Securing a Local Training Dataset Size in Federated Learning

YOUNG AH SHIN\textsuperscript{1}, GEONTAE NOH\textsuperscript{2}, IK RAE JEONG\textsuperscript{1}, AND JI YOUNG CHUN\textsuperscript{2}

\textsuperscript{1}Graduate School of Information Security, Korea University, Seoul 02841, South Korea
\textsuperscript{2}Department of Big Data and Information Security, Seoul Cyber University, Seoul 01133, South Korea

Corresponding authors: Ji Young Chun (jychun@iscu.ac.kr) and Ik Rae Jeong (irjeong@korea.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) Grant through the Korea Government (MSIT) under Grant 2021R1F1A1063992.

\textbf{ABSTRACT} Federated learning (FL) is an emerging paradigm that helps to train a global machine learning (ML) model by utilizing decentralized data among clients without sharing them. Although FL is a more secure way of model training than a general ML, industries where training data are primarily personal information, such as MRI images or Electronic Health Records (EHR), should be more precautious of privacy and security issues when using FL. For example, unbalanced dataset sizes may denote some meaningful information that can lead to security vulnerabilities even if the training data of the clients are not exposed. In this paper, we present a Privacy-Preserving Federated Averaging (PP − FedAvg) protocol specialized for healthcare settings to limit user data privacy leakage in FL. We particularly protect the size of datasets as well as the aggregated local update parameters by securely computing among clients based on homomorphic encryption. This approach ensures that the server does not access the size of datasets and local update parameters while updating the global model. Our protocol has the advantage of protecting the size of datasets when datasets are not uniformly distributed among clients and when some clients drop out each iteration.

\textbf{INDEX TERMS} Federated learning, homomorphic encryption, privacy-preserving, training dataset.

\section{I. INTRODUCTION}
Federated learning (FL) [1] has appeared in the realm of machine learning (ML) according to the increasing usage of mobile devices. This wealth of information and the power of computation in devices turned into a suitable environment for model training. In mobile keyboard prediction model [2], for instance, application users can easily choose the next word by clicking the recommendation presented on the screen instead of typing every word. Unlike the standard ML, FL ensures user data privacy by training a shared global model of a central server under coordination of distributed devices [3]. This approach is considered as privacy-preserving ML such that each mobile device can prevent its data leakage by not sending all the training data, but only a local update to the server.

While FL mitigates privacy risks from the centralized ML, yet it provides no theoretical guarantee of privacy [4]. This problem is critical in situations where local data are privacy-sensitive enough to be monitored by laws, e.g. healthcare [5]. Although it may not seem valuable, such information as training dataset sizes can turn into critical information for a malicious attacker when they are publicly exposed. Currently, there are many research areas that make efforts in privacy-preserving FL such as secure multi-party computation (SMC) [6], differential privacy (DP) [7] and homomorphic encryption (HE) [8]. However, existing FL systems are still facing some practical challenges, compatibility of efficiency and privacy. In FL, hundreds of rounds of communication are needed to train a global model and the size of updates to be sent is large. It is hard to find suitable privacy-preserving techniques while fulfilling the efficiency requirement at the same time.

In this work, we present a fundamental Privacy-Preserving Federated Averaging (PP − FedAvg) protocol that secures a
training dataset size from an aggregate server. Compared to previous studies, we consider the importance of protecting the size of datasets which can be overlooked when FL is applied to the medical environment. We specifically focus on the medical environment because it is expected that the meaning of the dataset sizes can be crucial in such contexts. To the best of our knowledge, our PP – FedAvg protocol firstly tackles the problem of confidentiality in the sizes of training datasets specialized for healthcare industries. PP – FedAvg realizes the privacy of the clients’ updated results and the training dataset sizes.

In the protocol, we mainly utilize partially homomorphic encryption, especially Paillier’s cryptosystem, to ensure the privacy of the user data while enabling the operation of addition on the ciphertext. Compared to other cryptographic algorithms, homomorphic encryption provides an advantage to guarantee the data confidentiality, even though it may bring out certain computational inefficiency. To overcome this issue, we state that PP – FedAvg assumes cross-silo settings where each client is an institution that has more flexibility of computation resources [4].

When each client finishes training through its local training datasets and updating its model parameters after iteration, the clients interact with each other through our protocol to mask the calculated results: sum of local update parameters and the sum of local training datasets, instead of sending each result directly to the server to aggregate. This system is essential in healthcare industries where the number of datasets implies sensitive information. The information can denote the number of patients in the local hospital, rare diseases among the regions, etc. We propose that our protocol is more enhanced in privacy-preserving since we contemplate the leakage of each client and overall local update vectors along with the size of datasets that the server inevitably uses when aggregating the results.

