ID-Based Multireceiver Homomorphic Proxy Re-Encryption in Federated Learning

CHUN-I FAN, YA-WEN HSU, and CHENG-HAN SHIE, National Sun Yat-sen University, Taiwan
YI-FAN TSENG, National Chengchi University, Taiwan

Data privacy has become a growing concern with advances in machine learning. Federated learning (FL) is a type of machine learning invented by Google in 2016. In FL, the main aim is to train a high-accuracy global model by aggregating the local models uploaded by participants, and all data in the process are kept locally. However, compromises to security in the cloud server or among participants render this process insufficiently secure. To solve the problem, this article presents an identity-based multireceiver homomorphic proxy re-encryption (IMHPRE) scheme that utilizes homomorphism operations and re-encryption to provide improved encrypted-data processing and access control. When this scheme is employed, participants can directly use public identities for encryption. The IMHPRE scheme is also secure against the chosen-plaintext attacks. Comparison results indicated that the IMHPRE outperforms its counterparts because it allows a cloud server to perform model aggregation on re-encrypted models for multiple receivers.

CCS Concepts: • **Security and privacy** → **Public key encryption**; • **Computing methodologies** → *Neural networks;*

Additional Key Words and Phrases: Federated learning, homomorphic re-encryption, multireceiver encryption, secure computing for artificial intelligence

ACM Reference format:
Chun-I Fan, Ya-Wen Hsu, Cheng-Han Shie, and Yi-Fan Tseng. 2022. ID-Based Multireceiver Homomorphic Proxy Re-Encryption in Federated Learning. *ACM Trans. Sensor Netw.* 18, 4, Article 55 (November 2022), 25 pages.
https://doi.org/10.1145/3540199

1 INTRODUCTION

Privacy in the cloud is of great scholarly and practical interest [36]. In recent years, computing processes such as deep learning and data mining have been offloaded to the cloud because they are becoming too onerous for local hardware to handle. However, privacy may be compromised in the cloud because cloud environments are usually beyond the control of users. To protect privacy, users

This work was partially supported by the Ministry of Science and Technology (MOST) of Taiwan under Grants no. 111-2218-E-110-001-MBK, no. 110-2923-E-110-001-MY3, and no. 110-2221-E-004-003. It also was financially supported by the Information Security Research Center at National Sun Yat-sen University in Taiwan and the Intelligent Electronic Commerce Research Center from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.
Authors' addresses: C.-I. Fan, Y.-W. Hsu, and C.-H. Shie (corresponding author), National Sun Yat-sen University, No. 70, Lienhai Rd., Gushan Dist., Kaohsiung 80424, Taiwan; emails: cifan@mail.cse.nsysu.edu.tw; she5976@gmail.com, hanhan3927@g-mail.nsysu.edu.tw; Y.-F. Tseng, National Chengchi University, No. 64, Sec. 2, Zhinan Rd., Wenshan Dist., Taipei 11605, Taiwan; email: yftseng@cs.nccu.edu.tw.

1550-4859/2022/11-ART55 $15.00
https://doi.org/10.1145/3540199

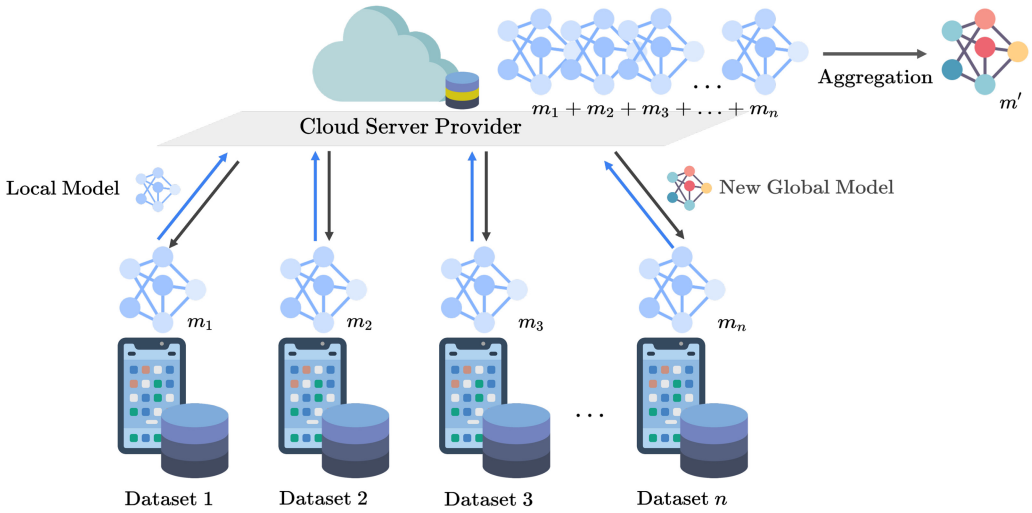


Fig. 1. Federated learning.

tend to upload ciphertext rather than plaintext or may even be unwilling to upload data; however, this impedes industrial operations or scholarly research (e.g., in clinical research involving large datasets).

Federated learning (FL) [23] is a novel machine learning architecture that relies on cloud servers to aggregate information from each entity to produce a highly accurate model. As shown in Figure 1, consider n participants in FL whose sensitive data are in a local dataset (Figure 1). Each participant trains local models $m_1, m_2, \ldots, m_n$ based on the local datasets and uploads these models to the cloud server for aggregation. By summing the gradients of the local model from each participant, the cloud server produces and returns a more accurate global model. Finally, participants update the local model with the received global model. This improved local model is then uploaded to the cloud for aggregation, and this cycle iterates until model convergence. Two privacy concerns in FL should be addressed. First, although FL ensures the local storage of data, participant privacy may be compromised when participants upload local models to the untrusted cloud server in plaintext form. Second, local models may leak information to each other. Correspondingly, FL must achieve the following to ensure privacy:

- Except for the data providers, no one should know any information about the uploaded model without permission, even information regarding the cloud server or other participants.
- The cloud server without the corresponding private key can only be allowed to aggregate the uploaded local model in ciphertext form.
- Participants can decrypt the returned global model, but no participant should be able to decrypt the local model of anyone else.

1.1 Methods for Attack and Defense in Federated Learning

Although client privacy is preserved in FL because the local dataset is not uploaded, many studies have indicated that classical FL has several weaknesses with regard to privacy. Specifically, members in an FL system typically use authentication and secure channels to ensure user secrecy. However, in FL training, the participants and aggregators in FL may become sources of data leakage. The following sections discuss the two primary attacks based on FL and the corresponding defense skills.

1.1.1 Poisoning/Backdoor Attack. The FL model relies on the gradient parameter generated from training on the participants' data to update the global model. If the uploaded gradient from the participant is not verified and aggregated, the accuracy of the public model may be compromised or a backdoor may be embedded. Cao et al. [8] presented a poisoning attack method for FL systems and demonstrated that the number of poisoned data points and the number of attackers affect the performance of the poisoning attack. Under the poisoning attacks, since the model loss does not decrease and the model cannot converge, the accuracy of the global model cannot be improved. The poisoning attack of Cao et al. [8] greatly affects training in FL systems. In addition, Chen et al. [10] presented a backdoor attack method for FL systems. Unlike the method of Cao et al. [8], when a specific input is encountered, the FL model that is embedding the backdoor will output an incorrect answer.

1.1.2 Privacy Attack. A privacy attack occurs when information about the participants is disclosed. Aggregators are often assumed to be trustworthy, but this strong assumption often fails to hold in practice. Therefore, the next most realistic assumption that one can make is of a curious but honest aggregator. Under this assumption, the aggregator honestly aggregates the gradient parameters from the participants into the public model. However, as noted by many studies, participants may pass information on model gradient parameters instead of local data. For example, Zhu et al. [44] formulated their **deep leakage from gradient** (**DLG**) method that can obtain the local training data from public shared gradients without relying on extra information. Studies, such as that of Zhu et al. [44], have demonstrated that the attacker can still obtain some information through model gradient parameters, which leads to privacy leakage problems. Privacy attacks take one of three forms depending on the attacker's objectives: feature inference attacks, property inference attacks, and membership inference attacks. DLG [44] is a type of feature inference attack where the aim is to recover the dataset of the participants. By contrast, in property inference attacks, the aim is to extract the attributes. Wang et al. [39] formulated a property inference attack model that can infer the existence of an objective attribute, such as brown eyes in a face recognition dataset, that corresponds to some property or population of participants. Truex et al. [37] also presented a black-box membership inference attack method to infer the membership of an individual model by an **application programming interface** (**API**; such as Google Prediction API or Amazon ML) without any model parameter.

1.1.3 Methods for Defending Against Poisoning or Backdoor Attacks. Poisoning and backdoor attacks work primarily because malicious participants upload the modified gradients for aggregation; this leads to an updated global model that is inaccurate or that deliberately outputs incorrect results. Methods for defense thus focus on the gradient. For example, Yin et al. [43] proposed two robust distributed gradient descent algorithms based on median and trimmed mean operations. Naseri et al. [30] demonstrated that local differential privacy and central differential privacy methods can effectively defend against backdoor attacks in FL. The local differential privacy is usually processed to protect the local gradient at the client side. The central differential privacy is processed after aggregation at the server side, which prevents inference attacks from global model attacks. Another approach to defense is based on the fact that only a small proportion of the model's weight is affected by the poisoning attack. Inspired by the aforementioned ideas, Wu et al. [40] formulated a federated pruning method in which a portion of neurons of the global model are removed, and their experimental results demonstrated that the method led to a low loss of accuracy and backdoor attack rate.

1.1.4 Methods for Defending Against Privacy Attacks. Another type of defense guards against privacy attacks, leakages with regard to gradient information, or black-box inference attacks

against the global model. Therefore, defense methods are usually designed to work on the local dataset or both the gradient and global model. Many studies have formulated differential privacy-based defenses for preventing poisoning attacks and backdoor attacks. Abadi et al. [1] presented local differential privacy defense methods that are based on a differentially private stochastic-gradient-descent algorithm, by adding noise to the gradient to prevent leakage of gradient information. Other defense methods in the literature are not based on differential privacy. For example, Zhu et al. [44] proposed a defense method based on gradient compression that prunes the gradient to be below a threshold magnitude to prevent feature inference attacks. Defense methods with pruning and differential privacy can resist most poison or backdoor attacks and privacy attacks. However, these two defense methods require the removal of a few neurons or the injection of noise, which decrease the accuracy.

Note that FL is a form of **multiparty computation** (**MPC**). Therefore, privacy can be preserved in FL through a redefinition of its protocol or components with **secure MPC** (**SMPC**) methods. FL has been applied in different scenarios such as pneumonia detection [2] and Android malware detection [18] to perform machine learning with privacy protection. Primarily, SMPC relies on **homomorphic encryption** (**HE**) [17] as an essential component to ensure that computation is secure in the absence of decryption. The secure computations in HE can cover the operator used in FL aggregation, but SMPC becomes more complex with more participants in the computation. However, HE provides FL with the most secure solution that FL then takes as the basis for computation on the encrypted data; this happens without any risk of knowledge leakage or loss of accuracy in the FL model.

However, HE in FL requires sharing a key for different local sides to perform computations on its encrypted data. A mere usage of HE may not be enough to deal with the privacy issues in FL. Therefore, in our work, we combine HE with proxy re-encryption skill, which can reduce the communication cost of keys agreement. Furthermore, although HE cannot solve poisoning or backdoor attacks, the malicious model's provider can be tracked from the certificate or certificateless public key and be disqualified from participating. Section 1.2 will show the discussions of different mechanisms with homomorphism appropriate to FL.

