Digital Communications and Networks(DCN)

HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/dcan

PEPFL: A Framework for a Practical and Efficient Privacy-Preserving Federated Learning

Yange Chen[ac], Baocang Wang[*abc], Hang Jiang, Pu Duan[d], Benyu Zhang[d], Chengdong Liu[e], Zhiyong Hong[f], Yupu Hu[a]

[a]**State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China**
[b]**Cryptographic Research Center, Xidian University, Xi'an 710071, China**
[c]**China and School of Information Engineering, Xuchang University, Xuchang 461000, China**
[d]**Secure Collaborative Intelligence Laboratory, Ant Group, Hangzhou 310000, China**
[e]**Beijing Application Institute of Information Technology, Beijing 100091, China**
[f]**Facility of Intelligence Manufacture, Wuyi University, Jiangmen 529020, China**

Abstract

As an emerging joint learning model, federated deep learning is a promising way to combine model parameters of different users for training and inference without collecting users' original data. However, a practical and efficient solution has not been established in previous work due to the absence of efficient matrix computation and cryptography schemes in the privacy-preserving federated learning model, especially in partially homomorphic cryptosystems. In this paper, we propose a practical and efficient privacy-preserving federated learning framework (PEPFL). First, we present a lifted distributed ElGamal cryptosystem that can be applied to federated learning and solve the multi-key problem in federated learning. Secondly, we develop a practical partially single instruction multiple data (PSIMD) parallelism scheme that can encode a plaintext matrix into single plaintext to conduct the encryption, improving effectiveness and reducing communication cost in partially homomorphic cryptosystems. In addition, a novel privacy-preserving federated learning framework is designed by using momentum gradient descent (MGD) with a convolutional neural network (CNN) and the designed cryptosystem. Finally, we evaluate the security and performance of PEPFL. The experiment results demonstrate that the scheme is practicable, effective, and secure with low communication and computational costs.

KEYWORDS: federated learning, partially single instruction multiple data, momentum gradient descent, ElGamal, multi-key, homomorphic encryption

1. Introduction

As one of the most promising technologies, federated deep learning has played a significant role in joint learning model and has been widely studied and leveraged in various fields such as image classification [1], medical prediction [2], computer vision [3], and automatic systems [4, 5], which are becoming increasingly popular in applications such as self-driving cars, radiology image processing and cybersecurity threat de-

*B. Wang (Corresponding author) (email: bcwang79@aliyun.com).

[1]Y. Chen and B. Wang* is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, 710071, China; Cryptographic Research Center, Xidian University, Xi'an, 710071, China; School of Information Engineering, Xuchang University, Xuchang, 461000, China (email: ygchen428@163.com; bcwang79@aliyun.com).

[2]H. Jiang and Y. Hu are with the School of Telecommunications Engineering, Xidian University, Xi'an, 710071, China (e-mail: 2250311784@qq.com; yphu@mail.xidian.edu.cn).

[3]P. Duan and B. Zhang are with the Secure Collaborative Intelligence Laboratory, Ant Group, Hangzhou, 310000, China (e-mail: p.duan@antgroup.com; benyu.z@antgroup.com).

[4]C. Liu is with the Beijing Application Institute of Information Technology, Beijing, 100091, China (e-mail: liucd@139.com).

[5]Z. Hong is with the Facility of Intelligence Manufacture, Wuyi University, Jiangmen, 529020, China; Yue-Gang-Ao Industrial Big Data Collaborative Innovation Center, Wuyi University, Jiangmen, 529020, China (e-mail: hzy_wyu@163.com).

tection[6]. In deep learning, for traditional centralized learning, it requires collecting a large amount of user data to train the model. However, user data may contain sensitive private information, which may lead to user information disclosure. Therefore, federated deep learning has aroused widespread concern in industry and academia, and has been proposed to alleviate privacy issues [6, 7].

Although federated learning preserves privacy to a certain extent, researchers [6, 8] have shown that an aggregation server or attackers can still restore some sensitive information by the shared gradients or weights. To prevent sensitive information from being leaked, some investigators proposed new privacy-preserving federated learning frameworks based on different cases or encryption schemes [9, 10]. However, these schemes do not consider practicability owing to users frequently interact with the server online except [11]. Meanwhile, based on fully homomorphic encryption (FHE), privacy-preserving deep learning schemes utilize various single instruction multiple data (SIMD) methods to perform parallel computing, such as vector-based [12] and lattice-based [13] methods. However, FHE data have high inflation rate and high ciphertext homomorphism computational cost, which cannot efficiently support federated multi-party calculation. Therefore, partially homomorphic encryption schemes have been studied in many schemes [8, 9, 11], but parallel matrix computation has not been leveraged in these schemes similar to FHE schemes [12, 13, 14], leading to low effectiveness and high communication costs owing to the encryption of single or vector plaintext. In addition, few schemes have considered the multi-key question in federated learning with efficient multi-key or distributed encryption schemes.

Based on the aforementioned issues, to address the interactivity, parallel computing, and multi-key questions, in this paper, we propose a practical and efficient privacy-preserving federated learning framework (PEPFL), the first privacy-preserving scheme supporting the parallel partially single instruction multiple data (PSIMD) method in partially homomorphic cryptosystems to improve the efficiency. The framework leverages the users' noninteraction method [11] so that the system model becomes practicable and the proposed lifted distributed ElGamal cryptosystem can solve the multi-key problem in federated learning. In summary, the contributions of PEPFL can be summarized as follows:

- **Privacy-preserving framework**. We propose a noninteractive and efficient privacy-preserving collaborative training and prediction framework with momentum gradient descent (MGD) in the federated learning model. By introducing trainers to interact with the aggregation server, the framework can realize user offline training and prediction without online interaction in the federated learning process. The framework protects the privacy of all users and trainers and accelerates convergence with MGD.

- **Cryptography scheme**. We design a novel lifted distributed ElGamal cryptosystem that can realize distributed encryption and decryption while maintaining homomorphism. The cryptosystem avoids collusion issues and solves the multi-user multi-key problem of different public and private key pairs.

- **PSIMD parallel technology**. We present an efficient PSIMD method that can encode a plaintext matrix into a single plaintext for encryption and decode the prediction result plaintext into a plaintext matrix. The method realizes efficient encryption and reduces the communication costs compared with the prior work in partially homomorphic cryptographies.

- **Security analysis and implementation**. We analyze the security of the solution and evaluate the performance of the prototype system. The analysis and experiment results demonstrate that PEPFL is secure and provides effective communication for all users and trainers.

Organization. The remainder of this paper is organized as follows: First, we discuss the related work in more detail in Section II. Then we introduce some preliminaries in Section III. Section IV establishes the system and threat models. Section V describes the details of the PEPFL. Section VI analyzes the security of the constructed framework. The performance analysis and experimental results are shown in Section VII. Finally, we draw the conclusions in Section VIII.