The rest of the paper organizes as follows. In Section II, we explain the basic preliminaries, then we summarize the previous works related to our protocol in Section III. In Section IV, we emphasize the need to protect the size of datasets in medical environments. After presenting an applicable scenario and the proposed scheme in Section V, we show analysis on security and efficiency in Section VI. In Section VII, we discuss some of the remarks and future works. Lastly, we conclude in the Section VIII.

II. PRELIMINARIES

In this section, we briefly discuss the main concept of the federated learning and the underlying cryptographic primitive of our construction.

A. FEDERATED LEARNING

We describe a general FedAvg algorithm [1] in Table 1 that is used in our proposed protocol. In the algorithm, $k$ is the number of clients, $p$ is the ratio of clients that is chosen for each round, $b$ is the local mini-batch size, $t$ is the amount of local epochs, and $\alpha$ is the learning rate. Each client $i$ ($1 \leq i \leq k$) has its own dataset and the index of the dataset on the client $i$ is $D_i$. Let the number of the elements in $D_i$ be $n_i$ and $n = \sum_{i=1}^{k} n_i$. $\ell(\omega; d)$ is the loss of the prediction on the batch $d$ made with the model parameters $\omega$.

At the beginning phase, the server initializes the global model parameters and sends the current parameters to the clients in a secure manner. For example, clients can verify the received parameters by conducting public key encryption with the server. In our protocol, this can be conveniently done with the help of the Key Generation Center (KGC). Afterwards, each client computes the updated local model parameters with the global model parameters and its own dataset, then sends the computed values to the server. The server updates the global model parameters using the received values from clients, and iterates the process repeatedly.

Our proposed protocol proceeds after each client finishes computing the local update during the iterations. Before sending the updates to the server, we suggest an additional communication among the participating clients in order to mask the sum of local updates and the total sizes of training datasets. Most of the previous works assume that the size of datasets is fixed. Nevertheless, FL in the real world does not set the total amount of datasets as it may naturally increase within local devices every time the service is used.

| TABLE 1. FedAvg protocol. |
|----------------------------|
| **Server:**               |
| initialized model parameters, $\omega_1$ |
| for each round $j = 1, 2, 3, \ldots$ do |
| $m \leftarrow \max(\text{round}(p \cdot k), 1)$ |
| $S_j \leftarrow$ random set of $m$ clients |
| for each client $i \in S_j$ simultaneously do |
| $\omega_{j+1} \leftarrow \text{Client}(i, \omega_j)$ |
| $\omega_{j+1} \leftarrow \sum_{i=1}^{m} \frac{n_i}{n} \omega_{j+1}$ |
| **Client(i, \omega):**    |
| $D \leftarrow$ split $D_i$ into batches of size $b$ |
| for each local epoch $i = 1, 2, 3, \ldots, t$ do |
| for batch $d \in D$ do     |
| $\omega \leftarrow \omega - \alpha \nabla \ell(\omega; d)$ |
| return $\omega$ to server  |

B. HOMOMORPHIC ENCRYPTION

Homomorphic Encryption (HE) is a type of public-key encryption scheme that allows computation on ciphertexts without decryption. It provides the same result as encrypting after operating on messages. Generally, HE is classified into three types of schemes according to the number of possible operations [9]: Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SWHE), and Fully Homomorphic Encryption (FHE). While PHE only enables one operation between addition and multiplication operations, FHE realizes multiple types of operations on the ciphertexts. SWHE also achieves both operations like FHE, but the number of operations is limited.
HE is widely used in the field where data utilization is inevitable and also data confidentiality must be considered as well. According to Phong et al. [10] work, it was shown that it is possible to recover original data from the leakage information of gradients in deep learning. Many HE approaches in the FL literature [11], [12], [13], [14], [15], [16], [17], [18] have emerged to protect the model parameters in order to ensure user data privacy in cross-silo settings.

Generally, PHE is more widely preferred rather than FHE for the reason that computation costs of multiplication are significantly expensive. In our proposed system, we consider Paillier’s cryptosystem [8], a PHE that allows the addition of ciphertexts through multiplications of a random plaintext number. Therefore, we determine that Paillier’s cryptosystem is an adequate solution for our protocol.

Paillier’s HE mainly consists of five algorithms $HE = P(HE.Gen, HE.Enc, HE.Dec, HE.Add, HE.Mul)$ with a message space $\mathcal{M}$. Given that $c = HE.Enc_{pk}(m)$ for a public key pair $(pk, sk)$ is $HE.Gen(1^k)$ and $HE.Dec_{sk}(c) = HE.Gen(1^k) = m$. Given that $c_1 = HE.Enc_{pk}(m_1)$ and $c_2 = HE.Enc_{pk}(m_2)$ for a public key pair $(pk, sk)$ is $HE.Gen(1^k)$ and a constant $k$, $HE.Add_{pk}(c_1, c_2) = HE.Enc_{pk}(m_1 + m_2)$ (1)

$HE.Mul_{pk}(k, c_2) = HE.Enc_{pk}(k \cdot m_2)$ (2)

holds for every message $m_1, m_2 \in \mathcal{M}$ without decrypting any ciphertext. For instance, for a ciphertext $c_2$ and a constant value 3, $HE.Mul_{pk}(3, c_2)$ is equal to $HE.Add_{pk}(c_2, HE.Add_{pk}(c_2, c_2))$.