1.2 Homomorphic Encryption in FL

HE can be considered the ideal solution for aggregating local models in ciphertext form. The processing of encrypted data is called homomorphic evaluation. Because the value of a decrypted result is the same as that calculated directly from the plaintext, some studies have used HE to develop secure applications. For example, in 2017, Aono et al. [3] proposed a secure deep learning scheme with HE. However, in this scheme, homomorphic evaluations can only be performed on the ciphertext encrypted under the same key. Participants must establish a common HE key and use it for encryption. In such a situation, privacy can be compromised at the input because each participant can also use the same key to reveal the local model of the other party. Therefore, HE maintains input data confidentiality only if the additional secure channel assumption is satisfied, which makes HE impracticable for FL systems.

Different from HE, **multi-key fully homomorphic encryption** (**MKFHE**) [24] supports homomorphic evaluation performed on the ciphertext encrypted under different keys. However, MKFHE requires all secret keys involved in the input ciphertext of homomorphic evaluation to decrypt the final result. In other words, the data provider who has the corresponding private key should either share its key with others or interactively decrypt the evaluated ciphertext using SMPC. However, this results in considerable communication overhead among users. Furthermore, the sharing of secret keys with other receivers is undesirable because it enables receivers to

decrypt the input ciphertexts before homomorphic evaluation. Accordingly, MKFHE methods with these properties are also unsuited to being directly applied to FL systems.

Homomorphic proxy re-encryption (**HPRE**) methods that simultaneously support the processing of encrypted data and provide access control are another option for FL systems. Specifically, **proxy re-encryption** (**PRE**) [4] enables a third party to re-encrypt the ciphertext of one key to the ciphertext of another key without decrypting the ciphertext. Some studies have leveraged HPRE to improve security in cloud applications. For example, in 2017, Shafagh et al. [35] proposed an HPRE scheme for Internet of Things applications that allows for query processing over encrypted data. In 2019, Kawai et al. [21] enhanced the security level of the HPRE scheme to protect the cloud data processes. And in 2020, Ma et al. [26] used HPRE and aggregate signature techniques to secure the multiparty learning. In HPRE, with the re-encryption process, no user, including the receivers who can decrypt the result with their private keys, can obtain any information about the original data. Furthermore, participants are not required to establish a common key or communicate with each other for decryption. Therefore, compared with HE and MKFHE, HPRE is a more suitable cryptosystem for privacy protection in FL.

HPRE can be implemented through various approaches. In 1998, Blaze et al. [4] proposed the first PRE scheme based on the ElGamal cryptosystem, but the scheme's inherently bidirectional property makes it unsuitable for many applications. In 2015, Samanthula et al. [34] proposed a privacy-preserving data-sharing framework. In 2019, Gao et al. [16] proposed a secure profile-matching scheme under multiple keys. They utilized an ElGamal-like PRE algorithm to simultaneously provide additive homomorphism and re-encryption. In 2017, Ding et al. [13] proposed an HPRE scheme for secure data processing based on the BCP cryptosystem [7]. However, the proposed scheme allows two noncolluding servers to decrypt or re-encrypt the uploaded data without the permission of the data providers. Therefore, in 2019, Nateghizad et al. [31] proposed a **Homomorphic One-Direction Proxy Re-Encryption** (**HOPE**) scheme to solve this problem. In 2020, Luo et al. [25] proposed an identity-based HPRE scheme; this scheme differs from the aforementioned methods in that it is more flexible and does away with the need for certificate verification in the Public-Key Encryption system. On the other hand, an FL system needs to submit a new global model to multiple receivers after updating the gradients. Hence, a multireceiver setting, such as that in [14, 15, 38], should be also considered in HPRE.

1.3 Contributions

Although existing HPRE systems have become more practical, they can only re-encrypt ciphertext for one receiver at a time, making them inflexible in FL applications. For example, the cloud server must re-encrypt the uploaded models of all participants and perform homomorphic evaluations on the re-encrypted models for the same receiver separately during model aggregation. The process incurs a large computational cost and communication overhead. To address these limitations, we formulated an **identity-based multireceiver HPRE** (**IMHPRE**) scheme. In the proposed IMHPRE scheme, re-encryption and homomorphic evaluations are only performed once for multiple participants. Furthermore, different from traditional HPRE methods, participants in the IMHPRE scheme can use their private keys and other users' public identities directly to construct re-encryption keys, which affords greater convenience.

The proposed IMHPRE scheme is also more practicable than other methods for FL. Note that in the IMHPRE scheme, participants can also directly encrypt data for others. However, this results in large local overheads because encryption for multiple receivers is more computationally intensive than encryption for one receiver. Crucially, FL requires participants to repeatedly upload their local model for training aggregation, which increases the computational cost. By contrast, in IMHPRE with a re-encryption key, the overhead from encrypting data for others can be offloaded to a cloud

server. Similar to a data backup to the cloud, users can upload data encrypted only with its identity. Furthermore, the cloud server without the re-encryption key can neither re-encrypt the ciphertext nor perform homomorphic evaluations on them. Consequently, with these properties, IMHPRE can provide a more flexible and privacy preserving for FL systems.

To meet the requirements in privacy-preserving FL, the proposed IMHPRE scheme provides the following features.

(1) The proposed scheme avoids the need for certificate verification in the PKE system.
(2) An individual cannot obtain any information about an uploaded model without the corresponding private key.
(3) The proposed scheme supports homomorphic addition performed on re-encrypted local models for multiple receivers.
(4) Participants can decrypt the aggregated model using their private keys, but no participant can decrypt the local model of anyone else.
(5) The proposed scheme outperforms its counterparts in terms of ciphertext length and computation cost.

2 PRELIMINARIES

This section introduces the technological and cryptographic preliminaries underlying the proposed IMHPRE scheme. The mathematical assumptions used in the security proofs and the definition of IMHPRE are also presented in this section.

2.1 FL

In FL, a global model is trained on the basis of all the data from different entities, enabling each entity to maintain its data securely. The Federated Averaging Algorithm [28] is one of many algorithms for FL. Let S denote a fixed set of participants. Each participant k holds n_k data points, and n_σ denotes the sum of the data points from S. Moreover, the index t represents the t-th communication round of the cloud server, which aggregates the local models from the set of participants. In general, and as illustrated in Figure 2, FL proceeds per the following steps.

(1) Each participant downloads the current global model w_t from the cloud server.
(2) Each participant updates its local model $w_t^{(k)}$ with w_t.
(3) Each participant uses its local data to train its local model based on the stochastic gradient descent algorithm and generates the new local model $w_{t+1}^{(k)}$.
(4) Each participant uploads $w_{t+1}^{(k)}$ to the cloud server.
(5) The cloud server aggregates the received local models and updates the global model as

$$w_{t+1} = \sum_{k \in S} \frac{n_k}{n_\sigma} w_{t+1}^{(k)} \ , \ n_\sigma = \sum_{k \in S} n_k. \tag{1}$$

The aforementioned steps are iterated until the global model converges. Note that each local model comprises a set of weights, and the cloud server aggregates all received local models by calculating the weighted sum. Thus, FL requires the ability to execute addition operations on the server side.

2.2 Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be source multiplicative and target multiplicative cyclic groups, respectively, and let p be their prime order. A function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfying the following properties is called a bilinear map where g_1, g_2 are two generators of $\mathbb{G}$, and α, $\nu \in \mathbb{Z}_p$.

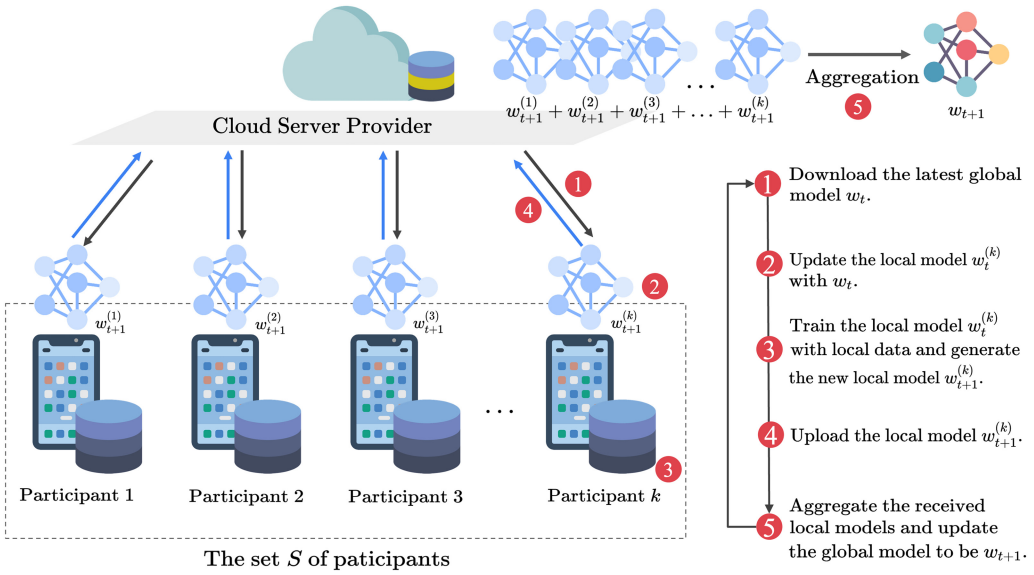


Fig. 2. Federated learning.

— **Bilinearity:** $e(g_1{}^{\alpha}, g_2{}^{\nu}) = e(g_1, g_2)^{\alpha\nu}$.
— **Non-Degeneracy:** Since $\mathbb{G}$ and $\mathbb{G}_T$ are two groups of the same order, $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.
— **Computability:** There exists an efficient algorithm to compute $e(g_1, g_2)$.

2.3 Computing Discrete Logarithm

The proposed scheme encrypts the plaintext m in the form of z^m to provide additive homomorphism. Accordingly, users must solve the discrete logarithm problem to reveal the plaintext m during decryption. Gao et al. [16] indicated that Pollard's kangaroo method [33] can compute a discrete logarithm efficiently when the plaintext is less than 40 bits. And its time complexity is $O(\sqrt{\mathcal{M}})$, where $\mathcal{M}$ is the message space of m. In Section 4, the process for solving the problem is denoted as $DLP()$.

2.4 The *q-SP-DBDH*$_{\mathbb{G},e}$ Assumption

The *q-SP-DBDH*$_{\mathbb{G},e}$ problem provides a stronger variant of the **Successive-Power** (**SP**) version **Decision Bilinear Diffie-Hellman** (**DBDH**) problem [5] for giving access to a sequence of powers. First, let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map with two multiplicative groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p. The definitions are as follows.

Definition 2.1. The *q-SP-DBDH*$_{\mathbb{G},e}$ problem [19] is that given $\{\mathbb{G}, \mathbb{G}_T, e, \mu, \mu^a, \mu^b, \mu^c, \mu^{c/a}, \mu^{c/a^2}, \ldots, \mu^{c/a^q}, \mathcal{V}\}$ where generators $\mu, \mu^a, \mu^b, \mu^c, \mu^{c/a}, \mu^{c/a^2}, \ldots, \mu^{c/a^q} \in \mathbb{G}$, q is an integer, and $\mathcal{V} \in \mathbb{G}_T$, decide whether $\mathcal{V} = e(\mu, \mu)^{abc}$ for $a, b, c \in \mathbb{Z}_p^*$.

Definition 2.2. Define that an algorithm $\mathcal{A}$ with an output $\beta \in \{0, 1\}$ has advantage ϵ in solving the *q-SP-DBDH*$_{\mathbb{G},e}$ problem if

$$\left| Pr[\mathcal{A}(\mu, \mu^a, \mu^b, \mu^c, \mu^{c/a}, \ldots, \mu^{c/a^q}, e(\mu, \mu)^{abc}) = 1] - Pr[\mathcal{A}(\mu, \mu^a, \mu^b, \mu^c, \mu^{c/a}, \ldots, \mu^{c/a^q}, \mathcal{V}) = 1] \right| \geq \epsilon, \tag{2}$$

where $a, b, c \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and $\mathcal{V} \overset{R}{\leftarrow} \mathbb{G}_T$. The *q-SP-DBDH*$_{\mathbb{G},e}$ assumption holds if no polynomial-time algorithm has non-negligible advantage in solving the *q-SP-DBDH*$_{\mathbb{G},e}$ problem.