2. Related Work

In this section, we review the existing works on privacy-preserving deep learning from several aspects. Existing schemes are mostly studied according to three technologies: differential privacy, secure multi-party computation (SMC), and homomorphic encryption (HE). For instance, Zhao *et al.* [15] proposed a privacy-preserving collaborative deep learning model with unreliable participants that exploits a function mechanism to protect the local data privacy of participants with differential privacy. Phong *et al.* [8] presented a privacy-preserving deep learning model that uses additively homomorphic encryption to realize asynchronous federated learning. However, it is impractical to leverage the same private key without considering the multi-key question. Xu *et al.*

[6]https://searchenterpriseai.techtarget.com/feature/Deep-learning-and-neural-networks-gain-commercial-footing

[16] proposed a privacy-preserving federated learning approach (HybridAlpha) based on SMC protocol of function encryption and differential privacy. Truex *et al*. [17] developed a privacy-preserving federated learning hybrid approach with differential privacy, SMC, and threshold HE. Through relevant literature research, we conduct a detailed analysis of the following aspects:

Focusing on SIMD, Liu *et al*. [13] proposed an oblivious neural network (MiniONN) combining additive HE with SIMD, secret sharing, and oblivious transfer to support privacy-preserving prediction. However, it has a high computation cost for clients. Subsequently, Juvekar *et al*. [14] presented a secure neural network inference (GAZELLE) to realize lower latency than [13] and designed a lattice-based SIMD homomorphic encryption scheme with optimized encryption switching protocols. However, this model is unfit for nonlattice cryptosystems. Xie *et al*. [12] proposed a secure deep neural network inference scheme (BAYHENN) combining Bayesian deep learning and HE. BAYHENN adopts a vectorizable homomorphic encryption scheme (SIMD) with the encoding function and BFV FHE, but it does not consider the security of the communication process owing to the activation function returned to the user in plaintext. These schemes containing SIMD technologies, which cannot be utilized in existing privacy-preserving federated learning schemes with partially HE, are leveraged in FHE or the cryptosystems based on the lattice. In addition, FHE cannot be used in large-scale federated learning or neural networks due to its ciphertext characteristics and multiple interactions under multiple keys.

Focusing on multi-key privacy-preserving schemes, Liu *et al*. [18] presented a distributed two-trapdoor public-key cryptosystem to reduce the associated key management costs. However, a mistake in the parameter setting was indicated by Li *et al.* [19], and the strong private key employed by the cloud service provider (CSP) reduces security. Li *et al*. [20] proposed a multi-key privacy-preserving deep learning framework in cloud computing that employs multi-key fully homomorphic encryption (MK-FHE) and a double decryption mechanism. However, MK-FHE has more interactions and low efficiency. Ma *et al*. [21] developed a multi-key privacy-preserving deep learning model (PDLM) with a distributed two-trapdoor public-key cryptosystem (DT-PKC). However, it has low efficiency and low classification accuracy. Chen *et al*. [22] proposed multi-key variants of FHE with packet ciphertexts applied to oblivious neural network inference. These schemes utilize DT-PKC or FHE which have some limitations as noted above; moreover, the multi-key question is not considered in federated learning.

Focusing on stochastic gradient descent (SGD), Zhang *et al*. [23] proposed an elastic averaging stochastic gradient Descent (EASGD) algorithm with both synchronous and asynchronous models. However, the algorithm requires all users to own the entire dataset and participate in the entire model training, without considering the privacy issue or communication costs. To improve the convergence rate, momentum term with momentum gradient descent (MGD) was introduced [24, 25]. Liu *et al*. [26] proposed momentum federated learning (MFL) to accelerate convergence using MGD. However, this method does not consider the privacy issue. Some schemes [11, 27] considered privacy in the model training process, but they only leverage SGD without considering momentum. To solve the above problems, we design a multi-key and noninteractive privacy-preserving federated learning scheme with PSIMD and MGD.

3. Preliminaries

3.1. *Secure Multiparty Computation (SMC)*

Secure multiparty computation (SMC) is a set of cryptographic protocols proposed to realize secure computing that solves the problem of jointly calculating a function between some untrustworthy participants. The technique was first introduced in Yao's millionaires' problem in 1982 [28].

Specifically, P participants maintain the privacy information $x_1, x_2, \ldots, x_P$ and compute and share the information of a given function $Y = F(x_1, x_2, \ldots, x_P)$ jointly while preserving the privacy of the private key sk_i. The protocol can securely ensure that honest participants obtain the correct results, even if the participants (fewer than or equal to $n-1$ participants) conspire with each other.

3.2. *(Lifted) ElGamal Cryptosystem*

The lifted ElGamal or ElGamal cryptosystem [29] encrypts and decrypts the message m and can satisfy the additive and multiplicative homomorphic properties respectively.

1) **Key generation**: Given the large primes p and p' satisfying $p = 2p' + 1$. Choose a generator $h \in \mathbb{Z}_p^*$ such that $g = h^2$, and pick a private key x randomly from $\mathbb{Z}_{p'}^* = \{1, 2, \ldots, p' - 1\}$. Then it computes the public key y as follows:

$$y = g^x. \tag{1}$$

2) **Encryption**: To encrypt a message m, it first chooses a number r randomly from $\mathbb{Z}_{p'}^*$. Then a ciphertext $C = E(m) = (A, B)$ can be calculated as

$$A = g^r, B = g^m \cdot y^r (B = m \cdot y^r). \tag{2}$$

3) **Decryption**: On input a ciphertext (A, B), the message $g^m(m)$ can be obtained:

$$g^m = \frac{B}{A^x} (m = \frac{B}{A^x}). \tag{3}$$

In formula (2)(3), the formula outside the brackets is the lifted ElGamal cryptosystem, and the formula inside the brackets is the ElGamal cryptosystem. In addition, because the message m is small in size, m can be extracted from g^m by replace Pollard's Rho method [30] with the lookup table method.

Homomorphic property: Given the ciphertexts $E(m_1)$ and $E(m_2)$ of two messages m_1 and m_2 with the random numbers r_1 and r_2, respectively. When $E(m_1) = (g^{r_1}, g^{m_1}y^{r_1})$ and $E(m_2) = (g^{r_2}, g^{m_2}y^{r_2})$, the followed nature is satisfied:

1) Additive homomorphism. The ciphertext $E(m_1 + m_2)$ can be calculated as follows:

$$E(m_1 + m_2) = (g^{r_1+r_2}, g^{m_1+m_2}y^{r_1+r_2}). \tag{4}$$

2) Multiplication nature:

$$E(m_1)^{m_2} = (g^{r_1m_2}, g^{m_1m_2}y^{r_1m_2}). \tag{5}$$

When $E(m_1) = (g^{r_1}, m_1y^{r_1})$ and $E(m_2) = (g^{r_2}, m_2y^{r_2})$, the multiplicative homomorphism is followed:

$$E(m_1 \cdot m_2) = (g^{r_1+r_2}, m_1m_2y^{r_1+r_2}) \tag{6}$$

3.3. *Lifted Distributed ElGamal Cryptosystem (LDEC)*

In early research, the distributed ElGamal cryptosystem leveraged in [31, 32] can only be utilized on P servers without collusion. According to the ElGamal [29] and distributed ElGamal cryptosystems [31], we propose a lifted distributed ElGamal cryptosystem with P ($P \neq 1$) users substituting P server. The schemes can be described as follows:

Key Generation: Given a multiplication cyclic group $\mathbb{G}$ of prime order p with a generator g, each user U_i chooses a private key x_i randomly from $\mathbb{Z}_p^*$, and calculates and sends the public key $y_i = g^{x_i}$ to the server. Then the server computes the associated public key $y = \prod_{i=1}^{P} y_i = g^{\sum_{i=1}^{P} x_i}$.

Encryption: To obtain a plaintext vector $m \in \mathbb{G}$ and the associated public key y, P users share a random number r jointly from $\mathbb{Z}_p^*$ with SMC [33]. Then each user U_i outputs a ciphertext $C = E(m) = (A, B)$, where $A = g^r, B = g^m \cdot y^r$.