To understand the estimated computational and timing costs of Paillier’s HE in FL, we refer to BatchCrypt’s measurements [14] as shown in Table 2. The time consumption of encrypting the model updates and the size of the encrypted message may be quite a burden for a server to aggregate all the client’s encrypted gradients. Therefore, we show the experimental results of python-paillier [19] implemented by BatchCrypt as a baseline to measure the trade-off between efficiency and security. For both cases of key sizes in 2048 and 3072 bits, the message size is 6.87MB, consisting of 900,000 32-bit floating-point values. The library involves the transformation of floating-point numbers into integers.

| TABLE 2. The computational and timing costs measured by BatchCrypt [14] with python-paillier. |
|---|
| Key Size (bit) | Ciphertext Size (MB) | Encryption Time (min) | Decryption Time (min) |
| 2048 | 527.17 | 19.216 | 5.952 |
| 3072 | 754.62 | 51.852 | 16.563 |

As seen in Table 2, it shows that there would be an additional amount of computation and time consumption for the participants in order to gain privacy on both of each local gradient and dataset size. We state that the trade-off is achievable and effective by the reason that our protocol has no loss of accuracy and is also adequate for cross-silo settings, where the security guarantee is highly concerned.

III. PREVIOUS APPROACHES IN PRIVACY PRESERVING FEDERATED LEARNING

In this section, we briefly review the related works and compare them with our protocol, PP – FedAvg, in the aspect of privacy on dataset sizes. Recently, many mechanisms have been proposed to enhance privacy preservation in FL, specifically to prevent the leakage of clients’ updates when communicating with the central server. In 2017, Bonawitz et al. [20] firstly introduced a practical Secure Aggregation (SA), a four-round protocol with threshold secret sharing and Diffie-Hellman key agreement, which ensures the central server to aggregate users’ updates but does not inspect the individual updates. Since then, many research works including SAFELearn [16], VerifyNet [21], VeriFL [22] devoted to demonstrating a secure aggregation protocol. Xu et al. [23] presented HybridAlpha to realize efficiency and resilience while utilizing Secure Multi-party Computation (SMC) in FL techniques.

Meanwhile, the SA protocol can be combined with homomorphic encryption (HE) to enable a server to perform certain arithmetic operations on encrypted local model updates. While HE provides a privacy guarantee to the encrypted data, there exist several considerations to implement in FL. For example, Paillier’s HE, the HE that utilized in our proposed system, requires more complex and expensive operations for modular multiplications and exponentiation [18]. In addition, it also needs modifications to handle floating point values since it only enables encrypting integers.

Nevertheless, much studies [11], [12], [13], [14], [17], [18] over the recent years have worked on solving the problem of efficiency degradation in HE. In 2017, Hardy et al. [11] proposed a logistic regression model using HE to preserve user data privacy in vertical FL settings. In 2018, Truex et al. [12] presented a method to prevent possible attacks on the transmitted message by combining MPC and DP. In 2019, Zhang et al. [13] proposed PEFL, which used Paillier’s cryptosystem to protect local gradients from the honest-but-curious server. In 2020, Batchcrypt [14] proposed an efficient cross-silo FL with Paillier’s HE that encodes a batch of quantized gradients into long integers to reduce the computation overhead.

In 2021, Ma et al. [15] improved the original multi-key HE protocol to prevent model updates leakage, enabling clients to cooperate in order to encrypt or decrypt the encrypted updates. In the same year, Cheng et al. [17] presented SecureBoost that realizes no loss of accuracy from the regular FL method while devising the gradient tree boosting algorithm. In their scheme, they also utilized HE to securely sum up the possible splits as well as local model updates in vertical FL. Park et al. [18] also proposed a privacy-preserving FL algorithm with HE by using different private keys for each node in the system. Although the experimental results show
TABLE 3. Comparison of our protocol to previous works.