2.5 IMHPRE

Some definitions for the security model of the proposed IMHPRE scheme are presented in this subsection. Notably, the scheme contains eight algorithms that are suitable for FL applications.

2.5.1 Algorithms. The proposed IMHPRE scheme has eight algorithms—**Setup**, **KeyGen**, **Enc**, **DecO**, ***ReKeyGen***, **ReEnc**, **DecR**, and **Add** as follows.

— **Setup**$(\varphi, \mathcal{M})$:
The algorithm takes the security parameter φ and the maximum number $\mathcal{M}$ of receivers as inputs. Then it outputs the master public key $\mathcal{MPK}$ and the master secret key $\mathcal{MSK}$.
— **KeyGen**$(\mathcal{MSK}, ID_i)$:
The algorithm takes $\mathcal{MSK}$ and the i-th user's identity ID_i as inputs. Then it outputs the user's private key sk_{ID_i}.
— **Enc**$(\mathcal{MPK}, ID_i, s_i, m)$:
The algorithm takes $\mathcal{MPK}$, ID_i, the secret value used for re-encryption s_i, and a plaintext m as inputs. It then outputs the original ciphertext CT.
— **DecO**$(\mathcal{MPK}, sk_{ID_i}, CT)$:
The algorithm takes $\mathcal{MPK}$, sk_{ID_i}, and CT as inputs, and then outputs the plaintext m.
— **ReKeyGen**$(\mathcal{MPK}, ID_i, sk_{ID_i}, s_i, S)$:
The algorithm takes $\mathcal{MPK}$, ID_i, sk_{ID_i}, s_i, and a receiver set S as inputs. Then it outputs a re-encryption key $RK_{ID_i \to S}$.
— **ReEnc**$(\mathcal{MPK}, RK_{ID_i \to S}, CT)$:
The algorithm takes $\mathcal{MPK}$, $RK_{ID_i \to S}$, and CT as inputs. Then it outputs the re-encrypted ciphertext CT_r.
— **DecR**$(\mathcal{MPK}, sk_{ID_j}, CT_r)$:
The algorithm takes $\mathcal{MPK}$, the private key sk_{ID_j} of receiver $ID_j \in S$, and CT_r as inputs. It then outputs the plaintext m.
— **Add**$(CT_{r_1}, CT_{r_2}, \ldots, CT_{r_t})$:
The algorithm takes re-encrypted ciphertexts CT_{r_1}, $CT_{r_2}, \ldots$, and CT_{r_t}, respectively, from users 1, 2, ..., and r, respectively, as inputs, and then outputs the aggregated ciphertext CT_{Add}.

2.5.2 Algorithms Applied in FL. As illustrated in Figure 3, the aforementioned algorithms can be applied to an FL system. In practice, a trusted entity, **key generation center** (**KGC**), first executes **Setup** to initialize the system and establish the private key via **KeyGen** for each participant. Subsequently, participants can use **Enc** to protect the uploaded local models. Furthermore, each participant sends the re-encryption key generated by **ReKeyGen** to the **cloud service provider** (**CSP**), which allows the CSP to re-encrypt their local models for all participants and perform model aggregation by executing **ReEnc** and **Add**, respectively. Finally, each participant can execute **DecR** to directly decrypt the aggregated global model using its private key. Through these algorithms, IMHPRE can provide the correct aggregated model to the participants while protecting their privacy at the local models.

2.5.3 Security Model. With the following **INDistinguishability against selective IDentity under Chosen-Plaintext Attack** (**IND-sID-CPA**) game, the security model of the proposed IMHPRE scheme is defined in *Definition* 2.3 based on the IND-sID-CPA security defined by Boneh

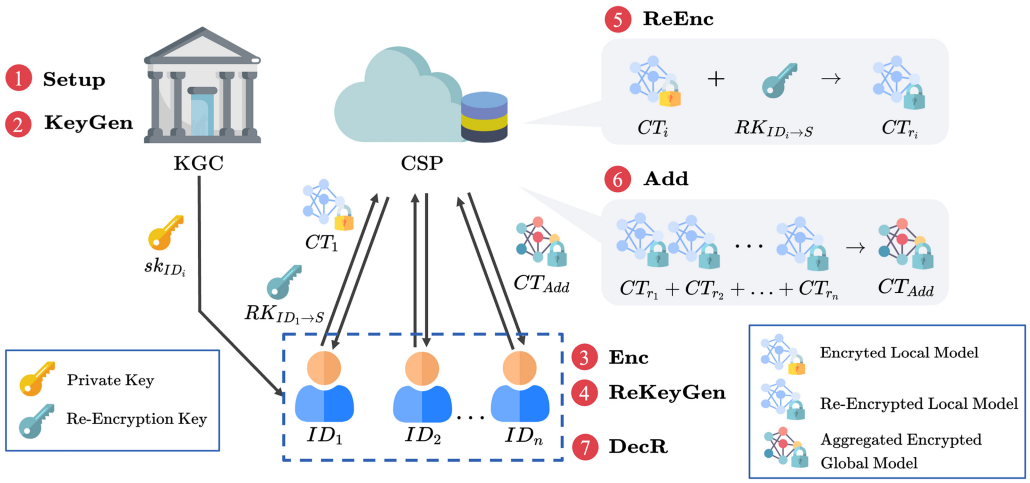


Fig. 3. Algorithms applied in federated learning.

and Boyen [6] where no adversary can know any information about a plaintext from its ciphertext without the corresponding private key.

Definition 2.3 (The IND-sID-CPA Game). Let $\mathcal{A}$ be a **probabilistic polynomial time** (**PPT**) adversary and $\mathcal{S}$ be a simulator that simulates the following game.

- **Initialization:** Adversary $\mathcal{A}$ chooses a challenge identity ID^* and sends it to simulator $\mathcal{S}$.
- **Setup:** $\mathcal{S}$ sets up the system, generates $\{\mathcal{MPK}, \mathcal{MSK}\}$ key pair, and sends $\mathcal{MPK}$ to $\mathcal{A}$.
- **Phase 1:** $\mathcal{A}$ makes the following queries.
  - Private key query $\mathbb{Q}_{sk}(ID_A)$: $\mathcal{A}$ sends an identity ID_A to $\mathcal{S}$. If $ID_A \neq ID^*$, $\mathcal{S}$ returns the private key of ID_A to $\mathcal{A}$.
  - Re-encryption key query $\mathbb{Q}_{rk}(ID_A, R)$: $\mathcal{A}$ sends an identity ID_A and a receiver set R to $\mathcal{S}$. If $ID_A \neq ID^*$, $\mathcal{S}$ returns the re-encryption key to $\mathcal{A}$.
- **Challenge:** $\mathcal{A}$ sends two different plaintexts m_0 and m_1 to $\mathcal{S}$. Then, $\mathcal{S}$ randomly chooses $\beta \in \{0, 1\}$ and a string s^*, and then returns $C^* = \textbf{Enc}(\mathcal{MPK}, ID^*, s^*, m_\beta)$ to $\mathcal{A}$.
- **Phase 2:** $\mathcal{A}$ continuously makes queries defined in **Phase 1**.
- **Guess:** $\mathcal{A}$ guesses $\beta' \in \{0, 1\}$ and sends it to $\mathcal{S}$. $\mathcal{A}$ wins the game if $\beta' = \beta$.

The advantage of $\mathcal{A}$ to win the game is defined as

$$Adv^{IND\text{-}sID\text{-}CPA}(\mathcal{A}) = \left| Pr[\beta = \beta'] - \frac{1}{2} \right|. \tag{3}$$

One can observe that the event $\beta = \beta'$ happening means the adversary learning the information about the encrypted message. Note that the event happens with probability at least $\frac{1}{2}$. This is why the advantage of the adversary in winning the game is defined as formula (3). We illustrate the game in Figure 4. A scheme is said to be IND-sID-CPA secure if there is no PPT adversary that can win the IND-sID-CPA game with non-negligible advantage.

3 RELATED WORKS

This section introduces three HPRE schemes and two multireceiver PRE schemes. In addition, Section 6 compares them against the proposed IMHPRE scheme with respect to various properties, particularly ciphertext length and computation cost.

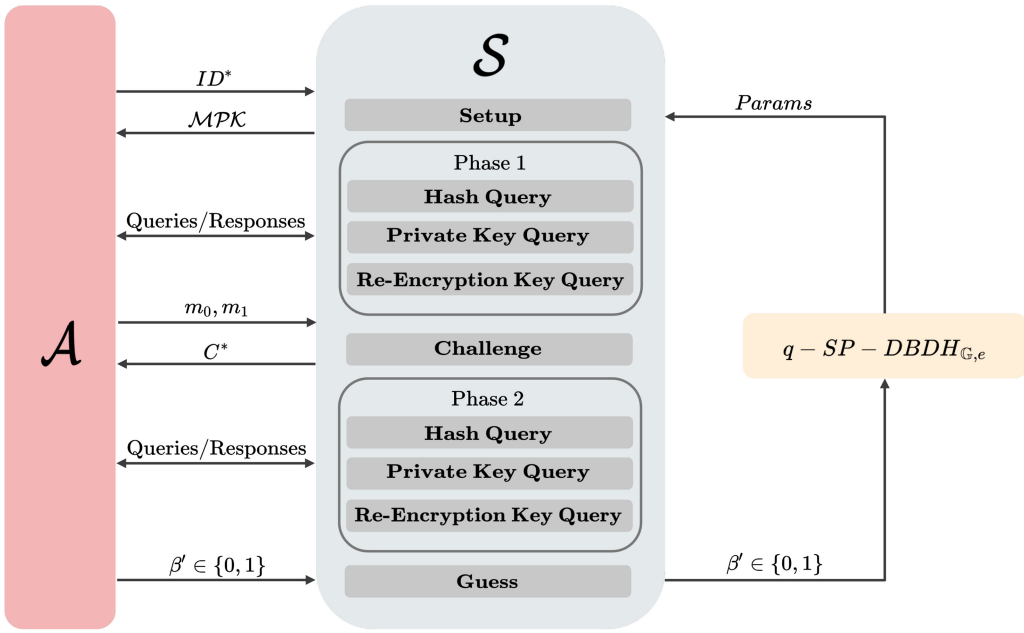


Fig. 4. The IND-sID-CPA game.

3.1 Homomorphic Proxy Re-Encryption

In 2018, Nayak and Tripathy proposed an HPRE scheme called SEMKC [32]. The scheme utilizes two noncolluding servers for secure homomorphic evaluations. In addition, the used re-encryption key is bidirectional, allowing for the conversion of the user's ciphertext to the server's ciphertext and vice versa. However, the presence of two servers and two instances of ciphertext conversion result in a greater system overhead.

In 2019, Kawai et al. proposed an HPRE scheme called G-HPRE [21]. The scheme can achieve CCA2 security for the original ciphertext and CPA security for the re-encrypted ciphertext in the standard model. G-HPRE also supports homomorphic evaluations performed on the ciphertext encrypted under the same receiver's public key. However, when multiple receivers are present, separate ciphertext conversion and homomorphic evaluation are required for each receiver. This results in considerable computational and communication overhead, making it inflexible in practice.

In 2019, Nateghizad et al. proposed an HPRE scheme called HOPE [31]. The scheme is based on PRE cryptography and the BCP cryptosystem [7]. Although the scheme can efficiently perform homomorphic evaluations on the ciphertext re-encrypted from multiple data sources, it has the same drawback as the scheme of Kawai et al. [21]. Furthermore, the size of the evaluated ciphertext increases with the number of ciphertext instances involved in the homomorphic evaluation. This results in greater communication overhead in the system and complicates the decryption of the receivers, making it impractical.

3.2 One-to-Many Proxy Re-Encryption

In 2015, Xu et al. proposed a multireceiver PRE scheme called CIBPRE [41]. The scheme can re-encrypt the ciphertext of a set of receivers to the ciphertext of another set of receivers. Although the scheme has some advantages, such as a constant re-encrypted ciphertext size, it requires extra

modular exponentiation operations for decryption. Since modular exponentiation is a computationally expensive algebraic operation, decryption can be a costly process for the receivers.

