, the probability is less than 2×10^{-6}.
- **Malicious clients and CP**: The CP cannot access the client information because random noise is added to the client information by the CS. Even if several malicious clients cooperate with the CP, the CP cannot access the local model parameters because the CS samples random noise for each client.
- **Malicious CS and CP**: When the CS and the CP cooperate to access a shared local model parameter, the client's information may be leaked. If all the distributed servers participating in the secure aggregation algorithm of SMC are compromised and cooperate with each other, there is no way to protect personal information. This paper assumes that the CS and the CP may be compromised simultaneously but do not cooperate to access client data. These assumptions are needed for building SMC-based DHC. To improve security in practice, we can increase the number of CPs participating in the secure aggregation algorithm, reducing the probability that multiple CPs and CS are malicious servers that cooperate with each other. In addition, an authentication procedure for the distributed cloud servers can be performed at the KGC to guarantee the servers participating in SMC are honest.

5. Privacy Preserving Federated Learning

In the proposed PPFL system, each client participating in the training process encrypts local model parameters trained with local data, using its own private key to protect the

trained local model parameters. Thereafter, the clients transmit the encrypted local model parameters to the CS. The CS updates the global model parameters with the local model parameters encrypted with different keys by exploiting the partial homomorphic decryption capabilities of CPs. As a result, the proposed PPFL algorithm ensures data confidentiality between the CS and the clients, as well as data confidentiality among the clients because each client has its own private key and does not send the private key to other third parties. The detailed procedure of the proposed PPFL is described in the following subsections.

5.1. Homomorphic Key Generation and Distribution

As shown in step ① of Figure 2, individual public-private key pairs (pk_i, sk_i) for $i \in \{1, \cdots, N_c\}$ are generated at the KGC and are sent to clients for encryption and decryption through secure channels, where N_c is the number of clients participating in the training process of the proposed system. Before the KGC distributes the key pairs, it performs an authentication procedure for the clients and delivers the public-private key pairs to authenticated clients. The clients' public keys for encryption and a list of authenticated clients are transmitted to the CS and the CP, and the CS utilizes only local model parameters from authenticated clients. In addition, the partially private keys $[psk_i^{(1)}, psk_i^{(2)}, \cdots, psk_i^{(K)}]$ generated by the KGC are only sent to the CS and the CPs through secure channels for cooperative decryption, respectively.

5.2. FL Local Model Training

The CS selects a machine learning model to be trained on the client's side using local data. A deep neural network model is selected; however, other machine learning models can also be used for the proposed PPFL algorithm. The CS determines the percentage of clients participating in the training process and randomly selects clients to participate in the actual training process. At the first iteration, the CS encrypts the initial global model parameters using the selected client's public keys and sends the encrypted global model parameters to the clients, as shown in step ②. In the following iterations, the CS sends the results of aggregating the local model parameters using homomorphic operations to the clients without additional encryption because the result of the homomorphic operation is also an encrypted value. The global model parameter vector encrypted with the public key of the i-th client c_i is represented as $[W_g]_i$, where W_g is a global model vector containing the global model parameters.

The i-th client decrypts the encrypted global model vector, $[W_g]_i$, using its own private key and uses the decrypted global model parameters for the local training process. Each client participating in the proposed PPFL performs the training process using the local data in a deep neural network initialized with the global model parameters, as shown in step ③. After the local training process, the i-th client obtains local model parameters and proceeds to encrypt a local model vector W_l^i containing the local model parameters using its own public key. The client sends the encrypted local model vector to the CS for secure aggregation, as described in step ④. Note that the local model vector encrypted with the public key of the i-th client is represented as $[W_l^i]_i$. After the CS receives the encrypted local model vector from the clients, the encrypted vectors are used to update the global model vector through cooperation with the CP.

5.3. Secure Global Model Update with DHC

We propose a secure averaging local model vector algorithm that updates the global model vector by calculating the average of the local model vectors received from the clients. The CS utilizes the cooperative decryption scheme to obtain the average local model vector encrypted with different public keys in the same FL-based system through cooperation with the CP, as shown in step ⑤ of Figure 2.