Decryption: On inputs a ciphertext (A, B), each user U_i generates a random number r_i, and calculates and sends A^{r_i} to the server, which avoids the collusion question caused by just sending A^{x_i} for servers in [32], because of $g^m = B/(A^{x_1} \cdot A^{x_2} \cdots A^{x_P}) = B\Big/A^{\sum_{i=1}^{P} x_i}$, namely P servers collude, causing users information leakage. The server computes $x_i = \sum_{j=1, j\neq i}^{P} x_j$ with SMC [33], then it calculates and sends $A^{r_i} \cdot A^{\sum_{j=1,j\neq i}^{P} x_j}$ to each user U_i. The message g^m on each user U_i can be obtained by:

$$g^m = \frac{B}{A^{x_i - r_i} A^{r_i} A^{\sum_{j=1,j\neq i}^{P} x_j}}. \tag{7}$$

Utilizing the lookup table method, the plaintext m can be output. Moreover, the scheme still satisfies the homomorphic property similar to the lifted ElGamal cryptosystem.

3.4. *Federated deep learning*

Federated learning provides an approach that can jointly train the model without sending user data to an aggregation server, which can realize the purpose of federated training and prediction. The method uses a federated averaging algorithm that was proposed by McMahan *et al*. [7] from Google.

In federated learning, to realize the optimization weight $W^* = \arg\min L(W)$, the model needs to be trained to acquire the minimize $L(W)$. Specifically, the trainer T_k has a training dataset $D_k = \{(\chi_k, y_k)\}_{k=1}^{M}$, where χ_k and y_k are the input data and the true label respectively and M represents the number of trainers. Aiming to obtain the optimization global cost function [7, 34] as follows:

$$\min L(W) \triangleq \sum_{i=1}^{M} \frac{N_i}{N} L_i(W), \tag{8}$$

where W is the model parameter, N_i is the sample size of the ith trainer, N is the total sample size satisfying $N = \sum_{i=1}^{M} N_i$, and $L_i(W)$ is the local cost function of the ith trainer. In our scheme, we leverage the following cross-entropy cost function to train a multi-classification problem as follows:

$$\text{loss}(x_i) = -\log\left(\frac{e^{x_i}}{\sum_j e^{x_j}}\right) = -x_i + \log\left(\sum\nolimits_j e^{x_j}\right). \tag{9}$$

In principle, the optimization question can be solved by SGD [8, 21], but its convergence is slow. MGD introduces the momentum term, which can accelerate the convergence rate compared to SGD [35]. Therefore, our scheme introduces the MGD to accelerate the convergence.

1) Local Update: The local update rule can be written as:

$$\begin{aligned} V_{t+1} &= \mu * V_t + \nabla L(W_{t+1}); \\ W_{t+1} &= W_t - \eta V_{t+1}, \end{aligned} \tag{10}$$

where V_{t+1} is the $(t+1)$th momentum term with the same dimension W_{t+1}, t is the iteration index, μ is the momentum factor, ∇ represents the gradient, η is the learning rate, and W_t is the weight of the tth iteration. Thus, each trainer updates the local weights W by using the above rule and returns the updated weights to the server for aggregation.

2) Global Aggregation: After obtaining all the momentum values V and weights W from the trainers, the aggregation server calculates the aggregation average of momentums and weights as follows.

$$\begin{aligned} \overline{V_{t+1}} &= \sum_{i=1}^{M} \frac{N_i}{N} \cdot V_{t+1}^i; \\ \overline{W_{t+1}} &= \sum_{i=1}^{M} \frac{N_i}{N} \cdot W_{t+1}^i, \end{aligned} \tag{11}$$

where $\overline{V_{t+1}}$ is the average of all momentums and $\overline{W_{t+1}}$ is the average of all weights from M trainers.

4. System and Threat Model

In this section, we give an overview of PEPFL and describe the system and threat models considered in the framework. The architecture of PEPFL is illustrated in Fig. 1.

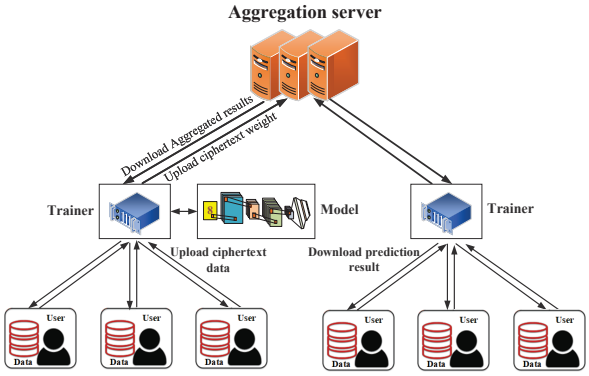


Fig. 1: PEPFL Framework

4.1. *System Model*

In our system model, there are three entities: users, trainers, and the aggregation server. This is different from traditional two entities (users and the aggregation server) and supports users offline. The training model on the trainer leverages convolutional neural network (CNN) to realize training and prediction. Ultimately, the federated training framework can obtain the optimization training parameter and prediction result. All communication lines are guaranteed in the SSL/TLS channel.

User: In the PEPFL system, a user is an entity that possesses a set of data and can provide them with encoded ciphertext for training and prediction in the federated learning task. First, every user generates the public and private key pairs, and then sends the public key to the server for the associated public key. With the associated public key, the user encodes and encrypts its data, and then sends the ciphertext data to a trainer. In addition, the user can decrypt and decode the data to obtain the prediction results.

Trainer: As a machine learning entity, the trainer has a deep learning model to train the model parameters and can collect the encrypted data from the users in the local area network (LAN). Specifically, the trainer can train the model collaboratively with the server by using the federated learning method. Through multiple iterations, the trainer can acquire a well-trained model that can be leveraged for prediction with the users' demand.

Server: As a coordinator and an aggregator, the server can publish the associated public key and calculate the federated average with the aggregated weights from the trainers. In addition, the server needs multiple iterations and cooperation to obtain the optimization model parameters. In our scheme, even if the server colludes with the other entities, it cannot obtain any data information from users and trainers.

4.2. *Threat Model*

In our PEPFL, we consider that the aggregation server and trainers are honest-but-curious entities. Assume that a compromised internal entity from the server or trainer is an adversary, namely, it will honestly follow the protocols but also try to acquire the privacy information from the federated learning system. In addition, the private key and public key of the users are generated by themselves, which avoids leakage of the private key.

1) An adversary compromised by the server can try to retrieve sensitive information from the ciphertext weights and calculate the private key from the public key of the users. The adversary can collude with a trainer or users.

2) An adversary compromised by a trainer can acquire all encrypted data and parameters from the aggregation server or users. In addition, the adversary can collude with the server or users.

3) An adversary compromised by a user tries to obtain sensitive information from encrypted training and prediction data. Moreover, multiple users can collude with each other, or users can collude with the trainers or the aggregation server.

5. Our proposed scheme

In this section, we give a detailed description of PEPFL. First, users provide the encoded ciphertext data to the trainer in the LAN. After collecting sufficient ciphertext data, the trainer trains the model with the initial weights from the server. In our scheme, the ciphertext training process is not the focus of our research. Then the trainer uploads the updated ciphertext momentums and weights to the aggregation server. After the aggregation server finishes the aggregated average, the trainer downloads the aggregated results and iterates them to train the local updating model. When obtaining a well-trained model, the user can predict the data he needs. In the following details, we introduce the PSIMD building block, Equation Test Block (ETB), and homomorphic selection algorithm. Then, we describe the construction of PEPFL in detail.