In 2020, Maiti and Misra proposed a multireceiver PRE scheme called P2B [27]. The scheme can convert one user's ciphertext to a set of users' ciphertext. In practice, P2B utilizes the Lagrange interpolation polynomial to satisfy the multireceiver requirements. Taking advantage of the Lagrange interpolation polynomial, receivers in the proposed scheme can decrypt the ciphertext without knowing who a receiver is, which reduces some overhead. However, the authors constructed a polynomial by incorrectly multiplying elements in $\mathbb{G}$ and $\mathbb{Z}_p^*$, which made the scheme unworkable. Furthermore, similar to the CIBPRE scheme of Xu et al. [41], P2B does not allow for homomorphism for the re-encrypted ciphertext from different data providers. Note that although the proposed IMHPRE scheme is based on these one-to-many PRE schemes, it avoids the aforementioned problems mentioned and also allows for homomorphism.

4 THE PROPOSED IMHPRE SCHEME

This section introduces the proposed IMHPRE and applies it to FL. The IMHPRE can execute additive homomorphic operations and one-to-many re-encryption. Specifically, it can directly perform homomorphic addition on the re-encrypted ciphertexts for multiple receivers. IMHPRE can thus greatly benefit an FL system by providing both secure model aggregation and improved access control.

4.1 System Model

The proposed system comprises the following four components.

- **Key Generation Center (KGC):** The KGC is a trusted entity that sets up the system and owns the master public–secret key pair. It generates the private key for each user according to the master secret key and given identity.
- **Cloud Service Provider (CSP):** The CSP possesses large storage space and computational resources. Thus, the CSP can re-encrypt the ciphertext by using the re-encryption key and perform homomorphic computations according to the specific requests. Note that we assume that the CSP is honest-but-curious, which means that it follows the scheme honestly but is curious about more information than it is allowed.
- **Data Provider (DP):** A DP uploads its private data to the CSP. It can also upload a re-encryption key, allowing the CSP to re-encrypt its data to a specific group of users.
- **Data Requester (DR):** A DR requests the re-encrypted data or specific calculation results. After receiving the responses from the CSP, the DR can decrypt them with its private key. Furthermore, in the FL system, a DP must update the local model from the latest global model, which is generated by aggregating the local models from all participants. The aforementioned process is iterated until the global model converges. Therefore, a DP must also receive the global model to update its local model, and it is thus also a DR.

4.2 The IMHPRE Scheme

In this subsection, the proposed IMHPRE scheme is described in detail. It contains eight algorithms: **Setup**, **KeyGen**, **Enc**, **DecO**, **ReKeyGen**, **ReEnc**, **DecR**, and **Add**. And the notations used in the scheme are defined in Table 1.

4.2.1 Setup$(\varphi, \mathcal{M})$. For system initialization, KGC executes the algorithm **Setup**$(\varphi, \mathcal{M})$ as follows.

Table 1. The Notations

Notation	Meaning
φ	a security parameter
$\mathcal{M}$	the maximum number of DRs
p	a large prime number of length φ
$\mathbb{G}$, $\mathbb{G}_T$	two multiplicative groups of prime order p
e	a bilinear map, where $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$
g, h, u	three generators of group $\mathbb{G}$
α	a secret value chosen by KGC
H_1	a one-way hash function
$\mathcal{MPK}$	the master public key
$\mathcal{MSK}$	the master secret key
ID_i	the identity of the i-th user
sk_{ID_i}	the private key of ID_i
$DLP()$	the process of solving the discrete logarithm problem
m	a plaintext
s_i	the secret value chosen by ID_i for re-encryption
CT	an original ciphertext
CT_r	a re-encrypted ciphertext
CT_{Add}	an aggregated ciphertext
S	a chosen group of DRs
X	an element randomly chosen from group $\mathbb{G}_T$
$RK_{ID_i \rightarrow S}$	the re-encryption key for re-encryption from user ID_i to S
r_1, r_2, r_3	three randomly chosen numbers

(1) Determine the security parameter $\varphi \in \mathbb{N}$ and the maximum number $\mathcal{M}$ of DRs.
(2) Construct a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ where $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative groups of prime order p, and $| p | = \varphi$.
(3) Choose three generators $g, h, u \in \mathbb{G}$ and one secret value $\alpha \in \mathbb{Z}_p^*$.
(4) Compute $v = e(g, h)$ and $z = e(u, h)$.
(5) Compute h^α and u^α.
(6) Choose a one-way hash function H_1: $\{0, 1\}^* \rightarrow \mathbb{Z}_p^*$.
(7) Output the master public key $\mathcal{MPK} = \{\mathbb{G}, \mathbb{G}_T, e, v, z, g, h, h^\alpha, u, u^\alpha, H_1\}$ and the master secret key $\mathcal{MSK} = \alpha$.

Afterward, KGC broadcasts $\mathcal{MPK}$ to all participants and stores $\mathcal{MSK}$ secretly.

4.2.2 KeyGen$(\mathcal{MSK}, ID_i)$. As mentioned in Section 4.1, KGC with the master public/secret key pair can generate the private key for a given identity. It executes algorithm **KeyGen** as follows.

(1) Input $\mathcal{MSK}$ and the i-th user's identity ID_i.
(2) Output the private key $sk_{ID_i} = g^{\frac{1}{\alpha + H_1(ID_i)}}$.

Next, KGC sends sk_{ID_i} to the user ID_i so that the user can apply it for decryption.

4.2.3 Enc$(\mathcal{MPK}, ID_i, s_i, m)$. Taking advantage of the cloud server's storage and computation capability, a DP, say ID_i, usually uploads data to CSP. To protect data confidentiality, ID_i can encrypt data in advance by executing **Enc** as follows.

(1) Input $\mathcal{MPK}$, ID_i, the secret value s_i, and a plaintext m.
(2) Randomly choose a number $r_1 \in \mathbb{Z}_p^*$.

(3) Compute $c_1 = \left(h^{\alpha} \cdot h^{H_1(ID_i)}\right)^{r_1} = h^{r_1(\alpha + H_1(ID_i))}$.
(4) Compute $c_2 = v^{r_1} \cdot z^m$.
(5) Compute $c_3 = \left(u^{\alpha} \cdot u^{H_1(ID_i)}\right)^{r_1} \cdot u^{s_i} = u^{r_1(\alpha + H_1(ID_i))} \cdot u^{s_i}$.
(6) Output the original ciphertext $CT = \{c_1, c_2, c_3\}$.

Note that the values of s_i and u^{s_i} are stored as secrets used for re-encryption.

4.2.4 DecO$(\mathcal{MPK}, sk_{ID_i}, CT)$. A DP ID_i with the corresponding private key can execute **DecO** to decrypt its uploaded ciphertext CT as follows.

(1) Input $\mathcal{MPK}$, sk_{ID_i}, and $CT = \{c_1, c_2, c_3\}$.
(2) Compute $w = e(sk_{ID_i}, c_1)$.
(3) Output the plaintext $m' = DLP(c_2 \cdot w^{-1})$ by performing Pollard's method [33].

4.2.5 ReKeyGen$(\mathcal{MPK}, ID_i, sk_{ID_i}, s_i, S)$. For sharing data with other users, a DP ID_i can establish a re-encryption key by executing **ReKeyGen** as follows. Specifically, the key is built based on the DP's private key and DRs' public identities. And it can transform the DP's ciphertext into DRs' ciphertext without leaking any information about data content and the involved private key.

(1) Input $\mathcal{MPK}$, ID_i, sk_{ID_i}, s_i, and DRs' identities $\mathcal{S} = \{ID_1^*, ID_2^*, \ldots, ID_n^*\}$.
(2) Choose $r_2, r_3 \in \mathbb{Z}_p^*$, and $X \in \mathbb{G}_T$ at random.
(3) For each $ID_i^* \in S$, compute $y_i = \left(h^{\alpha} \cdot h^{H_1(ID_i^*)}\right)^{r_2} = h^{r_2(\alpha + H_1(ID_i^*))}$.
(4) Compute $rk_1 = v^{r_2} \cdot X$.
(5) Compute $rk_2 = sk_{ID_i} \cdot u^{r_3}$.
(6) Compute $rk_3 = h^{r_3}$.
(7) Compute $rk_4 = z^{-s_i r_3} \cdot X$.
(8) Set $rk_5 = \{y_1, y_2, \ldots, y_n\}$.
(9) Output the re-encryption key $RK_{ID_i \to S} = \{rk_1, rk_2, rk_3, rk_4, rk_5\}$.

Finally, DP ID_i sends $RK_{ID_i \to S}$ to CSP so that CSP can convert the DP's ciphertext to the ciphertext of the receivers included in the set S.

4.2.6 ReEnc$(\mathcal{MPK}, RK_{ID_i \to S}, CT)$. To meet the data-sharing requirements, CSP can execute the algorithm **ReEnc** to re-encrypt the original ciphertext based on the re-encryption key received from DP ID_i as follows.

(1) Input $\mathcal{MPK}$, $RK_{ID_i \to S} = \{rk_1, rk_2, rk_3, rk_4, rk_5\}$, and $CT = \{c_1, c_2, c_3\}$.
(2) Let $c_{r1} = rk_1$.
(3) Compute $c_{r2} = c_2 \cdot e(rk_2, {c_1}^{-1}) \cdot e(c_3, rk_3) \cdot rk_4$.
(4) Let $c_{r3} = rk_5$.
(5) Output the re-encrypted ciphertext $CT_r = \{c_{r1}, c_{r2}, c_{r3}\}$.

Afterward, CSP sends CT_r to the DRs in S.

4.2.7 DecR$(\mathcal{MPK}, sk_{ID_j}, CT_r)$. As mentioned in Section 4.1, a DR, say ID_j, requires the re-encrypted data or specific calculation results for potential uses. After receiving the response CT_r from CSP, DR ID_j can decrypt the content by executing **DecR** as follows. In particular, DR ID_j can directly perform the decryption using its private key without knowing the other receivers' identities.

(1) Let ID_j be the j-th receiver in S.
(2) Input $\mathcal{MPK}$, sk_{ID_j}, and $CT_r = \{c_{r1}, c_{r2}, c_{r3}\}$.

(3) Retrieve y_j from $c_{r3} = \{y_1, y_2, \ldots, y_n\}$.
(4) Compute $w' = e(sk_{ID_j}, y_j)$.
(5) Compute $X' = c_{r1} \cdot w'^{-1}$.
(6) Output the plaintext $m'' = DLP(c_{r2} \cdot X'^{-1})$ by performing Pollard's method [33].

4.2.8 Add$(CT_{r_1}, CT_{r_2}, \ldots, CT_{r_t})$. CSP can execute **Add** to add up the data embedded in t ciphertexts without decrypting them in advance by performing the following operations.

(1) Input the re-encrypted ciphertexts $\{CT_{r_1}, CT_{r_2}, \ldots, CT_{r_t}\}$ from t different DPs.
(2) Parse each $CT_{r_i} \rightarrow \{c_{r1_i}, c_{r2_i}, c_{r3_i}\}$, for $i = 1, 2, \ldots, t$.
(3) Compute $c_{r1}{}' = \prod_{i=1}^{t} c_{r1_i}$.
(4) Compute $c_{r2}{}' = \prod_{i=1}^{t} c_{r2_i}$.
(5) Set $c_{r3}{}' = \{y_1', y_2', \ldots, y_n'\} = \{\prod_{i=1}^{t} y_{1_i}, \prod_{i=1}^{t} y_{2_i}, \ldots, \prod_{i=1}^{t} y_{n_i}\}$ where $c_{r3_i} = \{y_{1_i}, y_{2_i}, \ldots, y_{n_i}\}$ for $i = 1, 2, \ldots, t$.
(6) After performing the homomorphic addition computation as above, output the ciphertext $CT_{Add} = \{c_{r1}{}', c_{r2}{}', c_{r3}{}'\}$.