| Schemes                  | Year | Cryptographic Techniques | Adversary Model | Protecting Object | FL Framework |
|--------------------------|------|--------------------------|-----------------|-------------------|--------------|
| Bonawitz et al. [20]     | 2017 | SS, KA, AE               | HBC, AA         | ✓                 | x            | horizontal   |
| Hardy et al. [11]        | 2017 | AHE (PA)                 | HBC             | ✓                 | x            | vertical     |
| Truek et al. [12]        | 2018 | AHE(TP), DP, SMC         | AA              | ✓                 | x            | horizontal   |
| HybridAlpha [23]         | 2019 | SMC                      | AA              | ✓                 | x            | horizontal   |
| PEFL [13]                | 2019 | AHE (PA)                 | Honest           | ✓                 | x            | horizontal   |
| BatchCrypt [14]          | 2020 | AHE (PA)                 | Honest           | ✓                 | x            | horizontal   |
| VerifyNet [21]           | 2021 | SS, KA, HH               | HBC             | ✓                 | x            | horizontal   |
| Ma et al. [15]           | 2021 | MKHE                     | HBC             | ✓                 | x            | horizontal   |
| VeriFL [22]              | 2021 | SS, KA, HH, COM          | HBC             | ✓                 | x            | horizontal   |
| SAFELearn [16]           | 2021 | FHE, SMC                 | Honest           | ✓                 | x            | horizontal   |
| SecureBoost [17]         | 2021 | AHE (PA)                 | HBC             | ✓                 | x            | vertical     |
| Park et al. [18]         | 2022 | AHE                      | AA              | ✓                 | x            | horizontal   |
| PP - FedAvg (Ours)       | 2022 | AHE (PA)                 | HBC             | ✓                 | ✓            | horizontal   |

For cryptographic techniques, SS denotes secret sharing with key agreement, KA denotes key agreement protocol, and AE denotes authenticated encryption. AHE(·) denotes additively homomorphic encryption and its specific protocol type used in the scheme. PA denotes Paillier’s HE and TP denotes threshold Paillier’s HE. DP denotes differential privacy. SMC denotes secure multi-party computation. HH denotes homomorphic hash function and MKH denotes multi-key HE. For adversary model, HBC represents honest-but-curious adversary in its threat model. AA denotes active adversaries meaning dishonest adversaries in the system architecture. Honest denotes trusted participants that follow the overall process in the system.

that there still need more improvements in computational overhead, we note that HE has the potential for achieving privacy preservation.

Table 3 shows an overview of the previous works that focused on the leakage of information sent by clients. Each scheme is analyzed by the main cryptographic techniques, the adversary models of each entity, the objects that are protected, and the specific FL framework on which it relies. Although most of the systems are able to protect the dataset sizes as well as the local gradients through their schemes by the same way as protecting the local gradients, we find that they did not recognize the necessity of securing the dataset sizes in their protocols. In our scheme, we protect each and overall sizes of datasets while also protecting each and overall local gradients from the server. The server only gets to access the global update after eliminating the random value in each ciphertexts.

IV. SIGNIFICANCE OF PROTECTING DATASET SIZES IN HEALTHCARE ENVIRONMENTS

In this section, we emphasize the need to preserve the privacy of local and overall dataset sizes and exemplify what dataset sizes connote in healthcare settings. As shown in Table 3, we find that most existing works overlook the importance of protecting dataset sizes when sending local updates to the server. However, it is crucial for medical organizations to take notice in guaranteeing the confidentiality, especially when implementing FL techniques on Electronic Health Records (EHR) data [24].

An EHR, usually known as an electronic record of a patient’s chart [25], contains a patient’s medical history, medications, visiting dates, medical test results, biological information, etc. Recently, various ML methods have been presented to practically utilize EHR data in a more privacy-preserving way, FL. For example, Liu et al. [26] proposed a FADL, a federated autonomous deep learning technique to make a prediction of the patients’ mortalities using EHR data while not gathering each hospital’s database. Fioletto et al. [27] presented a data sharing method in FL to predict a diagnosis of a rare disease without sharing the datasets among the institutions. Sheller et al. [28] implemented FL with dispersed data from ten hospitals to distinguish the area of brain cancer in Magnetic Resonance Imaging (MRI) while keeping the patients’ data in the hospital.

Despite the fact that most prior works seek to protect user data privacy by adopting FL techniques, it is found that the size of the training datasets is naturally being sent to the server. In Vaid et al. [29] experiment, we can see that number of patients and their characteristics are transparently revealed to assess if FL outperforms locally trained models in the context of COVID-19. However, this may denote meaningful information when it is leaked to the other parties. For instance, the dataset size can indicate the number of patients in the local hospital, which can be sensitive to be publicized. Moreover, depending on the hospital’s efforts to make it confidential, it is possible to be categorized as a trade secret. It may have more connotations, such as the number of types of patients’ diseases, medications, or surgeries.