Figure 3 illustrates the procedure of cooperative decryption and secure local model vector updates. First, the client sends an encrypted local value $E_{pk_i}(m)$ to the CS. The CS adds the encrypted random variable $E_{pk_i}(r)$ to the received ciphertext using homomorphic

addition, where r is a random integer number, and then the CS can obtain $E_{pk_i}(m+r)$. The CS then forwards the ciphertext to the CPs. The (N_s-1) CPs perform partial decryptions, and the other CP performs the combined decryption to obtain the sum of the local value and random noise $(m+r)$. As explained in Figure 3, the CP can calculate the average of the sum of the local value and random noise $(m_{ave}+r_{ave})$ when receiving the sum from multiple clients. The sum's average is encrypted and sent back to the CS. Finally, the CS can remove the average of random values from the sum's average through homomorphic addition and obtain the encrypted average local value $E_{pk_i}(m_{ave})$ since the CS has the random values.

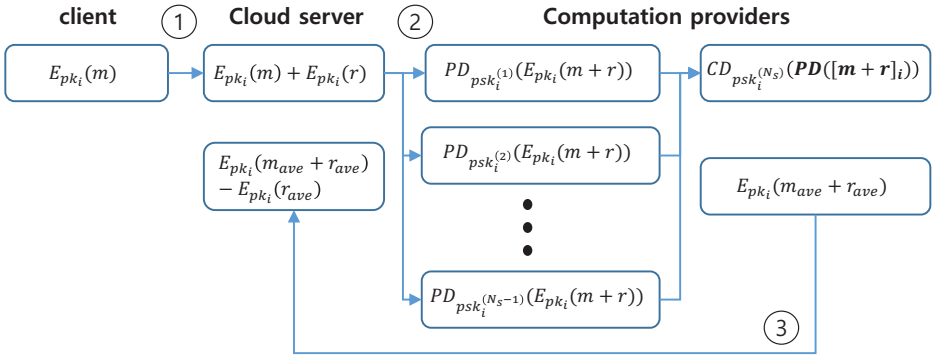

Figure 3. Diagram for secure averaging local model vector algorithm.

Algorithm 1 describes the proposed secure averaging local model algorithm where one CS and one CP exist. In the proposed algorithm, local model parameters encrypted with different keys are input, and global model parameters encrypted with different keys are output. The CS and CP have the partial private keys $psk_i^{(1)}$ and $psk_i^{(2)}$, respectively. The detailed procedure of the secure averaging local model vector algorithm is as follows:

1. The CS receives the encrypted local model vectors from the clients participating in the training process. Note that the local model vector encrypted with the public key of the i-th client is represented as $[W_l^i]_i$. Thereafter, the CS generates N_c random vectors with the same size as the local model vector and encrypts them using the client's public key, as shown in lines 2–3 of Algorithm 1;
2. The CS performs homomorphic addition operations with the encrypted local model vectors using the encrypted random vectors in line 5. The result of homomorphic addition between $[W_l^i]_i$ and $[R_i]_i$ is represented as $[S_i]_i$. Then, the CS partially decrypts the result of the homomorphic addition using the partial private key $psk_i^{(1)}$ in line 6. This process is repeated for N_c local model vectors. Subsequently, the CP sends the partially decrypted vectors $[PD^1([S_1]_1), PD^1([S_2]_2), \cdots, PD^1([S_{N_c}]_{N_c})]$ to the CP, as shown in lines 8–9;
3. As shown in lines 10–11, the CP partially decrypts the partially decrypted vectors using the partial private key $psk_i^{(2)}$ and obtains $[(W_l^1+R_1), \cdots, (W_l^{N_c}+R_{N_c})]$. Afterwards, the CP adds all the decrypted vectors and divides the sum by the number of clients N_c to obtain a vector containing the average parameters in line 12. The result is represented as W_{sum};
4. The CP encrypts W_{sum} using the public keys of the N_c clients in lines 13–15 and sends the encrypted vectors $[[W_{sum}]_1, \quad [W_{sum}]_2, \cdots, [W_{sum}]_{N_c}]$ to the CS in line 16;
5. Finally, the CS calculates the encrypted average global model vectors for the clients by performing the homomorphic addition operation with the encrypted sum of random noises $[\frac{\sum_k -R_k}{N_c}]_i$, as shown in lines 17–19.