5.1. *PSIMD Building Block*

In many types of prior work, fully homomorphic encryption (FHE) schemes with SIMD have been leveraged [12, 14, 36], but they incur high communication and computational costs due to the peculiarity of FHE. In partially homomorphic cryptosystems, existing schemes do not utilize SIMD. To improve the effectiveness of partially homomorphic encryption, we

propose a PSIMD scheme to accelerate the model training and improve communication efficiency.

Owing to the high communication costs of image ciphertext in single data or vector data encryption method, to reduce the costs and accelerate the communication efficiency, we present a packed matrix homomorphic encryption scheme, namely, PSIMD, which involves encoding and decoding functions that are capable of packing the matrix into an extensional element in the real number R with the floating-point storage method. The concreted process is followed:

For a matrix A, first, we turn a number $A_{ij}\{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n\}$ of the matrix into the floating-point number which represents the integer part I and base exponent e, i.e., (I, e) with the same e by recalibration. The negative number becomes a positive number greater than the max number which is set as half of n [37]. Due to the data $\{A_{ij}\}$ of a matrix A stores sequentially in a device and a float number occupies four bytes space, to pack the matrix A into a plaintext number $Q = A_{11} \parallel A_{12} \parallel \ldots \parallel A_{mn}$ for encrypting, it needs expending the $m * n$ matrix into $m * n * 4$ bytes linear space as a package by right shift processing. Then it can encrypt the package as a whole to transfer to the server as shown in Fig. 2. The encoding algorithm is denoted as Encode(A), which is as follows in Algorithm 1.

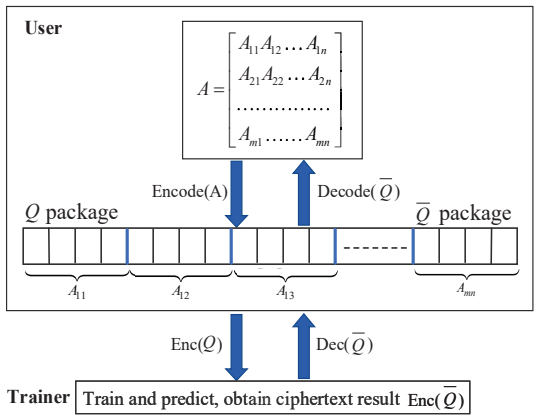


Fig. 2: PSIMD workflow

Algorithm 1: PSIMD Encoding Algorithm

Input: A
Output: $E_{pk}(Q)$
Trainer:
  Perform the operation float(A) satisfying (I, e) form;
  Store the data A_{ij} of the matrix A sequentially;
  Pack the sequential data as a plaintext package Q by right shift processing of the matrix A;
  Encrypt the extended plaintext number Q for $E_{pk}(Q)$.

After obtaining the ciphertext prediction result $E_{pk}(\overline{Q})$, the user decrypts the ciphertext $E_{pk}(\overline{Q})$ to obtain $\overline{Q}$. Then, the result package $\overline{Q}$ is divided into $m \times n$ numbers by every four bytes for a group and is restored into a matrix A'. The decoding algorithm is shown as follows in Algorithm 2.

Algorithm 2: PSIMD Decoding Algorithm

Input: $E_{pk}(\overline{Q})$
Output: A'
Trainer:
  Decrypt $E_{pk}(\overline{Q})$ for $\overline{Q}$;
  Divide $\overline{Q}$ into $m \times n$ numbers by four bytes a group;
  Restore the matrix A' by left shift processing.

Compared with the existing single plaintext encryption or vector encryption, this method greatly improves the encryption speed, reduces the ciphertext length and improves the communication efficiency.

5.2. Equation Test Block (ETB)

To test whether the two plaintexts $\overline{W}$ and W^* of the ciphertext aggregation average $E_{pk}(\overline{W})$ and final global ciphertext model weights $E_{pk}(W^*)$ are equal, we design the following building block.

Given two ciphertexts $E_{pk}(\overline{W}) = (g^{r_1}, g^{\overline{W}} y^{r_1})$ and $E_{pk}(W^*) = (g^{r_2}, g^{W^*} y^{r_2})$, to test if $\overline{W}$ and W^* are equal, we compare $E_{pk}(\overline{W})$ and $E_{pk}(W^*)$ as follows:

$$(\frac{A_1}{A_2}, \frac{B_1}{B_2}) = (g^{r_1 - r_2}, g^{\overline{W} - W^*} y^{r_1 - r_2}). \tag{12}$$

After decrypting the above formula, if $g^{\overline{W} - W^*} = 1$, namely, $\overline{W} - W^* = 0$, the two plaintexts $\overline{W}$ and W^* are equal plaintexts. Otherwise, the two plaintexts are not equal.

In the decryption process, the associated private key $sk = \sum_{i=1}^{P} x_i = \sum_{i=1}^{P} sk_i$, which can be obtained from SMC [38], is needed.

5.3. Homomorphic algorithm selection(HAS)

Since the lifted ElGamal or ElGamal cryptosystem supports additive or multiplicative homomorphism respectively, it is necessary to select an appropriate cryptosystem for homomorphism calculations in the ciphertext training process of the trainers. Therefore, we design the HAS algorithm that satisfies both additive homomorphism and multiplicative homomorphism by homomorphic selection algorithm in Algorithm 3, which is applied in the CNN ciphertext training process of the trainers. In the PEPFL, we adopt the proposed LDEC.

5.4. Construction of PEPFL

In this subsection, we describe the workflow of the scheme in Fig. 3. According to the workflow, the steps of our scheme include the following phases: the initialization phase, the encoding and encryption phase, the model training phase, and the prediction phase.