Securing each and overall dataset sizes through our protocol also provides security when each size is not uniformly
FIGURE 1. Overview of our protocol.

distributed among the clients. Though Girgis et al. [30] proposed shuffling and subsampling amplifications of data and clients with communication constraints to be secure against the untrusted server, it does not apply to circumstances where the datasets are related to rare diseases, which hardly occur as often as a certain amount of cases among the local hospitals.

Moreover, our proposed system guarantees security of user dataset size when clients dropout in the next iteration. NVIDIA Clara [31], an application framework optimized for healthcare, sets the number of participating clients to be added and removed by the server every iteration. This makes it possible for honest-but-curious adversaries to calculate the dropout client’s dataset size. Such information may indirectly lead to certain attacks such as Inference Attacks [32]. In addition, an attacker given this information is able to estimate which client is more attributing to a global model performance. This offenses availability and degradation of the performance in the process. Our focus in this paper is to prevent leakage of dataset sizes after each client repeatedly iterates training the model using stochastic gradient descent (SGD) method.

V. PRIVACY-PRESERVING FedAvg PROTOCOL

In this section, we construct our solution for Privacy-Preserving Federated Averaging Protocol (PP−FedAvg). We present a possible application in healthcare settings. Then we describe specific steps of our protocol and show that our protocol protects a training dataset size from the central server.

A. SCENARIO

We prescribe a practical scenario of our proposed protocol in the medical field. In general cases, the FedAvg algorithm [1] in Section II-A reveals the size of dataset on each client to the central server so as to update the global model. Each client inevitably sends the subset of the private set of training examples during iterations. Since this could be critical in the medical field where information such as the number of patients in specific diseases is sensitive between the regions, we suggest implementing our protocol in the following scenario:

In the scenario, we explain the participating nodes as categorized below:

- **Key Generation Center (KGC):** A trusted authority that manages the overall system. It issues a HE key pair of the server and a symmetric key for secure communications among the clients, e.g. key agreement protocols.
- **Server:** A central server that aggregates the encrypted dataset sizes as well as local updates sent from the participating hospitals, then forwards the updated global model to each hospital. In the real world, it can be maintained by multiple hospitals or hosted on the cloud.
- **Client:** Hospital institutions that train their local models with Electronic Health Records (EHR) and cooperate to send the final result to the server.

Our proposed environment is one of the cross-silo setting that consists of more reliable clients. Hospitals can outperform other general clients in cross-device settings regarding computation and storage performance. Each hospital surpasses the mobile device in computation power and is less likely to be dropped out due to the network connection. This is quite different from general federated learning settings in that we do not need to concentrate on providing robustness property of the system against dropout situations.

The central server can be managed by the participating clients or by third parties to update the global model by each iteration. This allows hospitals to share the training model in a stable environment without leaking client’s data. Training datasets in healthcare industries are usually clinical records derived from EHRs. They are subject to compliance with related regulations and laws such as Health Insurance Portability and Accountability Act (HIPAA) [33]. The proposed PP−FedAvg protocol enables utilizing sensitive medical data at large-scale while reducing the burden of compliance.

When each hospital finishes training the local model with the local dataset, it initiates our protocol to encrypt its updated local model parameter and the size of the dataset, respectively. After receiving encrypted local update parameters and the datasets of the participating clients, the central server is able to compute the global model parameter while not knowing each and aggregated results.

B. OUR PROTOCOL

We describe our privacy-preserving federated averaging protocol, PP−FedAvg, that protects the size of training datasets of each participating client. Our system, which is depicted...
TABLE 4. Our PP – FedAvg protocol.

| PP – FedAvg Protocol: |
|-----------------------|
| **Setup.** There are a server and k honest-but-curious clients. k is the number of clients, p is the ratio of clients that is chosen for each round, b is the local mini-batch size, t is the number of local epochs, and α is the learning rate. |

1) The server runs HE.Gen(1, λ) to get a public and private key pair (pk, sk). And the server initializes the model parameters \( \omega_1 \).