Algorithm 1 Secure averaging local model algorithm

Input: $[W_l^1]_1, [W_l^2]_2, \ldots, [W_l^{N_c}]_{N_c}$.
(@CS) generates N_c random vectors $R_1, R_2, \ldots, R_{N_c}$ and encrypts the random vectors
using the public keys.
for $i \leq N_c$ **do**
    (@CS) $[S_i]_i \leftarrow [W_l^i]_i \cdot [R_i]_i$.
    (@CS) Partially decrypts $[S_i]_i$ using $psk_i^{(1)}$
end for
(@CS) Sends partially decrypted vectors $[PD^1([S_1]_1), PD^1([S_2]_2), \ldots, PD^1([S_{N_c}]_{N_c})]$
to the CP.
(@CP) Partially decrypts the partially decrypted values using $psk_i^{(2)}$ and obtains
$[(W_l^1 + R_1), \ldots, (W_l^{N_c} + R_{N_c})]$.
(@CP) Calculates $W_{sum} = \frac{\sum_k W_l^k + \sum_k R_k}{N_c}$.
for $i \leq N_c$ **do**
    (@CP) Encrypts W_{sum} using the public key pk_i.
end for
(@CP) Sends the encrypted values $[[W_{sum}]_1, [W_{sum}]_2, \ldots, [W_{sum}]_{N_c}]$ to the CS.
for $i \leq N_c$ **do**
    (@CS) $[W_g]_i \leftarrow [W_{sum}]_i \cdot [\frac{\sum_k -R_k}{N_c}]_i$.
end for
Output: global weight vectors for the clients $[[W_g]_1, [W_g]_2, \ldots, [W_g]_{N_c}]$.

After updating the global model vector by performing the proposed secure averaging local model vector algorithm, the CP sends the updated global model vector to the clients for the next federated round. The clients execute the local training process using the updated global model vector as shown in Section 5.2 after decrypting the encrypted global model vector using its own private key. Thereafter, they send the newly trained local model vector to the CS, and then the CS and CP work together to update the global model vector. These procedures are repeated until convergence is achieved.

5.4. Data Structure and Protocol

The HE scheme increases data security but has the disadvantage of incurring communication overhead. Especially since the data length after encryption is independent of the plaintext length to be encrypted, the communication efficiency is significantly reduced if only one parameter is encrypted and sent. The proposed system establishes a data structure that can transfer multiple parameters to alleviate this efficiency degradation. The data structure for a weight w_i consists of a bit representing the sign, a zero bit to prevent an overflow caused by homomorphic additions, and the remaining bits signifying the weight's value. The number of weights included in one ciphertext can be calculated as $D = \left\lfloor \frac{K}{L_o} \right\rfloor$, where $\lfloor \cdot \rfloor$ is a round-down operation and L_o is the data length used for representing weights. Then, the data format to be encrypted can be expressed as follows: $[w_D^{Lo \cdot (D-1)}, w_{D-1}^{Lo \cdot (D-2)}, \cdots, w_1]$. Furthermore, since the secure aggregation operations are performed in the plaintext space except for the process of adding noise, the proposed algorithm can be implemented using only homomorphic operations for integer processing. Therefore, in the proposed system, integer numbers are used for data transmission, and they are represented as floating-point numbers using a decimal point pre-agreed between the clients and the servers after decryption.

6. Performance Analysis

6.1. Computational Overhead

6.1.1. Computational Overhead on the Client's Side

In the proposed PPFL algorithm, additional encryption operations are performed to protect the trained local model parameters, and extra decryption operations are performed

to reflect the global model parameters to the learning model on the client's side. In the PCK scheme used in the proposed algorithm, the exponentiation operation has a dominant effect on encryption and decryption. The exponentiation operation g^r requires $1.5 \times N_r$ multiplications, where g is a generator of order $(p-1)(q-1)/2$, $r \in \mathbb{Z}_K$ is a random number, and N_r is the length of r in the DHC scheme in [21]. Thus, the computational complexity of the encryption operation in the proposed PPFL algorithm is given as $\mathcal{O}(N_r \cdot N_w)$, where N_w is the number of elements of the local model vector. Similarly, the computational complexity of the decryption in the proposed PPFL algorithm is also represented as $\mathcal{O}(N_r \cdot N_w)$.