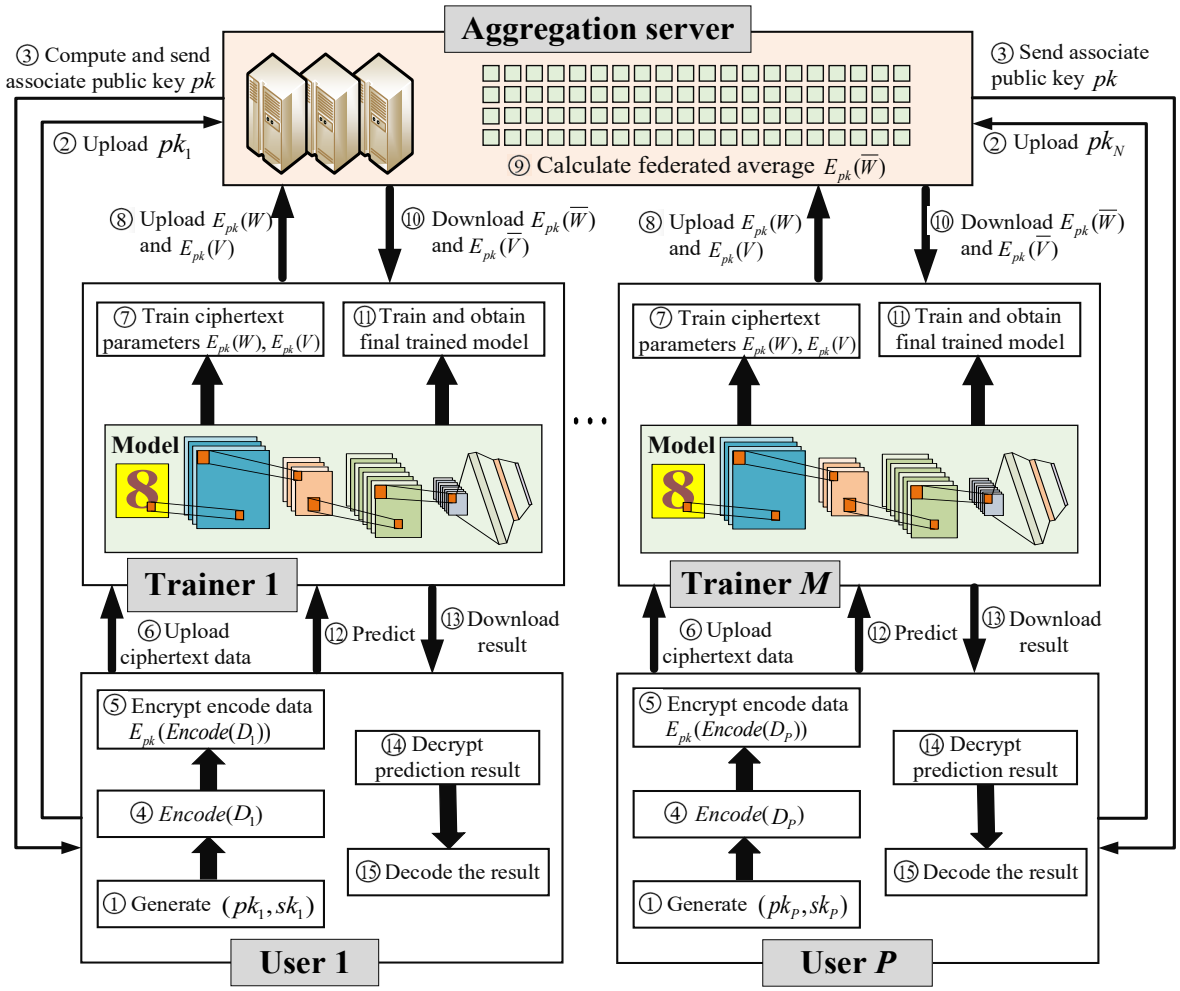


Fig. 3: PEPFL workflow

Initialization phase: Firstly, user U_i generates public and private key pairs (pk_i, sk_i). Then user U_i uploads the public key pk_i to the aggregation server. The server collects the public key pk_i from P users, and then calculates and publishes the associated public key $pk = \prod_{i=1}^{P} pk_i = g^{\sum_{i=1}^{P} x_i}$.

Encoding and encryption phase: After obtaining the associated public key pk, user U_i encodes his data D_i for $Encode(D_i)$ with the PSIMD encoding algorithm and encrypts the encoding data $Encode(D_i)$ for $E_{pk}(Encode(D_i))$ with lifted distributed ElGamal encryption. Then user U_i sends the ciphertext $E_{pk}(Encode(D_i))$ to trainer T_i.

Specifically, first, the user U_i preprocesses and segments the figure data D_i for the matrix data $A_k\{k \in \{1, 2, \ldots, s\}\}$, where s is the number of matrices, namely, the figure of an $a \times b$ matrix, the user divides the matrix into multiple small matrices $A_k = \{A_{mn}^{(k)}\}\{1 \leqslant m \leqslant a, 1 \leqslant n \leqslant b\}$. Second, user U_i encodes every A_k for linear Q_k with $A_{mn}^{(k)}$ which has a four byte floating number. Then, user U_i encrypts the linear Q_k as a whole for $E_{pk}(Q_k)$ with the associated public key pk. Owing to [11] or other partially homomorphic cryptosystem only encrypting a single element $\{A_{mn}\}$ or a vector of a matrix in sequence, while our scheme encrypts equivalent to a matrix, the efficiency of our P-SIMD package method is superior to that of [11]. In addition, user U_i encrypts the label y_i for $E_{pk}(y_i)$.

After obtaining $\{E_{pk}(Q_1)\|E_{pk}(y_1), E_{pk}(Q_2)\|E_{pk}(y_2), \ldots, E_{pk}(Q_s)\|E_{pk}(y_s)\}$, user U_i uploads these ciphertext data to a trainer T_i in local domain via TLS/SSL secure channel.

Model training phase: After obtaining ciphertext data, trainer T_i performs the ciphertext training on the CNN model, which can acquire the ciphertext weight $E_{pk}(W)$ and the ciphertext momentum $E_{pk}(V)$ from the CNN ciphertext training process, and the ciphertext training process referred to other papers [39, 40]. In terms of the MGD algorithm, we can update the ciphertext momentums and weights by using a secure federated average algorithm.

More specifically, first, according to the local update rule, we can calculate the ciphertext momentums and weights for every trainer with the homomorphic method as follows:

$$
\begin{aligned}
&E_{pk}(V_{t+1}^i) \\
=&E_{pk}(\mu \cdot \overline{V_t^i} + \nabla L(W_{t+1}^i)) \\
=&E_{pk}(\overline{V_t^i})^{\mu} \cdot E_{pk}(\nabla L(W_{t+1}^i)).
\end{aligned} \tag{13}
$$

From the above formula, we can gain the ciphertext weight on every trainer:

$$
\begin{aligned}
&E_{pk}(W_{t+1}^i) \\
=&E_{pk}(\overline{W_t^i} - \eta V_{t+1}^i) \\
=&E_{pk}(\overline{W_t^i}) \cdot E_{pk}(V_{t+1}^i)^{-\eta}.
\end{aligned} \tag{14}
$$

After obtaining all $E_{pk}(V_{t+1}^i)$, $E_{pk}(W_{t+1}^i)$ from M trainers, the aggregation server computes the ciphertext federated aggregation average $E_{pk}(\overline{V_{t+1}})$, $E_{pk}(\overline{W_{t+1}})$ as follows.

$$
E_{pk}(\overline{V_{t+1}}) = \sum_{i=1}^{M} \frac{N_i}{N} E_{pk}(V_{t+1}^i); \tag{15}
$$

$$
E_{pk}(\overline{W_{t+1}}) = \sum_{i=1}^{M} \frac{N_i}{N} E_{pk}(W_{t+1}^i). \tag{16}
$$

By continuous iterations of local update and global aggregation, the server can obtain the optimal ciphertext weight, and the privacy-preserving federated learning protocol (PFL) is shown in Algorithm 4, where f is the maximum number of rounds. Compared to the traditional federated averaging algorithm [7], our PFL improves the convergence rate of FL.

The optimal ciphertext weight is estimated by comparing $E_{pk}(\overline{W_{t+1}})$ and $E_{pk}(W^*)$ to realize ciphertext convergence, which judges if $\overline{W_{t+1}}$ and W^* are equal according to ETB or if the number of rounds reaches the maximum iterations value. The ciphertext aggregation algorithm is shown in Algorithm 5.

Prediction phase: After the trainers obtain the well-trained model, the users can realize the prediction on the trainers. First, the user U_i uploads the encrypted data $E_{pk}(\mathbb{Q}_i)$ to a trainer, where $\mathbb{Q}_i$ is encoded data. The trainer predicts the ciphertext data $E_{pk}(\mathbb{Q}_i)$ to attain the ciphertext prediction result $E_{pk}(\mathfrak{R}_i)$. Then, the user U_i downloads the ciphertext result $E_{pk}(\mathfrak{R}_i)$ and decrypts it for $\mathfrak{R}_i$ with the lifted distributed ElGamal decryption algorithm. Finally, the user U_i restores $\mathfrak{R}_i$ for the corresponding matrix by using the PSIMD decoding algorithm.