Finally, CSP sends CT_{Add} to DRs in S. Note that each DR ID_j can decrypt it by executing the algorithm **DecR**$(\mathcal{MPK}, sk_{ID_j}, CT_{add})$ as it can always do decryption on a ciphertext with the format $(v^r X, z^m X, \{y_1, y_2, \ldots y_n\})$. In addition, although Pollard's kangaroo method [33] can only compute a discrete logarithm efficiently when the plaintext is less than 40 bits, the size is large enough for actual data. Moreover, the proposed IMHPRE scheme can also provide multiplicative homomorphism by replacing z^m with m in the encryption algorithm.

4.3 Correctness

The correctness of the proposed IMHPRE scheme is shown as follows.

— The correctness of the original ciphertext decryption:

$$\begin{aligned} w &= e(sk_{ID_i}, c_1) = e(g^{\frac{1}{\alpha + H_1(ID_i)}}, h^{r_1(\alpha + H_1(ID_i))}) = e(g,h)^{r_1} = v^{r_1}. \\ m' &= DLP(c_2 \cdot w^{-1}) = DLP(v^{r_1} \cdot z^m \cdot v^{-r_1}) = m. \end{aligned} \tag{4}$$

— The correctness of the re-encrypted ciphertext decryption:

$$\begin{aligned} c_{r1} &= rk_1 = v^{r_2} \cdot X, \\ c_{r2} &= c_2 \cdot e(rk_2, {c_1}^{-1}) \cdot e(c_3, rk_3) \cdot rk_4 \\ &= e(g,h)^{r_1} \cdot z^m \cdot e(sk_{ID_i}, h^{-r_1(\alpha + H_1(ID_i))}) \cdot e(u^{r_3}, h^{-r_1(\alpha + H_1(ID_i))}) \cdot e(c_3, rk_3) \cdot rk_4 \\ &= e(g,h)^{r_1} \cdot z^m \cdot e(g^{\frac{1}{\alpha + H_1(ID_i)}}, h^{-r_1(\alpha + H_1(ID_i))}) \cdot e(u^{r_3}, h^{-r_1(\alpha + H_1(ID_i))}) \cdot e(c_3, rk_3) \cdot rk_4 \\ &= z^m \cdot e(u^{r_3}, h^{-r_1(\alpha + H_1(ID_i))}) \cdot e(u^{r_1(\alpha + H_1(ID_i))} \cdot u^{s_i}, h^{r_3}) \cdot rk_4 \\ &= z^m \cdot e(u^{r_3}, h^{-r_1(\alpha + H_1(ID_i))}) \cdot e(u^{r_1(\alpha + H_1(ID_i))}, h^{r_3}) \cdot e(u^{s_i}, h^{r_3}) \cdot rk_4 \\ &= z^m \cdot e(u^{s_i}, h^{r_3}) \cdot e(u,h)^{-s_i r_3} \cdot X \\ &= z^m \cdot X, \\ c_{r3} &= rk_5 = \{y_1, y_2, \ldots, y_n\}, \end{aligned} \tag{5}$$

$$
\begin{aligned}
&y_j = h^{r_2(\alpha+H_1(ID_j))},\\
&w' = e(sk_{ID_j}, y_j) = e(g^{\frac{1}{\alpha+H_1(ID_j)}}, h^{r_2(\alpha+H_1(ID_j))}) = e(g,h)^{r_2} = v^{r_2},\\
&X' = c_{r1} \cdot w'^{-1} = v^{r_2} \cdot X \cdot v^{-r_2} = X,\\
&m'' = DLP(c_{r2} \cdot X'^{-1}) = DLP(z^m \cdot X \cdot X^{-1}) = m.
\end{aligned} \tag{6}
$$

— The correctness of homomorphic addition:
Given the ciphertext $CT_{Add} = \{c_{r1}', c_{r2}', c_{r3}'\}$, we have that

$$
\begin{aligned}
&c_{r1}' = \prod_{i=1}^{t} c_{r1_i} = \prod_{i=1}^{t} (v^{r_{2i}} \cdot X_i) = v^{\sum_{i=1}^{t} r_{2i}} \cdot \prod_{i=1}^{t} X_i,\\
&c_{r2}' = \prod_{i=1}^{t} c_{r2_i} = \prod_{i=1}^{t} (z^{m_i} \cdot X_i) = z^{\sum_{i=1}^{t} m_i} \cdot \prod_{i=1}^{t} X_i,\\
&c_{r3}' = \{y_1', y_2', \ldots, y_n'\} = \left\{\prod_{i=1}^{t} y_{1i}, \prod_{i=1}^{t} y_{2i}, \ldots, \prod_{i=1}^{t} y_{ni}\right\}\\
&\quad = \left\{h^{(\sum_{i=1}^{t} r_{2i})(\alpha+H_1(ID_1))}, h^{(\sum_{i=1}^{t} r_{2i})(\alpha+H_1(ID_2))}, \ldots, h^{(\sum_{i=1}^{t} r_{2i})(\alpha+H_1(ID_n))}\right\}.
\end{aligned} \tag{7}
$$

The decryption of CT_{Add} is successful due to the following.

$$
\begin{aligned}
&y_j = h^{(\sum_{i=1}^{t} r_{2i})(\alpha+H_1(ID_j))},\\
&w'' = e(sk_{ID_j}, y_j) = e\left(g^{\frac{1}{\alpha+H_1(ID_j)}}, h^{(\sum_{i=1}^{t} r_{2i})(\alpha+H_1(ID_j))}\right) = e(g,h)^{\sum_{i=1}^{t} r_{2i}} = v^{\sum_{i=1}^{t} r_{2i}},\\
&X'' = c_{r1}' \cdot w''^{-1} = v^{\sum_{i=1}^{t} r_{2i}} \cdot \prod_{i=1}^{t} X_i \cdot v^{-\sum_{i=1}^{t} r_{2i}} = \prod_{i=1}^{t} X_i,\\
&m_{result} = DLP(c_{r2}' \cdot X''^{-1}) = DLP\left(z^{\sum_{i=1}^{t} m_i} \cdot \prod_{i=1}^{t} X_i \cdot (\prod_{i=1}^{t} X_i)^{-1}\right) = \sum_{i=1}^{t} m_i.
\end{aligned} \tag{8}
$$

4.4 The IMHPRE Scheme in FL

The proposed IMHPRE scheme can be applied to implement the following four functions in an FL system:

— **Local Model Encryption**: A participant in FL can first encrypt its uploaded local model with algorithm **Enc**. Accordingly, information about the uploaded local model cannot be obtained without the corresponding private key.
— **Re-Encryption Key Establishment**: To share local models with others, each participant can execute algorithm **ReKeyGen** using its private key and all participants' identities to establish a re-encryption key and upload it to the cloud server. Crucially, because the key with s_i can be used to re-encrypt all ciphertexts with the same s_i, the key can be re-used to the end of the process.
— **Model Aggregation**: With the re-encryption keys from all participants, the cloud server re-encrypts the received local models to all participants by executing the algorithm **ReEnc**. The cloud server then performs model aggregation on the re-encrypted models through the algorithm **Add** to produce a highly accurate global model.
— **Global Model Revelation**: Finally, leveraging PRE, each participant can execute **DecR** to decrypt the aggregated model with its private key. Note that although the IMHPRE scheme requires participants to solve the discrete logarithm problem during decryption, the

aggregated result with a size smaller than 40 bits can still be revealed efficiently through the use of Pollard's kangaroo method [33].

In the IMHPRE scheme, DPs must have permission to obtain information about the local models. Specifically, IMHPRE grants the FL system the ability to implement homomorphism operations and re-encryption. Thus, the cloud server can aggregate local models from different parties correctly without decrypting them in advance. Furthermore, participants in FL can only decrypt the aggregated result, which preserves model input privacy. Consequently, IMHPRE provides FL with access control and a secure aggregation method, which satisfies the privacy-preserving requirements of FL stated in Section 1.

5 SECURITY PROOF

This section presents the security proof of the proposed IMHPRE scheme according to the security model defined in Section 2. The result shows that the proposed scheme achieves IND-sID-CPA security. It means that attackers cannot obtain any information about the plaintext from a given ciphertext without the corresponding private key, which guarantees the data confidentiality.

Theorem 5.1. *The proposed IMHPRE scheme is IND-sID-CPA secure in the random oracle model if the q-SP-DBDH$_{\mathbb{G},e}$ assumption holds.*

Proof. The proof below adopts contradiction, assuming that the proposed scheme is not IND-sID-CPA secure. If there exists a PPT adversary $\mathcal{A}$ with a non-negligible advantage to win the IND-sID-CPA game, we can construct a polynomial-time algorithm $\mathcal{S}$ that also has the non-negligible advantage in solving the hard problem *q-SP-DBDH*$_{\mathbb{G},e}$.

With given system parameters $Params = \{\mathbb{G}, \mathbb{G}_T, e, \mu, \mu^a, \mu^b, \mu^c, \mu^{c/a}, \ldots, \mu^{c/a^q}, e(\mu,\mu)^{abc}\}$, $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

— **Initialization:** $\mathcal{A}$ targets a challenging identity ID^* and sends it to $\mathcal{S}$. In addition, to keep the record, $\mathcal{S}$ prepares the following four tables:
(1) T_{H_1}: This table stores the hash values that have been queried by $\mathcal{A}$.
(2) T_{sk}: This table stores the private keys that have been queried by $\mathcal{A}$.
(3) T_{rk}: This table stores the re-encryption keys that have been queried by $\mathcal{A}$.
(4) T_{sec}: This table stores the secret value used by ID_i for re-encryption.

— **Setup:** $\mathcal{S}$ executes the algorithm **Setup** to build the system as follows.
(1) Choose random numbers $t, x^*, \tau, x_1, \ldots, x_q \in \mathbb{Z}_p^*$ and $\mathcal{V} \in \mathbb{G}_T$.
(2) Set $h = \mu^a$ and compute $h^\alpha = \mu^t \cdot \mu^{-ax^*}$, which implies $\alpha = t/a - x^*$.
(3) Compute $u = h^\tau = \mu^{a\tau}$ and $u^\alpha = (\mu^\tau)^t \cdot (\mu^{a\tau})^{-x^*}$.
(4) Set $\gamma_i = \mu^{c/a^i}$ and compute $\mathcal{V}_i = e(\mu,\mu)^{bc/a^i}$, for $i = 1, \ldots, q$.
(5) Set $L(x) = \prod(tx + x_i)$ and $g = \mu^{cL(1/a)}$, which can be easily computed by using $\gamma_1, \gamma_2, \ldots, \gamma_q$.
(6) Compute $v = e(g,h) = e(\mu,\mu)^{acL(1/a)}$ and $z = e(u,h) = e(\mu,\mu)^{a^2\tau}$.
(7) Set $\mathcal{MSK} = \alpha$ and $\mathcal{MPK} = \{\mathbb{G}, \mathbb{G}_T, e, v, z, g, h, h^\alpha, u, u^\alpha, H_1\}$, where H_1 is modeled as a random oracle.
Finally, $\mathcal{S}$ sends $\mathcal{MPK}$ to $\mathcal{A}$.

— **Phase 1:** $\mathcal{A}$ makes the following queries.
- Hash query $\mathbb{Q}_{H_1}(ID_A)$: The oracle takes an identity as input and outputs $x^* + x_i$ or x^*. If there is the tuple (ID_A, hid_a) in T_{H_1}, $\mathcal{S}$ returns hid_a to $\mathcal{A}$. Otherwise, $\mathcal{S}$ performs the operations below.

(1) If $ID_A = ID^*$, set $hid_a = x^*$.
(2) If $ID_A \neq ID^*$, set $hid_a = x^* + x_a$.
(3) Add (ID_A, hid_a) to T_{H_1}.
(4) Return hid_a to $\mathcal{A}$.