6.1.2. Computational Overhead on the Server's Side

The computational complexity of the averaging local model parameter algorithm performed in the conventional FL algorithm can be expressed as $\mathcal{O}(N_w \cdot N_c)$. On the other hand, in the proposed PPFL algorithm, additional encryption, partial decryption, and homomorphic addition operations are required to perform the proposed secure averaging local model vector algorithm on the server's side. The encryption and partial decryption operations have a dominant impact on the computational complexity because the exponentiation operation in the encryption and partial decryption requires much more computation than the other operations. Moreover, as the number of clients and model parameters increases, the number of encryption and partial decryption operations to be performed also grows. Thus, the computational complexity of the secure averaging local model vector algorithm on the server's side can be represented as $\mathcal{O}(N_r \cdot N_w \cdot N_c)$.

6.2. Communication Overhead

6.2.1. Communication Overhead between Clients and the Cloud Server

In an FL scenario involving many clients, the communication overhead has a tremendous impact on performance. If a cryptosystem is utilized to preserve data privacy in an FL-based system, the communication overhead may be more significant than sending local parameters as a plaintext. In this paper's cryptosystem, the length of the ciphertext is affected only by the key size, regardless of the length of data the client sends to the server, and the length of data must be less than the key size. Thus, the closer the data length is to the key size, the less communication overhead is incurred because more information can be conveyed in one ciphertext. In the cryptosystem, since encryption requires a modular operation with a dividend K^2, the length of the ciphertext becomes $K \times 2$ bits. When the key size is K bits and the length of data to be transmitted is L_d bits, the transmission data volume after encryption becomes $2N/L_d$ times larger. As L_d is closer to K, the transmission data volume is approximately doubled. In the proposed system, K-bit data representing multiple local model parameters are generated to reduce the communication overhead.

6.2.2. Communication Overhead between the Cloud Server and the Computation Provider

In order to perform the proposed secure aggregation operation for local model vectors, we exploit the partial homomorphic decryption capabilities of CPs. The CS sends partially decrypted ciphertexts to the CP and receives ciphertexts from the CP in the proposed PPFL-based system. The length of the partially decrypted ciphertext is also $2 \times K$ bits in [21]. As shown in Algorithm 1, the amount of information communicated between the CS and the CP increases as the number of clients and model parameters increases. Thus, the communication overhead between the CS and the CP can be represented as $\mathcal{O}(N_c \cdot N_w)$ bits.

6.3. Overhead Comparison

Compared with the PPDL system in [2] that used the HE scheme, the proposed technique requires additional overhead to allow clients to use different private keys. In the proposed system, a certain degree of computational overhead for cooperative decryption and encryption processes is added, and the communication overhead is also added for the data exchanges between the CS and the CPs. The computational and communication overhead analysis was performed in Sections 6.1.2 and 6.2.2, respectively. The computa-

tional overhead comparison between the PPDL system and the proposed system is shown in Section 7.1, and the communication overhead between the CS and the CP is shown in Table 1.

6.4. Security Analysis

In the proposed algorithm, since model data is protected by the cryptosystem, attackers cannot access the data even if data are stolen from a communication network. Therefore, the attackers must break the cryptosystem to access local model data. Even in the case of insider attacks, attackers have to break the cryptosystem to access the data because clients have access only to global models and their local models, and servers have access only to encrypted data. The HE scheme can have a higher security level of cryptosystem if the key size increases. As the key size increases, the amount of computation required to break the encryption algorithm or system also increases. For example, it was shown that if the key sizes are 1024, 2048, 3072, 7680, and 15,360 bits in Paillier's cryptosystem-based HE scheme, the security level is given as 80, 112, 128, 192, and 256-bit, respectively in [24]. However, there exists a trade-off relationship between the security level and computation/communication overhead because the amount of encryption and decryption computation and the data length of the ciphertexts also increase. Numerical evaluations of computation and communication overheads with respect to the key size have been performed in Section 7.

With a higher security level of the cryptosystem, the proposed scheme can more robustly resist the attacks described in Section 4.2.