5.5. Decryption-prevention protocol

In the distributed ElGamal cryptosystem [31, 32], each user U_i sends A^{x_i} to the honest-but-curious server. When the server or an adversary obtains all A^{x_i} from every user, it can perform the calculation $G = A^{x_1} \cdot A^{x_2} \cdot \ldots \cdot A^{x_P} = g^{r \cdot \sum_{i=1}^{P} x_i}$ with the ciphertext $C = (g^r, g^m \cdot y^r)$, the server or the adversary can compute $g^m = g^m \cdot y^r / G$, and then it can obtain the plaintext m. To prevent decryption or attack from occurring, we propose a decryption-prevention protocol, as shown in Algorithm 6. The algorithm avoids the collusion question of multiple users.

Algorithm 3: Homomorphic Selection Algorithm

Input: m_1, m_2
Output: $E(m_1 + m_2), E(m_1 m_2)$
Judging training formula:
  if $m_1 + m_2$ is needed in training process
  then lifted ElGamal cryptosystem;
    given the ciphertext $E(m_1)$, $E(m_2)$,
    calculate $E(m_1 + m_2)$;
  end if
  if $m_1 \cdot m_2$ is needed in training process
  then ElGamal cryptosystem;
    given the ciphertext $E(m_1)$, $E(m_2)$,
    calculate $E(m_1 m_2)$.
  end if

Algorithm 5: Ciphertext Aggregation Algorithm (CAA)

Input: a global ciphertext model parameter $E_{pk}(\overline{W_{t+1}})$, the final global ciphertext model parameter $E_{pk}(W^*)$, the maximum round f
Output: The final global ciphertext model parameter $E_{pk}(W^*)$

Server:
  if the round $t \neq f$
  then continue training the PFL.
  else obtain the ciphertext average parameter $E_{pk}(\overline{W_{t+1}})$ and retreat from the training.
  end if

Trainer:
  After obtaining the ciphertext average weight $E_{pk}(\overline{W_{t+1}})$, in terms of ETB, judging:
  if $\overline{W_{t+1}} \neq W^*$
  then continue training the model on the CNN;
  else obtain the optimal ciphertext weight $E_{pk}(\overline{W_{t+1}})$.
  end if
  Predict the data from users according to the optimal ciphertext model.

Algorithm 6: Decryption-prevention Protocol

Input: A^{x_i}
Output: g^m
Server or Adversary:
  User U_i sends A^{x_i};
  if the number of users $P > 1$
  then it cannot decrypt utilizing lifted ElGamal cryptosystem.
  else it only can decrypt $D(C)$ exploiting LDEC.
  end if
  Output g^m.

Algorithm 4: Privacy-preserving Federated Learning Protocol (PFL)

Input: ciphertext model parameters $E_{pk}(W_i)$, $E_{pk}(V_i)$
Output: the final global ciphertext model weight $E_{pk}(W^*)$

Server:
  Initialize the final global model weight W^*, the initial weight $\overline{W_i}(0)$, the initial momentum $\overline{V_i}(0)$;
  Broadcast the initial parameters $\overline{W_i}(0)$, $\overline{V_i}(0)$ to every trainer;
  for each round t=1, 2, …, f **do**
    for i=1, 2, …, M in parallel **do**
      Each trainer i performs local update according to the following trainer's algorithm in PFL:
      Update $E_{pk}(V^i_{t+1}) \leftarrow E_{pk}(i, \overline{V_t})$; $E_{pk}(W^i_{t+1}) \leftarrow E_{pk}(i, \overline{W_t})$;
      Send $E_{pk}(V^i_{t+1})$, $E_{pk}(W^i_{t+1})$ to Server.
    end for
    Calculate federated aggregation average: $E_{pk}(\overline{V_{t+1}}) = \sum_{i=1}^{M} \frac{N_i}{N} E_{pk}(V^i_{t+1})$; $E_{pk}(\overline{W_{t+1}}) = \sum_{i=1}^{M} \frac{N_i}{N} E_{pk}(W^i_{t+1})$;
    Send $E_{pk}(\overline{V_{t+1}})$, $E_{pk}(\overline{W_{t+1}})$ to Trainer.
  end for

Trainer T_i**:**
  Trainer update $(i, E_{pk}(\overline{W_{t+1}}))$:
  Obtain the newest ciphertext model parameters $(i, E_{pk}(\overline{V_t}))$, $(i, E_{pk}(\overline{W_t}))$;
  for the number of iterations i=1, 2, …, S **do**
    Divide the dataset D_i into the batch size K randomly;
    for the batch number j=1, 2, …, $\frac{D_i}{K}$ **do**
      Calculate the ciphertext gradient $E_{pk}(\nabla L(W^i_{t+1}))$;
      Update the ciphertext momentum $E_{pk}(V^i_{t+1}) = E_{pk}(\overline{V_t})^{\mu} \cdot E_{pk}(\nabla L(W^i_{t+1}))$;
      Update the ciphertext weight $E_{pk}(W^i_{t+1}) = E_{pk}(\overline{W_t}) \cdot E_{pk}({V_{t+1}}^i)^{-\eta}$;
    end for
  end for
  Obtain local ciphertext model parameters $E_{pk}(V^i_{t+1})$, $E_{pk}(W^i_{t+1})$;
  if $E_{pk}(W_{t+1})$ is not an aggregation value judged by CAA in Algorithm 5;
  then send $E_{pk}(V_{t+1})$, $E_{pk}(W_{t+1})$ to Server;
  else
  Quit the update and the training is over.
  end if

6. Security analysis

The security of PEPFL is guaranteed by the ElGamal cryptosystem and lifted distributed ElGamal cryptosystem without random oracles under the decisional Diffie Hellman (DDH) [41] assumption. The ElGamal cryptosystem [29] and threshold cryptosystems [42] are semantically secure provided that the DDH problem is hard.

In this section, we first prove the semantic security and the other security of LDEC. Then, we expound on the security of the input, model, and inference result privacy.

Theorem 1 (Semantic Security). *If the ElGamal cryptosystem is semantically secure, then the LDEC is semantically secure.*

Proof. Knowing $y = g^{\sum_{i=1}^{P} x_i}$ and $A = g^r$, an adversary cannot acquire $y^r = g^{r\sum_{i=1}^{P} x_i}$; that is, the adversary cannot determine $g^c = g^{r\sum_{i=1}^{P} x_i}$ and cannot compute g^m from $B = g^m y^r$ by multiplying the inversion y^{-r}. Then, it cannot obtain m from g^m. Due to the DDH and discrete logarithm hard problem, in terms of the semantic security of ElGamal, the LDEC is semantically secure.