2) The server and clients run as follows for each round \( j = 1, 2, \ldots \).
   a) The server chooses \( p \) and computes \( m = \max(\text{round}(p \cdot k), 1) \). And the server chooses \( S_j \), which is a random set of \( m \) clients.
   b) The first client in \( S_j \) is the starting client \( i^* \). The starting client \( i^* \) chooses random numbers \( r_1, r_2, r_3 \).
   c) To get \( \omega_{j+1} \), each client \( i \) in \( S_j \) runs as follows:
      i) To get \( D_i \), each client splits \( D_i \) into batches of size \( b \).
      ii) For each local epoch, each client updates \( \omega_j \leftarrow \omega_j - \alpha \nabla \ell(\omega_j; d) \) for batch \( d \in D_i \).
      iii) After updating \( \omega_j \), each client sets \( \omega_{j+1} = \omega_j \).
   d) To get \( \omega_{j+1} \), the starting client runs \( \text{HE.Enc}_{pk}(n_1 \cdot \omega_{j+1} + r_1) \) and \( \text{HE.Enc}_{pk}(n_1 + r_2) \), respectively. And then, the starting client sends \( \omega_{j+1} \) to the next client.
   e) Each client in \( S_j \), except the starting client, receives \( \omega_{j+1} \) from the previous client. Each client in \( S_j \), except the starting client, runs as follows:
      i) \( c_{i+1}^{j+1} = \text{HE.Enc}_{pk}(n_1 \cdot \omega_{j+1}) \).
      ii) \( d_{i+1}^{j+1} = \text{HE.Enc}_{pk}(n_1 + r_2) \).
   f) To get \( \omega_{j+1} \), the starting client runs \( \text{HE.Enc}_{pk}(n_1 \cdot \omega_{j+1} + r_1) \) and \( \text{HE.Enc}_{pk}(n_1 + r_2) \), respectively. And then, the starting client sends \( \omega_{j+1} \) to the server.
   g) The server runs \( \text{HE.Dec}_{pk}(\omega_{j+1}) \) and \( \text{HE.Dec}_{pk}(\omega_{j+1}) \) to get \( \sum_{i=1}^{m_i} n_1 \cdot \omega_{j+1} \cdot r_3 \) and \( \sum_{i=1}^{m_i} n_1 \cdot r_3 \), respectively. And then, the server updates \( \omega_{j+1} = (\sum_{i=1}^{m_i} n_1 \cdot \omega_{j+1} \cdot r_3) / (\sum_{i=1}^{m_i} n_1 \cdot r_3) \).

in Table 4, mainly consists of a KGC, central server and \( m \) participating clients. First, the starting client encrypts \( n_1 \cdot \omega^1 + r_1 \) and \( n_1 + r_2 \) using the server’s public key of the homomorphic encryption in Section II, where \( D_i \) is the dataset on the client \( i \), \( n_i \) is the number of the elements in \( D_i \), \( \omega^i \) is the local model parameters of the client \( i \), and \( r_j \) is a random number selected by the starting client. Then, each client encrypts \( n_1 \cdot \omega^1 + n_2 \cdot \omega^2 + \cdots + n_{m_i} \cdot \omega^m + r_1 \) and \( n_1 + n_2 + \cdots + n_{m_i} + r_2 \) by homomorphic properties and sends them to the next client, where we assume that each user communicates over a secure channel to prevent eavesdropping attacks from the server, such as key agreement protocols. Although it seems to require additional computation overhead of generating the symmetric key as well as extra modular multiplications and exponentiation for HE, the cross-silo setting in the scenario enables sufficient resources to operate much secure FL. After the communications, the ending client sends the final ciphertexts of \( n_1 \cdot \omega^1 + n_2 \cdot \omega^2 + \cdots + n_{m_i} \cdot \omega^m + r_1 \) and \( n_1 + n_2 + \cdots + n_{m_i} + r_2 \) to the starting client. The starting client removes the random numbers, \( r_1 \) and \( r_2 \), and then multiplies by the random number \( r_3 \) to the ciphertexts using the homomorphic properties. The starting client sends them to the central server. At last, the central server can get the model parameters \( \omega_j \) by dividing the decrypted values. As shown in Table 4, we propose our PP – FedAvg protocol.

VI. ANALYSIS

A. SECURITY ANALYSIS

In the FedAvg algorithm [1], the server needs to compute \( \omega_{j+1} = \sum_{i=1}^{n} n_i \cdot \omega_i \) to get the updated global model parameters. Therefore, each client should send \( n_i \) and \( \omega_i \) to the server. In our protocol, the server is defined as honest-but-curious, meaning that it honestly follows the overall protocol, but also tries to acquire sensitive information during the process. To protect a training dataset size on each client, clients aggregate ciphertexts securely to get the sum of dataset sizes. Clients cooperatively compute the encrypted ciphertexts, \( \text{HE.Enc}_{pk}(\sum_{i=1}^{m_i} n_i \cdot \omega_i \cdot r_3) \) and \( \text{HE.Enc}_{pk}(\sum_{i=1}^{m_i} n_i \cdot r_3) \) and send these ciphertexts to the central server. After decrypting the ciphertexts to get \( \sum_{i=1}^{m_i} n_i \cdot \omega_i \cdot r_3 \) and \( \sum_{i=1}^{m_i} n_i \cdot r_3 \), the server computes \( \omega_{j+1} = \sum_{i=1}^{m_i} n_i \cdot \omega_i \cdot r_3 = \sum_{i=1}^{m_i} n_i \cdot \omega_i \cdot r_3 / (\sum_{i=1}^{m_i} n_i \cdot r_3) \). Finally, the server can get \( \omega_{j+1} \) without
knowing the size of dataset on each client and even the sum of the dataset sizes of clients. We note that if the central server can get \( n = \sum_{i=1}^{n} n_i \) then the server knows the dataset size of a specific client. For example, if the server can get \( n = \sum_{i=1}^{m} n_i \) and \( n' = \sum_{i=2}^{m} n_i \), then the server knows the size of dataset on the first client by computing \( n_1 = n - n' \).