- In the attack model of malicious clients, malicious clients can eavesdrop on the communication channels, and obtain ciphertexts and partially decrypted ciphertexts. Because the DHC is semantically secure, as described in [25], an attacker has to break the cryptosystem to obtain private data. If an honest client uses a longer key size, the attacker will have to use more computational resources to break the victim's cryptosystem.
- In the attack model of a single malicious server, malicious servers can eavesdrop on the communication channels, and obtain ciphertexts and partially decrypted ciphertexts. As in the client attack model, an attacker must break the cryptosystem to obtain private data. Even though CP acquires the model parameters and performs the combined decryption to obtain the plaintext, it cannot access private data because the plaintext includes a random noise added by the CS.
- In the attack model of malicious clients and servers, the malicious entities can cooperate; the malicious client may provide a private key to the malicious server. If all clients use the same private key, the malicious server can access all clients' private data because a malicious client can provide the private key to the malicious server. On the other hand, since the proposed system allows clients to use different private keys in the same FL-based system, the privacy leakage can be prevented in this attack model.

Based on the observation of the attack models, it is worth noting that the proposed system provides a much stronger level of security than the state-of-the-art system proposed in [2] where all clients use the same private key. Even though a client's cryptosystem is broken, the data privacy of the other clients in the proposed system is not affected by the compromised cryptosystem of the victim client because the clients use different private keys. Suppose that the amount of computational resources required to break the cryptosystem of the i-th client is $C_b^i(K)$, where K is the key size. If all clients use the same private key, the computational resource amount to break the system is expressed as $\min\{C_b^1(K), C_b^2(K), \ldots, C_b^{N_c}(K)\}$. As a result, in this case, if the most vulnerable client is broken in, the entire system can be easily compromised. On the other hand, the computational resource amount to break the proposed system is given by $\sum_i^{N_c} C_b^i(K)$. As the number of clients increases, the amount of computational resources needed to attack the system increases linearly.

7. Performance Evaluation

In this section, we have developed the proposed algorithm using Python and evaluated the performance on a workstation (3.6 GHz quad-core processor and 8 GB RAM) in terms of computation and communication overhead.

7.1. Computational Overhead

Figure 4 shows the running time measured for performing encryption and decryption according to the key size. In our simulation environment, the key sizes were selected as 1024, 2048, 3072, 7680, and 15,360 bits to achieve 80, 112, 128, 192, and 256-bit security levels, respectively. For example, the encryption took 11.7, 78.4, and 1552.5 ms for 80, 128, and 265 bit security, respectively, in Figure 4. As the key size increases, the running times for encryption and decryption increase exponentially because the exponent of the exponentiation operation in encryption and decryption increases.

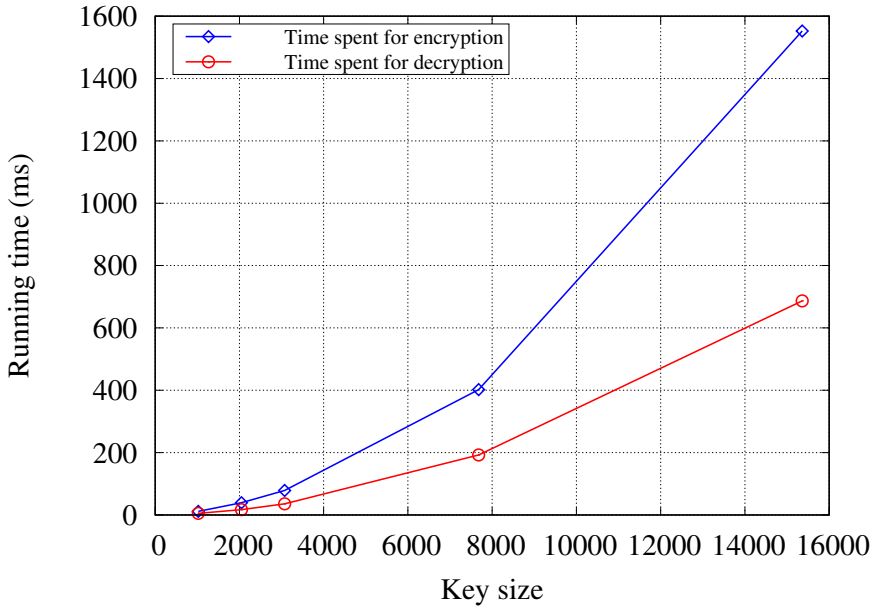

Figure 4. Running time to execute homomorphic encryption and decryption with respect to the key size.