Theorem 2 (Against Collusion and Adversary Security). *If the encryption schemes ElGamal and LDEC are semantically secure, then the security can be guaranteed under collusion or hacking attacks.*

Proof. In the distributed ElGamal cryptosystem [32], each server outputs $A_i = A^{x_i}$, and the plaintext can be obtained by $g^m = B\Big/\prod_{i=1}^{P} A^{x_i}$. When a server colludes with an external adversary or an adversary eavesdrops the ciphertext data B and all A_i, the adversary can obtain the plaintext m from these ciphertext. To avoid collusion or hacking attacks, our proposed LDEC improves A^{x_i} for A^{r_i}, and each user replacing each server in our scenario computes $A^{x_i - r_i}$ with the private key x_i to conduct decryption. When a user colludes with an external adversary, it can leak only the user ciphertext information, which is semantically secure and cannot obtain any sensitive data. Moreover, fewer than $P-1$ users colluding cannot decrypt the ciphertext data, and thus, security can still be guaranteed. When an adversary eavesdrops on these ciphertext informations $B, A^{r_i}, A^{r_i}A^{\sum_{j=1,j\neq i}^{P} x_j}$, he cannot gain the value g^m and then cannot acquire the plaintext m.

Theorem 3 (Input Privacy). *If the LDEC scheme is semantically secure, then the input privacy of PEPFL can be guaranteed.*

Proof. After encoding and encrypting the plaintext data to obtain $\{E_{pk}(Q_1)\|E_{pk}(y_1), E_{pk}(Q_2)\|E_{pk}(y_2), \ldots, E_{pk}(Q_s)\|E_{pk}(y_s)\}$, the user U_i sends these ciphertexts to a trainer. Because LDEC is semantically secure, the input security for trainers is guaranteed: (1) Fewer than $P-1$ users colluding cannot obtain trusted users' information owing to there being no private key to decrypt. (2) An adversary or a participant cannot obtain any information from the ciphertext except for the users themselves. Therefore, the input privacy can be guaranteed.

Theorem 4 (Model Privacy). *If the LDEC scheme is semantically secure, then the model privacy of PEPFL can be guaranteed.*

Proof. In the PFL protocol, all data are trained under the ciphertext, which protects confidentiality in terms of the semantic security of LDEC. We prove the security of the local update and global aggregation according to an ideal world and a real-world game [43].

Lemma 1. *Local update (@Trainer) is secure against a semi-honest adversary $\mathcal{A}_{trainer}$ in the PFL protocol.*

Proof. A challenge trainer can securely communicate with a semi-honest adversary $\mathcal{A}_{trainer}$. When constructing a simulator S, the simulator S replacing the trainer is built to interact with the adversary $\mathcal{A}_{trainer}$. In the real world, the view of $\mathcal{A}_{trainer}$ is:

$$V_{Real} = \{[\overline{V_t}]_{pk}, [\overline{W_t}]_{pk}, [\nabla L(W_{t+1})]_{pk}, [V^i_{t+1}]_{pk}, [W^i_{t+1}]_{pk}\}.$$

In the ideal world, the view of $\mathcal{A}_{trainer}$ is:

$$V_{Ideal} = \{[\overline{r_{11}}]_{pk}, [\overline{r_1}]_{pk}, [r_{12}]_{pk}, [r_{21}]_{pk}, [r_{11}]_{pk}\},$$

where the random numbers $\overline{r_{11}}$, $\overline{r_1}$, r_{12}, r_{21}, r_{11} ϵR.

Owning to the semantic security of LDEC, the adversary $\mathcal{A}_{trainer}$ cannot distinguish the ideal world from the real world:

$$\{IDEAL^{PFL}_{\mathcal{A}_{trainer},S}(V_{Ideal})\} \overset{c}{\approx} \{REAL^{PFL}_{\mathcal{A}_{trainer},trainer}(V_{Real})\}.$$

Lemma 2. *Global aggregation (@Server) is secure against a semi-honest adversary $\mathcal{A}_{Server}$ in the PFL protocol.*

Proof. In global aggregation process for the server, a semi-honest adversary $\mathcal{A}_{Server}$ interacts with the server, its view in the real world is:

$$V_{Real}{}' = \{[\overline{V_t}]_{pk}, [\overline{W_t}]_{pk}, [V^i_{t+1}]_{pk}, [W^i_{t+1}]_{pk}, [\overline{V_{t+1}}]_{pk}, [\overline{W_{t+1}}]_{pk}\}.$$

In the ideal world, a simulator S replacing the server generates the same number of random ciphertexts. The view of $\mathcal{A}_{Server}$ is:

$$V'_{Ideal} = \{[\overline{r_a}]_{pk}, [\overline{r_b}]_{pk}, [r_{a1}]_{pk}, [r_{b1}]_{pk}, [\overline{r_{a1}}]_{pk}, [\overline{r_{b1}}]_{pk}\}.$$

The adversary cannot distinguish the ideal world from the real world:

$$\{IDEAL^{PFL}_{\mathcal{A}_{Server},S}(V'_{Ideal})\} \overset{c}{\approx} \{REAL^{PFL}_{\mathcal{A}_{Server},Server}(V'_{Real})\}.$$

Theorem 5 (Inference Result Privacy). *If the LDEC scheme is semantically secure, then the inference result privacy of PEPFL can be guaranteed.*

Proof. In the prediction phase, user U_i sends the prediction ciphertext $E_{pk}(\mathbb{Q}_i)$ to a trainer, and the trainer trains and obtains the ciphertext prediction result $E_{pk}(\mathfrak{R}_i)$ by exploiting the well-trained model. All the data on the trainer are ciphertext, and thus, the ciphertext result is returned to the user U_i. Due to the semantic security of LDEC, confidentiality and inference result privacy can be guaranteed.

7. Performance Evaluations

In this section, we analyze and compare the performance of PEPFL. First, we discuss the communication and computational costs of PEPFL. Additionally, we evaluate the efficiency and accuracy of our scheme.

7.1. Complexity analysis

Communication cost. In the initialization phase, every user sends $O(\log p)$ communication costs to the server. After computing the associated public key pk, the server takes $O(\log p)$ costs for every user. In the encoding and encryption phase, user U_i sends $O(2s \log p)$ communication costs to trainer T_i. In the local update phase, the trainer T_i generates $O(2u \log p)$ communication costs for the server, where u is the number of the weight encoding packets from the P-SIMD method. After the global aggregation average, the server imposes $O(2u \log p)$ costs on every trainer. When the trainer obtains the well-trained model, users can predict the result they require. In the prediction phase, a user uploads $O(2s \log p)$ costs to a trainer who conducts the well-trained model. After predicting the ciphertext result, the trainer sends $O(\log p)$ cost to the predicted user.

Computational cost. To simplify the notation, we denote point multiplication/division as Mul/Div, encoding/decoding as Ecod/Dcod, encryption/decryption as Enc/Dec, an exponent as Exp, and a gradient as Gra. In the initialization phase, the server takes a computational cost of $(P-1)$Mul to compute the associated public key. In the encoding and encryption phase, it costs s(Ecod + 2Enc) for user U_i to compute the ciphertext data. After training and obtaining the ciphertext weight $E(W)$ in the local update process, every trainer T_i calculates the ciphertext momentum at a cost of $l(\mu$Exp + Gra + 2Enc) in each round, where l is the number of weights and momentums. When the server receives all the weights and momentums from the trainers, it costs Ml(Div + Mul + Enc) to compute the ciphertext federated aggregation average. In the prediction phase, user U_i takes s(Ecod + 2Enc) cost to compute the ciphertext data $E_{pk}(\mathbb{Q}_i)$ for the prediction. After obtaining the ciphertext predicted result, the user costs s(Dcod + Dec) to obtain the plaintext prediction result.