- Private key query $\mathbb{Q}_{sk}(ID_A)$: The oracle takes an identity as input. If $ID_A = ID^*$, $\mathcal{S}$ returns $\perp$. And if there is the tuple (ID_A, sk_{ID_A}) in T_{sk}, $\mathcal{S}$ returns sk_{ID_A}. Otherwise, $\mathcal{S}$ performs the operations as follows.

(1) Compute $sk_{ID_A} = \mu^{cL_j(1/a)}$ by using $\gamma_1, \gamma_2, \ldots, \gamma_{q-1}$.
(2) Add (ID_A, sk_{ID_A}) to T_{sk}.
(3) Return sk_{ID_A} to $\mathcal{A}$.

- Re-encryption key query $\mathbb{Q}_{rk}(ID_A, R)$: The oracle takes an identity and the receiver set as inputs. If $ID_A = ID^*$, $\mathcal{S}$ returns $\perp$ and aborts. And if there is the triple $(ID_A, R, RK_{ID_A \to R})$ in the table T_{rk}, $\mathcal{S}$ returns $RK_{ID_A \to R}$ to $\mathcal{A}$. Otherwise, $\mathcal{S}$ performs the operations as follows.

(1) Choose two random numbers $r_2, r_3 \in \mathbb{Z}_p^*$ and a random element $X \in \mathbb{G}_T$.
(2) Retrieve the tuple (ID_A, s_A) from T_{sec}.
(3) Compute $y_i = h^{r_2(\alpha + H_1(ID_i^*))}$ for each $ID_i^* \in R$.
(4) Compute $rk_1 = v^{r_2} \cdot X$.
(5) Compute $rk_2 = sk_{ID_A} \cdot u^{r_3} = \mu^{cL_j(1/a)} \cdot u^{r_3}$.
(6) Compute $rk_3 = u^{r_3}$.
(7) Compute $rk_4 = z^{-s_A r_3} \cdot X$.
(8) Set $rk_5 = \{y_1, y_2, \ldots, y_n\}$ and $RK_{ID_A \to R} = \{rk_1, rk_2, rk_3, rk_4, rk_5\}$.
(9) Add the triple $(ID_A, R, RK_{ID_A \to R})$ to T_{rk}.
(10) Return $RK_{ID_A \to R}$ to $\mathcal{A}$.

— **Challenge:** $\mathcal{A}$ selects two different plaintexts m_0 and m_1, and then sends them to $\mathcal{S}$. In response to $\mathcal{A}$, $\mathcal{S}$ randomly chooses $\beta \in \{0, 1\}$ and executes the algorithm **Enc**$(\mathcal{MPK}, ID_*, s^*, m_\beta)$ as follows.

(1) Set a random number r_1 as b/t.
(2) Choose a random number $s^* \in \mathbb{Z}_p^*$ and store the tuple (ID^*, s^*) to T_{sec}.
(3) Issue the query $\mathbb{Q}_{H_1}(ID^*)$ and retrieve the random value x^* from T_{H_1}.
(4) Set $c_1 = \mu^b = \left(\mu^t \cdot \mu^{-ax^*} \cdot \mu^{ax^*}\right)^{b/t} = h^{r_1(\alpha + H_1(ID^*))}$.
(5) As $c_2 = e(g,h)^{r_1} \cdot m_\beta = e(\mu,\mu)^{abcL(1/a)/t} \cdot m_\beta$, expand $L(x) = \sum_{i=1}^{q} p_i X^i$ and compute

$$c_2 = e(\mu,\mu)^{abc \cdot p_0/t} \cdot \prod_{i=1}^{q} e(\mu,\mu)^{bc/a^{i-1} \cdot p_i/t} \cdot m_\beta = \mathcal{V}^{p_0/t} \cdot \prod_{i=1}^{q} \mathcal{V}_i^{p_i/t} \cdot m_\beta.$$

(6) Compute $c_3 = \mu^{b\tau} \cdot \mu^{a\tau s^*} = \left(\mu^{\tau t} \cdot \mu^{-\tau a x^*} \cdot \mu^{\tau a x^*}\right)^{b/t} \cdot \mu^{a\tau s^*} = u^{r_1(\alpha + H_1(ID^*))} \cdot u^{s^*}$.
(7) Set and return $C^* = \{c_1, c_2, c_3\}$ to $\mathcal{A}$.

— **Phase 2:** $\mathcal{A}$ continuously makes queries as in **Phase 1**.
— **Guess:** $\mathcal{A}$ outputs $\beta' \in \{0, 1\}$ to $\mathcal{S}$. If $\beta' = \beta$, $\mathcal{S}$ outputs 1. Otherwise, $\mathcal{S}$ randomly chooses $\hat{\beta} \in \{0, 1\}$ and outputs $\hat{\beta}$.

If $\mathcal{V} = e(\mu,\mu)^{abc}$, $\mathcal{A}$ can obtain the correct ciphertext

$$C^* = \left\{ \mu^b, \left(e(\mu,\mu)^{abc}\right)^{p_0/t} \cdot \prod_{i=1}^{q} \mathcal{V}_i^{p_i/t} \cdot m_\beta, \mu^{b\tau} \cdot \mu^{a\tau s^*} \right\}.$$

Otherwise, $\mathcal{V}$ is an element randomly chosen from $\mathbb{G}_T$. $\mathcal{S}$ simulates the IND-sID-CPA game correctly. If $\mathcal{A}$ wins the IND-sID-CPA game with non-negligible advantage ϵ, $|Pr[\beta' = \beta] - \frac{1}{2}| \geq \epsilon$

under a correct simulation of the game. Let q_H denote the number of hash queries issued by $\mathcal{A}$. And assume that $\mathcal{A}$ issues q times of $\mathbb{Q}_{H_1}(ID_A)$. Hence, we have

$$\begin{aligned} &Pr[\mathcal{A}(\mu, \mu^a, \mu^b, \mu^c, \mu^{c/a}, \ldots, \mu^{c/a^q}, \mathcal{V} = e(\mu,\mu)^{abc}) = 1] \\ &= Pr[Not\ Abort] \cdot Pr[\mathcal{A}\ wins \mid Not\ Abort] \\ &= \left(1 - \frac{1}{q_H}\right)^q \cdot \left(\frac{1}{2} \pm \epsilon\right) \end{aligned} \tag{9}$$

and

$$\begin{aligned} &Pr[\mathcal{A}(\mu, \mu^a, \mu^b, \mu^c, \mu^{c/a}, \ldots, \mu^{c/a^q}, \mathcal{V} \leftarrow \mathbb{G}_T) = 1] \\ &= Pr[Not\ Abort] \cdot Pr[\mathcal{A}\ wins \mid Not\ Abort] \\ &= \left(1 - \frac{1}{q_H}\right)^q \cdot \frac{1}{2}. \end{aligned} \tag{10}$$

Thus,

$$\begin{aligned} &\left| Pr[\mathcal{A}(\mu, \mu^a, \mu^b, \mu^c, \mu^{c/a}, \ldots, \mu^{c/a^q}, e(\mu,\mu)^{abc}) = 1] - Pr[\mathcal{A}(\mu, \mu^a, \mu^b, \mu^c, \mu^{c/a}, \ldots, \mu^{c/a^q}, \mathcal{V}) \right. \\ &\left. \leftarrow \mathbb{G}_T = 1] \right| \geq \left| \left((1 - \frac{1}{q_H})^q \cdot (\frac{1}{2} \pm \epsilon) \right) - \left((1 - \frac{1}{q_H})^q \cdot \frac{1}{2} \right) \right| = \left(1 - \frac{1}{q_H}\right)^q \cdot \epsilon \end{aligned} \tag{11}$$

can be obtained. Note that when $q_H \approx q$, the value of $(1 - \frac{1}{q_H})^q \simeq e^{-1}$ which is non-negligible, where e is the base of the natural logarithm function. Consequently, considering the probability of the game aborting, $\mathcal{S}$ can still solve the q-*SP-DBDH*$_{\mathbb{G},e}$ problem with non-negligible advantage at least $(1 - \frac{1}{q_H})^q \epsilon$ within polynomial time.

6 COMPARISON

This section shows the comparison among the proposed IMHPRE scheme with five other schemes mentioned in Section 3 with respect to various properties, particularly ciphertext length and computational cost.

6.1 Properties Comparison

The proposed scheme provides identity-based multireceiver PRE with homomorphic operations, making it more functional than its counterparts. Table 2 details the properties of the various schemes, where "Non-Interaction" means that the delegator and the delegatee do not communicate with each other when constructing a re-encryption key.

6.2 Ciphertext Length Comparison

This subsection shows the comparison among SEMKC [32], G-HPRE [21], HOPE [31], CIBPRE [41], P2B [27], and the proposed IMHPRE in terms of initial ciphertext length and aggregated ciphertext length. Let $\mathbb{G}$ denote a group of order p where p is a 1,024-bit prime number. Therefore, the size of an element in $\mathbb{G}$, denoted by $|\mathbb{G}|$, is 1,024 bits. Finally, t represents the number of participants in FL. The schemes are compared in Table 3.

Note that since the proposed scheme can perform homomorphic additions on the re-encrypted ciphertexts for multiple receivers directly, the cloud server can generate only one aggregated ciphertext rather than t ciphertexts. Therefore, compared with SEMKC [32], G-HPRE [21] and

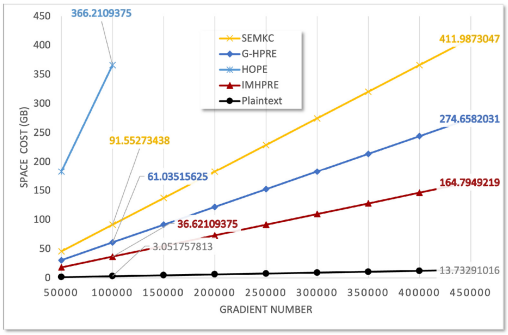
Table 2. Properties Comparison

	SEMKC [32]	G-HPRE [21]	HOPE [31]	CIBPRE [41]	P2B [27]	IMHPRE
ID-Based Setting	No	No	No	Yes	Yes	Yes
Homomorphism	Yes	Yes	Yes	No	No	Yes
Re-Encryption	Yes	Yes	Yes	Yes	Yes	Yes
Multireceiver	No	No	No	Yes	Yes	Yes
Non-Interaction	No	Yes	Yes	Yes	Yes	Yes
Number of Servers	Two	One	One	One	One	One
Security	*	CPA	CPA	CPA	CPA	CPA

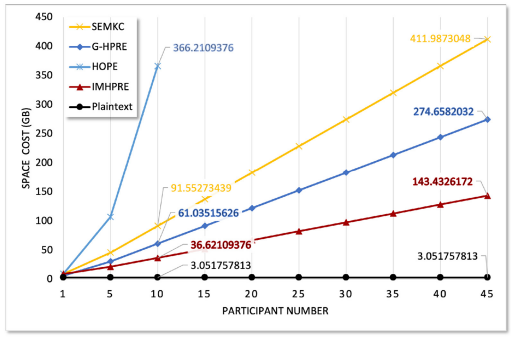
*Security was proved by using the "eal-world and ideal-world" paradigm.

Table 3. Ciphertext Length Comparison

	SEMKC [32]	G-HPRE [21]	HOPE [31]	CIBPRE [41]	P2B [27]	IMHPRE
Initial Ciphertext	$3\|\mathbb{G}\|$	$3\|\mathbb{G}\| + \|p\|$	$2\|p\|$	$4\|\mathbb{G}\|$	$3\|\mathbb{G}\|$	$3\|\mathbb{G}\|$
Aggregated Ciphertext	$3t\|\mathbb{G}\|$	$2t\|\mathbb{G}\|$	$t(3t+2)\|p\|$	NIL	NIL	$(t+2)\|\mathbb{G}\|$



(a) Different numbers of gradient parameters with 10 participants.



(b) Different numbers of participants with 100,000 gradient parameters (≈ the size of a simple FL model on MNIST [42]).