Figure 5 shows the running times measured for performing the proposed secure averaging local model algorithm in Algorithm 1 with respect to the number of clients in the cryptosystem with different key sizes. The convolutional neural network with 105,506 parameters was used for the simulation study, and the data length used for representing weights was set to 16 bits. As the number of clients increases, the number of homomorphic operations increases as the number of parameters to protect using the HE scheme increases, and thus the running time increases linearly. In Figure 5, the running time increases as the key size increases. In fact, if the key size is larger, the total number of ciphertexts to be delivered is smaller. However, as shown in Figure 4, the running time of homomorphic operations increases exponentially as the key size increases. As a result, the total running time of Algorithm 1 increases as the key size increases. Nevertheless, as the key size increases, the security level of the system increases. This is because the higher the key size, the greater the number of cases is required to break the cryptosystem. Thus, because the computational burden and security gain have a trade-off relationship, we can select an appropriate key size according to system requirements.

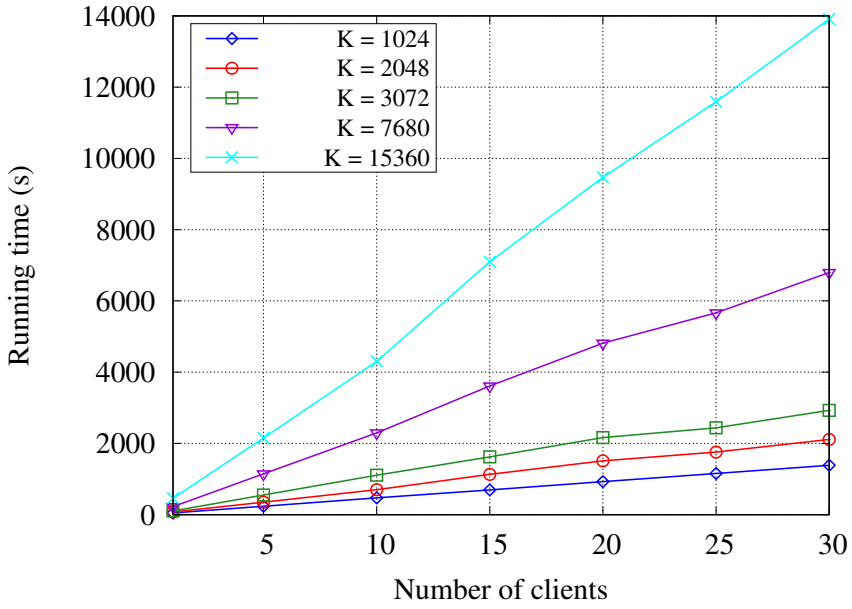

Figure 5. Running time to execute the proposed secure averaging local model algorithm with respect to the number of clients in the cryptosystem with different key sizes.

Figure 6 shows the running time for performing the proposed algorithm with respect to the neural network size of the federated learning. The number of clients was set to 10, and the data length used for representing weights was set to 16 bits for the simulation study. As the number of parameters increases, the running time increases because the amount of information to be processed by the homomorphic operation increases.

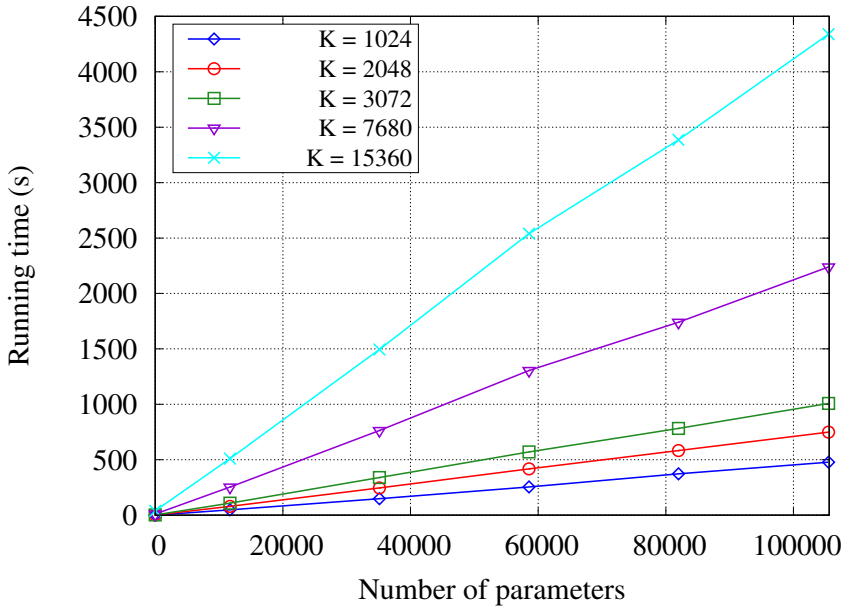

Figure 6. Running time to execute the proposed secure averaging local model algorithm with respect to the neural network size in the cryptosystem with different key sizes.