To conceal the dataset size on each client from the server, our protocol conceals the sum of the dataset sizes of clients from the central server.

### B. EFFICIENCY ANALYSIS

In this section, we compare our **PP – FedAvg** with the previous works to analyze the computation and communication costs required during the execution of each protocol. Table 5 shows the timing costs of homomorphic encryption (HE) operations, communications, and the number of HE key pairs required in each scheme. We calculate the complexities with Big O notations to weigh up our scheme’s benefits and trade-offs. For communications, we measured for values that need to be transmitted in privacy-preserving manner. Due to the fact that the roles of the starting client or the leader are different from the other clients, we separate the category of clients into the starting client(1st) and others. We mainly compare with PEFL [13] and BatchCrypt [14] on account of the fact that they both utilized the same additively homomorphic encryption (AHE) in the horizontal framework, as represented in Table 3.

In PEFL [13], when each local client finishes registration and authentication through the server, the client computes the local model update, adds the random value, and encrypts it with the public key of the server. Since the server does the aggregation on the encrypted results when received from the clients, it needs \( O(mk) \) operations of addition during the execution. Then, the server does the decryption on the aggregated result with its private key. For communications, while each client sends the updated result, the server sends the initial and updated global model at the beginning and end of the phase, respectively.

In BatchCrypt [14], it adopts the two types of clients at the beginning of the scheme. In the initializing phase, the server randomly selects the leader to generate and synchronize the clients’ HE key pairs, which requires extra communication costs \( O(k) \) for the leader. After receiving the key pair, each client then trains and sends the gradient range of each layer along with the size to the server before sending the encrypted result. This additional phase is required to realize quantization and demands more communication for the entities. When the server computes and returns the layer-wise clipping values, the clients encrypt the quantized weights with its public key and decrypt the update result when received the global update from the server.

As we aim to enhance the properties of privacy and security, protecting each and overall dataset size along with the local model update, there naturally exists a trade-off between security and efficiency. However, the result shows that the trade-off is realizable enough to be implemented in real-world settings. Compared to other schemes, our scheme shows a smaller number of operations from the server side. This is due to the semi-aggregation by the client side, which is part of the server’s role in the general schemes. The starting client masks its updated result as well as the dataset size, then sends the results to other clients. After receiving from the last client, it computes addition and lastly masks the results with the random value.

Furthermore, considering that the server is a semi-trusted entity, it is significant to diminish the role of the server in order to be secure under the FL framework. In our scheme, although the server can decrypt the encrypted local updates and the dataset sizes, it can only figure out the aggregated result because the random value \( r_3 \) can only be offset by the computation.

### VII. DISCUSSION

In this section, we provide further discussion on our protocol, **PP – FedAvg**. Currently, various works on FL have been proposed based on homomorphic cryptosystems to enhance user data privacy. However, realization is still confronting the practical issue due to the inefficiency problem in homomorphic encryption, especially when utilizing Pailler’s cryptosystem. Considering that encrypting the gradients and dataset sizes inevitably consumes communication and computational overheads, we highly recommend that our protocol is suitable for settings upon which security concerns more rely, e.g. medical environment.

---

**TABLE 5.** Security trade-off analysis of our PP-FedAvg compared to the related works.

| Schemes          | Encryption | Addition | Decryption | Communication |
|------------------|------------|----------|------------|--------------|
|                  | Client     | Server   | Client     | Server       | Client       | Others       | Server       |
| PEFL [13]        | \( O(m) \) | 0        | \( O(mk) \) | 0            | \( O(m) \)   | N/A          | \( O(m) \)   |
| BatchCrypt [14]  | \( O(m) \) | 0        | \( O(mk) \) | 0            | \( O(m) \)   | \( O(m + k) \)| \( O(mk) \)  |
| PP – FedAvg (Ours)| \( O(m + n) \)| 0     | \( O(m + n) \)| 0            | \( O(m+n) \) | \( O(m+n) \) | \( O(m+k) \) |

1st client denotes the leader or the starting client of which the role is different from the others. \( m \) denotes the operation’s input size of model updates. \( k \) denotes the number of clients. N/A denotes Not Applicable. \( n \) denotes the operation’s input size of datasets. Please note that it is natural to have fewer benefits in efficiency because of the extra operations needed to protect the dataset sizes. However, our protocol is applicable in that we reduced the total amount of operation on the server side. Also, when not considering the operations on the datasets, our scheme still has competitive results compared to the previous works.
To provide a better guideline for implementation, we construct the most adequate scenario for our PP – FedAvg where each client is comprised of medical institutions, commonly known as cross-silo FL. The participating entity has sufficient computing resources enough to tolerate the costs. This kind of environment has less burden on establishing a Trusted Authority that manages the overall system. The requirement of data privacy is highly strict because it must comply with related laws. We state that our protocol facilitates medical federated learning in such conditions.