Fig. 5. The space cost.

HOPE [31] and the proposed IMHPRE reduce the ciphertext space significantly. In addition, CIBPRE [41] and P2B [27] do not support ciphertext aggregation because they are without the property of homomorphism.

Figure 5(a) presents the space cost of the FL model for different numbers of gradient parameters when 10 participants are involved. The number of gradient parameters was initially set to be 100,000 because a simple FL model on **Modified National Institute of Standards and Technology** database (**MNIST**) [42] has approximately this number of parameters. As illustrated in Figure 5(a), the IMHPRE scheme has at least 50% lower space cost for 100,000 gradient parameters as compared with SEMKC [32], G-HPRE [21], HOPE [31], CIBPRE [41], and P2B [27]. Furthermore, compared with general FL, the space cost of the IMHPRE scheme increases from approximately 3 GB to 36 GB for 100,000 gradient parameters. However, since cloud storage is becoming cheaper, 36 GB is an acceptable size.

Table 4. Execution Environment

Hardware	Intel(R)Core(TM) i7-4650U CPU @ 1.70 GHz × 4
OS	Linux Ubuntu 16.04 LTS 32 bit
Program Library	MIRACL [29]
Parameter	Tate pairing, Type-A curve

Table 5. The Computation Cost of Each Operation

Notation	Meaning	Cost
T_m	the cost of a modular multiplication	$T_m \approx 0.001$ ms
T_e	the cost of a modular exponentiation	$\approx 240T_m$
T_i	the cost of a modular inversion	$\approx 11.6T_m$
T_{hm}	the cost of a map-to-point hash operation	$\approx 29T_m$
T_p	the cost of a pairing operation	$\approx 87T_m$
T_s	the cost of a scalar multiplication in an additive group or an exponentiation in a multiplicative group	$\approx 29T_m$
T_a	the cost of an addition in an additive group or a multiplication in a multiplicative group	$\approx 0.12T_m$

The results on the space required for different numbers of participants for 100,000 gradient parameters are shown in Figure 5(b). The proposed IMHPRE scheme has 143 GB space cost for 45 participants. Moreover, the space cost of the general FL scheme, in relation to gradient parameter numbers, is approximately 3 GB. Although the proposed IMHPRE scheme has approximately 12 times larger space cost against the general FL scheme, HOPE [31] is with approximately 122 times larger space cost relative to the general FL scheme for 10 participants.

In conclusion, the proposed IMHPRE scheme can provide secure FL for honest-but-curious third-party aggregators, and its space cost is reasonable for real-world applications.

6.3 Computation Cost Comparison

6.3.1 Experimental Setup. Table 4 details the experimental setup, including the hardware, operating system, program library, and parameters. The computation time for a modular multiplication T_m is approximately 0.001 ms. Specifically, Koblitz et al. [22] demonstrated that $T_e \approx 240T_m$, $T_s \approx 29T_m$, and $T_a \approx 0.12T_m$. Furthermore, Chung et al. [11] and James et al. [20] found that $T_{hm} \approx 29T_m$ and $T_i \approx 11.6T_m$. Moreover, according to the experimental results of Cao et al. [9] and Debiao et al. [12], $T_p \approx 3T_s$, which means that $T_p \approx 87T_m$ can be obtained. This subsection compares the schemes with respect to computational cost from the execution of modular multiplication conversions in various cryptographic operations (Table 5).

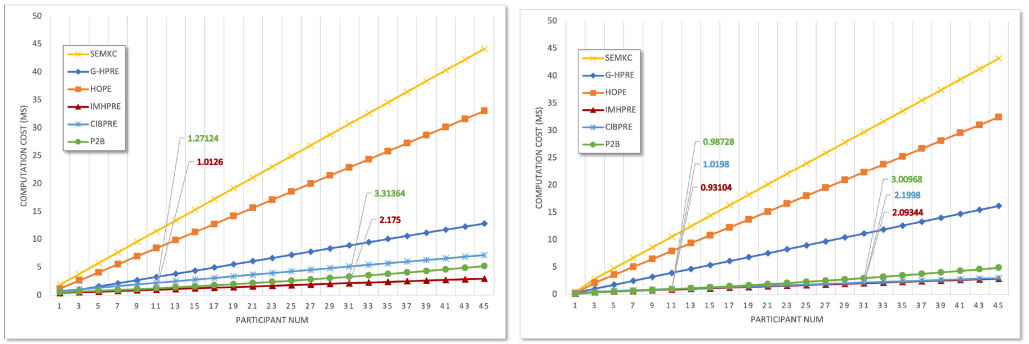
6.3.2 Computation Cost Comparison. Table 6 shows the computation cost of existing HPRE schemes, including SEMKC [32], G-HPRE [21], HOPE [31], and one-to-many PRE schemes, including CIBPRE [41], P2B [27], and the proposed IMHPRE scheme. Although the computational cost of the algorithm **ReKeyGen** increases with the number of participants, it remains reasonably practicable because the algorithm only needs to be executed once. By contrast, the algorithms **Enc**, **ReEnc**, **AGG**, and **DeEval** must be executed repeatedly because the FL steps iterate until model convergence where **AGG** is **Add** and **DeEval** is **DecR** with the parameters replaced by the aggregated ciphertext in the proposed scheme. In particular, **Enc**, **ReKeyGen**, **DeEval**, and **ReDec** are executed locally where **ReDec** can also be performed by **DecR** of the proposed scheme. **ReEnc** and **AGG** are executed in the cloud.

Table 6. Computation Cost Comparison

	Enc	**ReKeyGen**
SEMKC [32]	$4T_e + T_m$ $\approx 961T_m$	$t(2T_e)$ $\approx (480t)T_m$
G-HPRE [21]	$6T_s + 3T_a$ $\approx 174.36T_m$	$t(T_i + T_s)$ $\approx (40.6t)T_m$
HOPE [31]	$2T_m + 2T_e$ $\approx 482T_m$	$t(2T_m + 2T_e)$ $\approx (482t)T_m$
CIBPRE [41]	$(3t + 4)T_s + 3T_a + T_i$ $\approx (87t + 127.96)T_m$	$T_i + (t + 6)T_s + 3T_a$ $\approx (29t + 185.96)T_m$
P2B [27]	$5T_s + 2T_a$ $\approx 145.24T_m$	$(t^2 + t + 1)T_m + T_i + T_{hm} + (2t + 3)T_s + (t + 2)T_a$ $\approx (t^2 + 59.12t + 127.84)T_m$
IMHPRE	$5T_s + 4T_a$ $\approx 145.48T_m$	$T_m + (2t + 4)T_s + (t + 3)T_a$ $\approx (58.12t + 117.36)T_m$
	ReEnc	**ReDec**
SEMKC [32]	$t(2T_e)$ $\approx (480t)T_m$	$t(2T_e)$ $\approx (480t)T_m$
G-HPRE [21]	$t(3T_p + 2T_s + 2T_a)$ $\approx (319.24t)T_m$	$t(2T_m + T_e)$ $\approx (242t)T_m$
HOPE [31]	tT_e $\approx (240t)T_m$	$t(2T_m + T_e)$ $\approx (242t)T_m$
CIBPRE [41]	$(t - 1)T_m + T_i + 2T_p + tT_s + 2T_a$ $\approx (30t + 184.84)T_m$	$(t - 1)T_m + 2T_i + T_{hm} + 3T_p + tT_s + 3T_a$ $\approx (30t + 312.56)T_m$
P2B [27]	$T_p + T_a$ $\approx 87.12T_m$	$tT_m + T_i + T_{hm} + 2T_p + 2T_a$ $\approx (t + 214.84)T_m$
IMHPRE	$2T_p + 3T_a$ $\approx 174.36T_m$	$2T_i + T_p + 2T_a$ $\approx 110.44T_m$
	AGG	**DeEval**
SEMKC [32]	$t(6T_m + 12T_e)$ $\approx 2886T_m$	$(2t)T_e$ $\approx (480t)T_m$
G-HPRE [21]	$t(T_p + 2T_s + 2tT_a)$ $\approx (0.24t^2 + 145t)T_m$	$2T_i + T_s + T_a$ $\approx 52.32T_m$
HOPE [31]	$t(2tT_m)$ $\approx 2t^2T_m$	$(3t + 2)T_m + 2tT_e + (2t + 2)T_i$ $\approx (84.2t + 25.2)T_m$
CIBPRE [41]	NIL	NIL
P2B [27]	NIL	NIL
IMHPRE	$(t^2 + 2t - 2)T_a$ $\approx (0.12t^2 + 0.24t - 0.24)T_m$	$2T_i + T_p + 2T_a$ $\approx 110.44T_m$

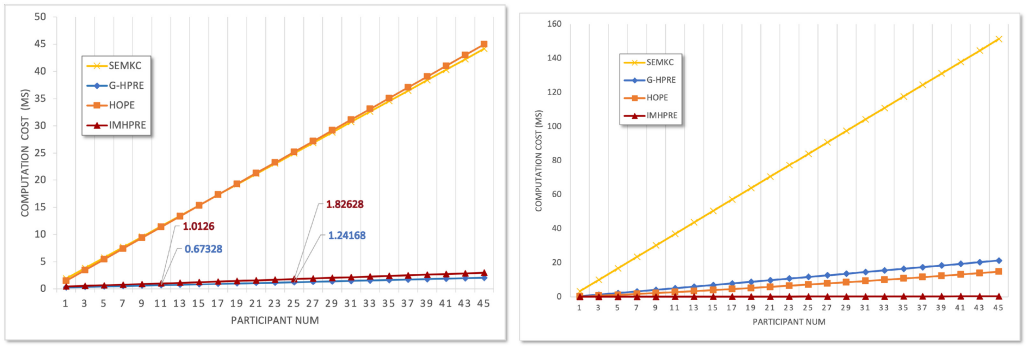
t: the number of participants.
NIL: nothing.

As the proposed scheme is based on the one-to-many PRE scheme, Table 6 lists the computation cost for CIBPRE [41], P2B [27], and IMHPRE. Specifically, CIBPRE [41] is inefficient because expensive modular exponentiation operations are required to process ciphertext containing the information of all receivers. The Lagrange interpolation polynomial adopted in P2B [27] can also incur considerable overhead. Furthermore, P2B [27] is unworkable because it incorrectly multiplies elements in $\mathbb{G}$ and $\mathbb{Z}_p^*$ when constructing the polynomial. Thus, the proposed IMHPRE scheme is generally more efficient than both P2B [27] and CIBPRE [41]. Moreover, the proposed IMHPRE scheme can provide homomorphism functionality to re-encrypted ciphertext for multiple receivers, which is achieved in neither CIBPRE [41] nor P2B [27].



(a) Local Side (Participant)-**Enc**+**ReKeyGen**+**ReDec** (b) Server Side (Aggregator)-**ReEnc**

Fig. 6. The computation cost of proxy re-encryption for participant and aggregator.



(a) Local Side (Participant)-**Enc**+**ReKeyGen**+**DeEval** (b) Server Side (Aggregator)-**ReEnc**+**AGG**.

Fig. 7. The computation cost of homomorphic proxy re-encryption for participant and aggregator.

Figures 6 and 7 detail the results for local and cloud computational cost incurred by PRE and HPRE for different numbers of participants. Compared with other methods, the proposed IMHPRE incurs the least computational expense in a single FL learning iteration. These results are discussed as follows.

Figure 6(a) indicates the total computational time taken at the local side of the execution of **Enc**, **ReKeyGen**, and **ReDec**, and Figure 6(b) does so for **ReEnc** at the server side. The proposed IMHPRE scheme, CIBPRE [41], and P2B [27] incurred a lower local-side and server-side computational expense than do SEMKC [32], G-HPRE [21], and HOPE [31]. Moreover, the computational cost of the re-encryption schemes increases with the number of participants. CIBPRE [41] and P2B [27] cannot execute homomorphism operations to merge the gradients in the ciphertexts. Thus, Figure 7 only illustrates the computational cost of the PRE schemes with homomorphism.