We performed simulations to compare the computational overhead of the proposed PPFL system and the PPDL system proposed in [2]. The key size K is 1024, and the number of parameters is 105,506 in the simulation environment. In Figure 7, it is seen that the computation overhead of the proposed system is about 2.3 times greater than that of the PPDL system. This is because the additional encryption and partial decryption

processes are performed at the servers to make clients have different keys in the proposed system. Despite the greater computational overhead of the proposed algorithm, the security intensity of FL systems is significantly improved because the clients use different private keys. Therefore, the proposed system can be deployed in a more adversarial environment where there exist many malicious clients and they are difficult to be identified. In future work, we will research how to reduce the overhead in PPFL while retaining the same strong security level.

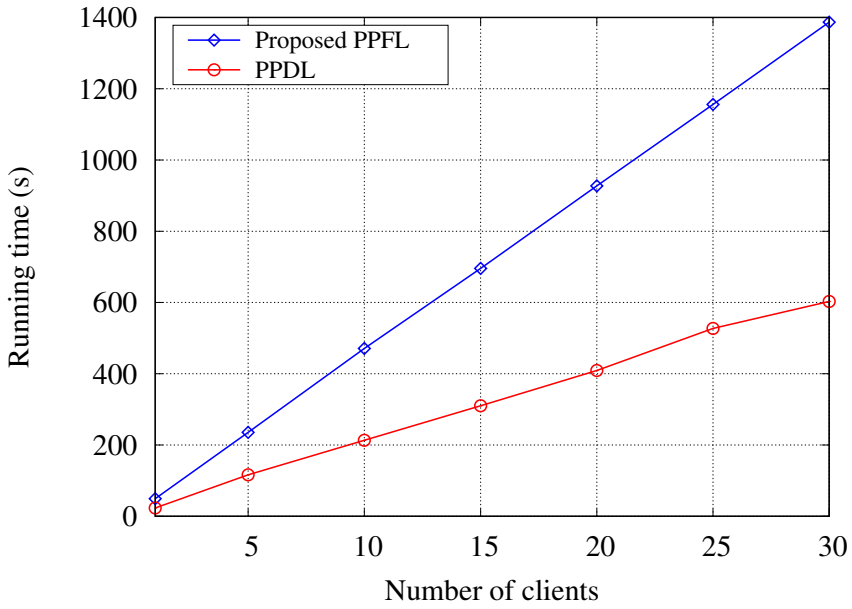

Figure 7. Running time to execute the proposed PPFL and the PPDL using the Paillier cryptosystem with respect to the number of clients.

7.2. Communication Overhead

The communication overhead increases in the proposed PPFL algorithm because the servers and clients communicate with each other using the encrypted model vectors to protect the model parameters. Table 1 shows the communication overhead with respect to the key size when the number of parameters is 105,506, and the data length for one parameter is 16 bits. The communication overhead remains almost constant regardless of the key size as shown in Table 1. If the key size increases, the ciphertext length may become longer, but since the number of parameters included in the ciphertext increases, the key size has little effect on the communication overhead of the proposed algorithm. In addition, as the number of clients increases, the communication overhead linearly increases because the amount of data exchanged between CS and CP increases.

Table 1. Communication overhead (KB).

K (Key Size)	1024	3072	15,360
Client–CS	422.1	422.4	422.4
CS–CP ($N_c = 10$)	8443	8448	8448
CS–CP ($N_c = 100$)	84,429	84,480	84,480

8. Conclusions and Future Work

This paper has proposed the PPFL system based on the HE scheme to protect shared model parameters in an FL-based system. Furthermore, we have proposed a technique for the secure aggregation of local model parameters encrypted with different keys in the same FL-based system. In the proposed system, the computational and communication costs

required to improve security level in FL were theoretically analyzed, and the performance of the proposed PPFL algorithm in terms of overhead was evaluated via simulations. In the future, our research focuses on how to further reduce the computation and communication costs in the proposed PPFL algorithm while retaining privacy preservation of clients, and also focuses on how to determine an appropriate number of clients participating in FL to expedite the learning and to reduce latency of FL-based services.