VIII. CONCLUSION

In this paper, we present a privacy-preserving federated averaging protocol that protects local training dataset sizes as well as the aggregated local parameters when sending them to the central server. Our protocol is suitable for cross-silo FL settings, more specifically in healthcare applications where such information can be essentially privacy-sensitive. In these circumstances, the information can denote the number or characteristics of patients in the local hospital. Moreover, our protocol is suitable when the size is not uniformly distributed among the hospitals and the clients’ dropout.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, “Communication-efficient learning of deep networks from decentralized data,” 2016, arXiv:1602.05629.

[2] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, “Federated learning for mobile keyboard prediction,” 2018, arXiv:1811.03604.

[3] Y. Zhao and H. Sun, “Information theoretic secure aggregation with user dropouts,” 2021, arXiv:2101.07750.

[4] P. Kairouz et al., “Advances and open problems in federated learning,” 2019, arXiv:1912.04977.

[5] N. Rieke, J. Hancos, W. Li, F. Miletarì, H. R. Roth, S. Albarqouni, B. M. Galtier, B. A. Landman, K. Maier-Hein, S. Ourselin, S. Patel, D. Ramage, A. Segal, and K. Seth, “Federated federated learning for mobile keyboard prediction,” 2018, arXiv:1811.03604.

[6] X. Guo, Z. Liu, J. Li, J. Gao, B. Hou, C. Dong, and T. Baker, “VeriFL: Communication-efficient and fast verifiable aggregation for federated learning,” IEEE Trans. Inf. Forensics Security, vol. 16, pp. 1736–1751, 2021, doi: 10.1109/TIFS.2020.3043139.

[7] F. Fioretto and P. V. Hentenryck, “Privacy-preserving federated learning via additively homomorphic encryption,” 2017, arXiv:1711.06707.

[8] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, “A survey on homomorphic encryption schemes: Theory and implementation,” Proc. 12th ACM Workshop Artif. Intell. Secur. (AISeC), 2019, pp. 3–11.

[9] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, “HybridAlpha: An efficient approach for privacy-preserving federated learning,” in Proc. 12th ACM Workshop Artif. Intell. Secur. (AISeC), 2019, pp. 13–23.

[10] S. Keyhani, P. L. Hebert, J. S. Ross, A. Federman, C. W. Zhu, and A. L. Siu, “Electronic health record components and the quality of care,” Med. Care, vol. 46, no. 12, pp. 1267–1272, Dec. 2008.

[11] Y. Zhou, “A hybrid approach to privacy-preserving federated learning,” in Proc. 12th ACM Workshop Artif. Intell. Secur. (AISeC), 2019, pp. 1–11.
YOUNG AH SHIN received the B.S. degree in convergence security from Sungshin University, Seoul, South Korea, in 2020. She is currently pursuing the Ph.D. degree in information security with Korea University, Seoul. Her main research interests include cryptographic protocols and privacy-preserving technologies, such as federated learning and blockchain.

GEONTAE NOH received the B.S. degree in industrial systems and information engineering, the M.S. degree in information management and security, and the Ph.D. degree in information security from Korea University, Seoul, South Korea, in 2008, 2010, and 2014, respectively. Currently, he is a member of the Faculty Member with the Department of Big Data and Information Security, Seoul Cyber University, Seoul. His research interests include cryptographic protocols, lattice-based cryptosystems, blockchain, and privacy-preserving technologies.

IK RAE JEONG received the B.S. and M.S. degrees in computer science and the Ph.D. degree in information security from Korea University, South Korea, in 1998, 2000, and 2004, respectively. From June 2006 to February 2008, he was a Senior Engineer at the Electronics and Telecommunications Research Institute (ETRI), South Korea. He is currently a Faculty Member with the Graduate School of Information Security, Korea University. His current research interests include cryptography, theoretical computer science, blockchain, and biometrics.

JI YOUNG CHUN received the B.S. degree in mathematics from Ewha Womans University, Seoul, South Korea, in 1997, and the M.S. and Ph.D. degrees in information security from Korea University, Seoul, in 2006 and 2011, respectively. Currently, she is a Faculty Member with the Department of Big Data and Information Security, Seoul Cyber University, Seoul. Her research interests include federated learning, big data security, cryptographic protocols, and privacy-enhancing technologies.