Figure 7(a) illustrates the relationship between the number of participants and total computational time for **Enc**, **ReKeyGen**, and **DeEval**. The proposed IMHPRE scheme and G-HPRE [21] have a much lower local-side computational cost than SEMKC [32] and HOPE [31]. In addition, Figure 7(b) illustrates the proposed IMHPRE also has the lowest server-side computational cost among the schemes.

In summary, as indicated in Figure 8, the proposed IMHPRE scheme is the most computationally efficient at both the local and server sides when executing **Enc**, **ReEnc**, **AGG**, and **DeEval** over one iteration even when the number of participants increases.

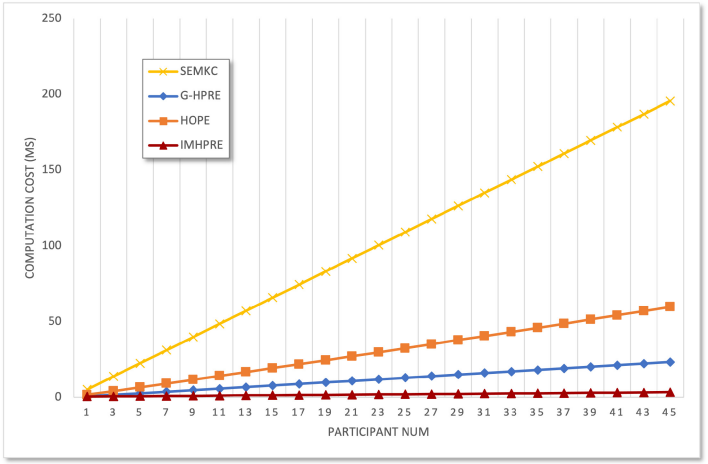


Fig. 8. The computation cost of **Enc**+**ReEnc**+**AGG**+**DeEval**.

6.4 Summary

The proposed IMHPRE scheme outperforms five other schemes terms of ciphertext length, computational cost, functionality, storage space, and computational efficiency. The proposed scheme achieves this through the execution of homomorphic addition on the re-encrypted ciphertext for multiple receivers. Hence, the IMHPRE scheme can be applied to FL systems that require numerous iterations.

7 CONCLUSION

Many works in the literature indicated that classical FL may suffer from poisoning attacks or privacy leak attacks, and the primary solution may lead to performance degradation. The federated learning methods based on HE will not degrade performance, and all computations are performed over ciphertexts. Furthermore, the methods based on HE will be immune to statistical attacks and other potential privacy leakages owing to the premise of security proofs. HE and MKFHE may appear to be suitable solutions, but they are inflexible. HPRE is another solution, but existing HPRE schemes can only re-encrypt the uploaded model for one receiver at a time, which is inefficient in FL because FL usually has multiple participants.

Thus, this study formulated the notion of IMHPRE, which can re-encrypt data for multiple receivers and perform homomorphic addition operations directly. Applied to FL, IMHPRE can aggregate models from different participants without decrypting any local model encrypted by its public key and the participants can obtain the global model using their private keys. Furthermore, we propose an IMHPRE scheme which can be proven secure against chosen-plaintext attacks. We also give a comparison to demonstrate the superiority of the proposed IMHPRE scheme relative to existing schemes. Compared to other works, our scheme is more efficient because the FL training does not stop aggregation until the global model converges.

In the future, we will focus on zero trust architectures. Note that the proposed scheme utilizes a trusted authority KGC to generate private keys for users. However, this renders the KGC a high-value target for attackers and raises some privacy concerns because both users and KGC know the private keys. To overcome such problem would be an interesting and worth-studying issue for privacy preservation in FL.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318.

[2] Taiwan AILabs. 2020. Harmonia: An open-source Federated Learning Framework Taiwan AILabs. (July 2020). Retrieved May 7, 2022 from https://ailabs.tw/healthcare/harmonia-an-open-source-federated-learning-framework/.

[3] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. 2017. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security* 13, 5 (2017), 1333–1345.

[4] Matt Blaze, Gerrit Bleumer, and Martin Strauss. 1998. Divertible protocols and atomic proxy cryptography. In *International Conference on the Theory and Applications of Cryptographic Techniques*. 127–144.

[5] Dan Boneh. 1998. The decision Diffie-Hellman problem. In *International Algorithmic Number Theory Symposium*. Springer, 48–63.

[6] Dan Boneh and Xavier Boyen. 2004. Efficient selective-ID secure identity-based encryption without random oracles. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* Vol. 3027 (2004). Springer, 223–238. https://doi.org/10.1007/978-3-540-24676-3_14

[7] Emmanuel Bresson, Dario Catalano, and David Pointcheval. 2003. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *International Conference on the Theory and Application of Cryptology and Information Security*. 37–54.

[8] Di Cao, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun. 2019. Understanding distributed poisoning attack in federated learning. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS'19)*, 233–239. https://doi.org/10.1109/ICPADS47876.2019.00042

[9] Xuefei Cao, Weidong Kou, and Xiaoni Du. 2010. A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges. *Information Sciences* 180, 15 (2010), 2895–2903.

[10] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Xiaodong Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. arXiv:abs/1712.05526. Retrieved from https://arxiv.org/abs/1712.05526.

[11] Yu Fang Chung, Kuo Hsuan Huang, Feipei Lai, and Tzer Shyong Chen. 2007. ID-based digital signature scheme on the elliptic curve cryptosystem. *Computer Standards & Interfaces* 29, 6 (2007), 601–604.

[12] He Debiao, Chen Jianhua, and Hu Jin. 2011. An ID-based proxy signature schemes without bilinear pairings. *Annals of Telecommunications-Annales des Télécommunications* 66, 11 (2011), 657–662.

[13] Wenxiu Ding, Zheng Yan, and Robert H. Deng. 2017. Encrypted data processing with homomorphic re-encryption. *Information Sciences* 409 (2017), 35–55.

[14] Chun-I Fan, Ling-Ying Huang, and Pei-Hsiu Ho. 2010. Anonymous multireceiver identity-based encryption. *IEEE Transactions on Computers* 59, 9 (2010), 1239–1249. https://doi.org/10.1109/TC.2010.23

[15] Chun-I Fan and Yi-Fan Tseng. 2015. Anonymous multi-receiver identity-based authenticated encryption with CCA security. *Symmetry* 7, 4 (2015), 1856–1881. https://doi.org/10.3390/sym7041856

[16] Chong-zhi Gao, Qiong Cheng, Xuan Li, and Shi-bing Xia. 2019. Cloud-assisted privacy-preserving profile-matching scheme under multiple keys in mobile social network. *Cluster Computing* 22, 1 (2019), 1655–1663.

[17] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*. 169–178.

[18] Ruei-Hau Hsu, Yi-Cheng Wang, Chun-I Fan, Bo Sun, Tao Ban, Takeshi Takahashi, Ting-Wei Wu, and Shang-Wei Kao. 2020. A privacy-preserving federated learning system for android malware detection based on edge computing. In *2020 15th Asia Joint Conference on Information Security (AsiaJCIS'20)*. 128–136. https://doi.org/10.1109/AsiaJCIS50894.2020.00031

[19] Malika Izabachene and David Pointcheval. 2009. New anonymity notions for identity-based encryption. In *Formal to Practical Security*. Springer, 138–157.

[20] Salome James, N. B. Gayathri, and P. Vasudeva Reddy. 2018. Pairing free identity-based blind signature scheme with message recovery. *Cryptography* 2, 4 (2018), 29.

[21] Yutaka Kawai, Takahiro Matsuda, Takato Hirano, Yoshihiro Koseki, and Goichiro Hanaoka. 2019. Proxy re-encryption that supports homomorphic operations for re-encrypted ciphertexts. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 102, 1 (2019), 81–98.

[22] Neal Koblitz, Alfred Menezes, and Scott Vanstone. 2000. The state of elliptic curve cryptography. *Designs, Codes and Cryptography* 19, 2 (2000), 173–193.

[23] Jakub Konečnỳ, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. 1996. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. (1996). Retrieved from https://doi.org/10.48550/ARXIV.1610.02527 arXiv:arXiv:1610.02527.

[24] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. 2012. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th annual ACM Symposium on Theory of Computing*. 1219–1234.
[25] Fucai Luo, Saif Al-Kuwari, Willy Susilo, and Duong Dung. 2020. Chosen-ciphertext secure homomorphic proxy re-encryption. *IEEE Transactions on Cloud Computing* PP (2020), 1–1.
[26] Xu Ma, Cunmei Ji, Xiaoyu Zhang, Jianfeng Wang, Jin Li, Kuan-Ching Li, and Xiaofeng Chen. 2020. Secure multiparty learning from the aggregation of locally trained models. *Journal of Network and Computer Applications* 167 (2020), 102754.
[27] Sumana Maiti and Sudip Misra. 2020. P2B: Privacy preserving identity-based broadcast proxy re-encryption. *IEEE Transactions on Vehicular Technology* 69, 5 (2020), 5610–5617.
[28] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. 1273–1282.
[29] MIRACL. 2018. MIRACL. (2018). Retrieved May 7, 2022 from https://github.com/miracl/MIRACL.
[30] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. 2020. Local and central differential privacy for robustness and privacy in federated learning. (2020).
[31] Majid Nateghizad. 2019. *Efficient Cryptographic Building Blocks for Processing Private Measurements in e-Healthcare*. Ph.D. Dissertation. Delft University of Technology. arXiv:2009.03561. Retrieved from https://doi.org/10.48550/arxiv.2009.03561
[32] Sanjeet Kumar Nayak and Somanath Tripathy. 2018. SEMKC: Secure & efficient computation over outsourced data encrypted under multiple keys. *IEEE Transactions on Emerging Topics in Computing* PP (2018), 1–1.
[33] John M. Pollard. 1978. Monte Carlo methods for index computation (mod p). *Mathematics of Computation* 32, 143 (1978), 918–924.
[34] Bharath K. Samanthula, Yousef Elmehdwi, Gerry Howser, and Sanjay Madria. 2015. A secure data sharing and query processing framework via federation of cloud computing. *Information Systems* 48 (2015), 196–212.
[35] Hossein Shafagh, Anwar Hithnawi, Lukas Burkhalter, Pascal Fischli, and Simon Duquennoy. 2017. Secure sharing of partially homomorphic encrypted IoT data. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. 1–14.
[36] PanJun Sun. 2020. Security and privacy protection in cloud computing: Discussions and challenges. *Journal of Network and Computer Applications* 160 (2020), 102642.
[37] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. 2021. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing* 14, 6 (2021), 2073–2089. https://doi.org/10.1109/TSC.2019.2897554
[38] Yi-Fan Tseng and Chun-I Fan. 2021. Anonymous multireceiver identity-based encryption against chosen-ciphertext attacks with tight reduction in the standard model. *Security and Communication Networks* 2021 (2021).
[39] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. 2019. Eavesdrop the composition proportion of training labels in federated learning. arXiv:abs/1910.06044.
[40] Chen Wu, Xian Yang, Sencun Zhu, and Prasenjit Mitra. 2020. Mitigating backdoor attacks in federated learning. ArXiv:abs/2011.01767.
[41] Peng Xu, Tengfei Jiao, Qianhong Wu, Wei Wang, and Hai Jin. 2015. Conditional identity-based broadcast proxy re-encryption and its application to cloud email. *IEEE Transactions on Computers* 65, 1 (2015), 66–79.
[42] Corinna Cortes Yann LeCun and Chris Burges. 1998. MNIST Handwritten Digit Database. (1998). Retrieved May 7, 2022 from http://yann.lecun.com/exdb/mnist/.
[43] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter L. Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. arXiv:abs/1803.01498.
[44] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in Neural Information Processing Systems* 32 (2019). https://proceedings.neurips.cc/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf.

Received 1 December 2021; revised 13 April 2022; accepted 2 May 2022