Author Contributions: Conceptualization, J.P. and H.L.; methodology, J.P.; investigation, J.P.; formal analysis, J.P. and H.L.; validation, J.P. and H.L.; writing—original draft preparation, J.P.; writing—review and editing, H.L.; and supervision, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00379, Privacy risk analysis and response technology development for AI systems).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
2. Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 1333–1345.
3. Hitaj, B.; Ateniese, G.; Perez-Cruz, F. Deep models under the GAN: Information leakage from collaborative deep learning. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 603–618.
4. Zhang, C.; Xie, Y.; Bai, H.; Yu, B.; Li, W.; Gao, Y. A survey on federated learning. *Knowl.-Based Syst.* **2021**, *216*, 106775. [CrossRef]
5. Khan, L.U.; Saad, W.; Han, Z.; Hossain, E.; Hong, C.S. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1759–1799. [CrossRef]
6. Lin, J.C.W.; Srivastava, G.; Zhang, Y.; Djenouri, Y.; Aloqaily, M. Privacy-preserving multiobjective sanitization model in 6G IoT environments. *IEEE Internet Things J.* **2020**, *8*, 5340–5349. [CrossRef]
7. Li, T.; Hu, S.; Beirami, A.; Smith, V. Ditto: Fair and robust federated learning through personalization. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 6357–6368.
8. Lin, F.P.C.; Hosseinalipour, S.; Azam, S.S.; Brinton, C.G.; Michelusi, N. Semi-decentralized federated learning with cooperative D2D local model aggregations. *IEEE J. Sel. Areas Commun.* **2021**, *in press*. [CrossRef]
9. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **2019**, *10*, 1–19. [CrossRef]
10. Ou, W.; Zeng, J.; Guo, Z.; Yan, W.; Liu, D.; Fuentes, S. A homomorphic-encryption-based vertical federated learning scheme for rick management. *Comput. Sci. Inf. Syst.* **2020**, *17*, 819–834. [CrossRef]
11. Zhang, C.; Li, S.; Xia, J.; Wang, W.; Yan, F.; Liu, Y. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In Proceedings of the 2020 USENIX Annual Technical Conference (USENIXATC 20), Online, 15–17 July 2020; pp. 493–506.
12. Cheng, K.; Fan, T.; Jin, Y.; Liu, Y.; Chen, T.; Papadopoulos, D.; Yang, Q. Secureboost: A lossless federated learning framework. *IEEE Intell. Syst.* **2021**, *in press*. [CrossRef]
13. Shokri, R.; Shmatikov, V. Privacy-preserving deep learning. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1310–1321.
14. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1175–1191.
15. Xu, G.; Li, H.; Liu, S.; Yang, K.; Lin, X. Verifynet: Secure and verifiable federated learning. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 911–926. [CrossRef]
16. Fang, H.; Qian, Q. Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning. *Future Internet* **2021**, *13*, 94. [CrossRef]
17. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
18. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [CrossRef]

19. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Singapore, 14–18 November 1999; pp. 223–238.
20. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
21. Liu, X.; Deng, R.H.; Choo, K.K.R.; Weng, J. An efficient privacy-preserving outsourced calculation toolkit with multiple keys. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 2401–2414. [CrossRef]
22. Katz, J.; Lindell, Y. *Introduction to Modern Cryptography*; CRC Press: Boca Raton, FL, USA, 2020.
23. Liu, X.; Choo, K.K.R.; Deng, R.H.; Lu, R.; Weng, J. Efficient and privacy-preserving outsourced calculation of rational numbers. *IEEE Trans. Dependable Secur. Comput.* **2018**, *15*, 27–39. [CrossRef]
24. Barker, E.; Barker, E.; Burr, W.; Polk, W.; Smid, M. *Recommendation for Key Management: Part 1: General*; National Institute of Standards and Technology, Technology Administration: Gaithersburg, MD, USA, 2006.
25. Bresson, E.; Catalano, D.; Pointcheval, D. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, 30 November–4 December 2003; Springer: Berlin/Heidelberg, Germany, 2003; pp. 37–54.