7.2. Functionality

In this subsection, we compare the functional advantages of the proposed method with those of the state-of-the-art privacy-preserving federated deep learning, as shown in Table I. In Table I, PPDL [8] proposed privacy-preserving deep learning models based on Paillier and LWE homomorphic encryptions respectively. However, the scheme does not consider the multi-key and MGD questions. PPFDL [9] and VerifyNet [10] achieved advantages with irregular users or verifiability, respectively. However, they do not consider MGD and SIMD. SecProbe [15] also considered irregular users in the federated training process. However, it does not guarantee the privacy of the aggregated result or support multi-key, MGD, and SIMD. Hybridalpha [16] and HAPPFL [17] protect the privacy of parameters and results, but they are not concerned with MGD and SIMD. MFL [26] improved the MGD but does not consider other functionalities, as shown in Table I. Our PEPFL scheme considers all the functionalities, as demonstrated in Table I with the LDEC.

7.3. Experimental analysis

To illustrate the performance and accuracy, PEPFL is simulated using a computer with an Intel(R) Core(TM) i7-7700HQ CPU @2.80GHz(8CPUs), 8GB RAM, and the 64-bit Windows operating system with Anaconda 3, PyCharm 2020.3.2 Professional Edition, Python 3.8.5, and PyTorch 1.7.0. MNIST dataset, which is composed of a training dataset with 50000 handwritten digit images and validation and test datasets each with 10000 images, is used. The cryptosystem employs the LDEC with PSIMD, which is compared with the other cryptosystems, such as LWE [8], Paillier [44], BCP [45].

*7.3.1. **Accuracy***

In this subsection, we tested the accuracy of PEPFL. Due to the complexity of ciphertext CNN, the CNN training process on the trainers is carried out in plaintext, which does not affect the accuracy. Therefore, the accuracy is tested without considering the privacy-preserving mechanism, and the experiment results are shown in Fig. 4.

Fig. 4 compares the classification accuracy according to the number of rounds and epochs. Fig. 4(a) plots the training accuracy of three kinds of different users {2, 3, 4} as the number of rounds changes. The figure shows that the accuracy is slightly variable when the number of rounds is more than 40. The results show that the accuracy has less influence as the number of rounds and users increases. Fig. 4(b) draws the accuracy under different batches as the number of local epochs increases. When the number of the local epochs is less than 70, it can be seen that the batch is smaller, and the accuracy is higher. When the number of local epochs is greater than 70, the accuracy is nearly 100%. The results demonstrate that the higher epochs, the better accuracy. When the number of epochs reaches a certain amount, the accuracy is always 100%.

Figs. 4(c) and 4(d) simulate against the number of epochs and rounds as the momentums change. Fig. 4(c) compares the accuracy of three different momentums {0.5, 0.7, 0.9} as the number of epochs increases. Obviously, as the number of local epochs increases, the accuracy improves constantly. When the number of local epochs approaches 80, the accuracy reaches 100%. However, when the momentum is greater than

Table 1: Functionality comparison

Scheme	Cryptographic techniques	Servers' number	Parameter privacy	Aggregation result privacy	Multi-key	MGD	PSIMD/SIMD
PPDL [8]	Paillier, LWE	1	✓	✓	×	×	✓
PPFDL [9]	Paillier	2	✓	✓	✓	×	×
VerifyNet [10]	Homomorphic hash, Pseudorandom function, Secret sharing	2	✓	✓	✓	×	×
SecProbe [15]	Differential privacy, Function mechanism	1	✓	×	×	×	×
Hybridalpha [16]	MPC, Differential privacy	2	✓	✓	✓	×	×
HAPPFL [17]	Threshold Paillier, MPC, Differential privacy	1	✓	✓	✓	×	×
MFL [26]	~	1	×	×	×	✓	×
PEPFL	Lifted distributed ElGamal	1	✓	✓	✓	✓	✓

1, the accuracy is below 10%, which is not plotted in Fig. 4(c). As shown in Fig. 4(d), as the number of rounds increases, the accuracy remains almost constant. The results show that as the number of rounds increases, the accuracy remains almost unchanged.

*7.3.2. **Performance***

In this subsection, we analyze the communication costs in different phases, where u is the number of weight encoding packets. In Fig. 5(a), we set p=2048 and u=1000, and the figure shows that the communication costs of the encryption and prediction phases constantly increase as the number of matrices increases, while the other phases remain almost unchanged. The results show that the number of matrices influences the encryption and prediction phases, but does not affect the other phases. In Fig. 5(b), we set s to 1000. As the number of the weight encoding packets u increases, the figure shows that the communication costs of the local update and aggregation phases continually increase, and the other phases are influenced only slightly.

In Fig. 6, because the computational costs of the initialization and aggregation phases approach zero and the local update phase contains the gradients, which causes the computational costs to be unable to be compared. Therefore, we only compare the computational costs in the encoding and encryption phase and the prediction phase. As shown in Fig. 6, as the number of matrices grows, the decryption time in the prediction phase increases distinctly. The results show that the decryption time is impacted clearly and the encryption time is rarely affected.

Then, we experimented with the encryption and decryption running times of the different cryptosystems (LWE, Paillier, BCP, and our cryptosystem) to compare their performance. The test data are square matrices with (5*5, 10*10, 15*15, 20*20, and 25*25) features and (1, 2, 3, 4, and 5) images in the encryption and decryption processes.

As shown in Fig. 7, as the number of matrix rows (columns) versus features or the number of images increases, the running time of encryption constantly increases. Figs. 7(a) and 7(b) show that the encryption time of Paillier is the largest, and that of our method is the smallest. The results indicate that our scheme has shorter encryption running time than the other schemes. Fig. 8 illustrates the decryption running time as the number of matrix rows (columns) versus an increasing number of features or images. From Figs. 8(a) and 8(b), it is apparent that Paillier has the longest decryption time and that our method has the shortest decryption time. The results show that our scheme has a better decryption time than the other schemes.

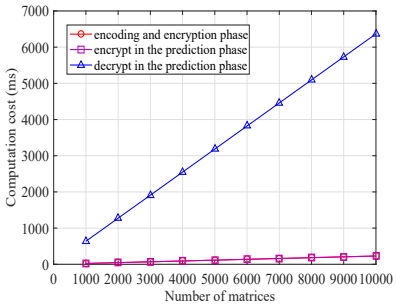
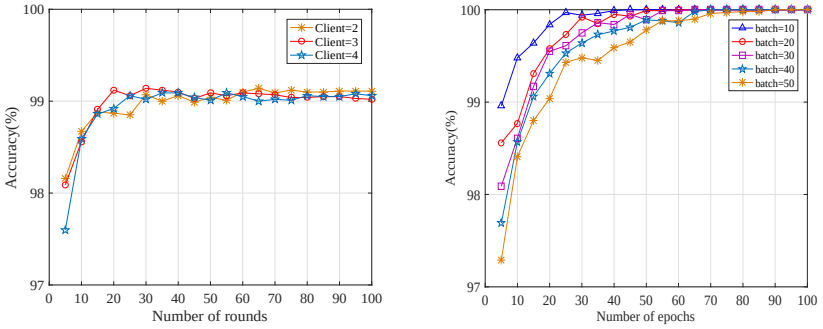


Fig. 6: Computational cost

Finally, we tested the decryption method in the ElGamal decryption with Pollard's Rho and the lookup table method. As shown in Fig. 9, as the number of matrix rows (columns) increases, the running time of decryption constantly increases with Pollard's Rho and the lookup table method, but that of Pollard's Rho method increases distinctly. The results show that the lookup table method is superior to Pollard's Rho method. Therefore, we selected the lookup table method to decrypt our scheme.

8. Conclusion

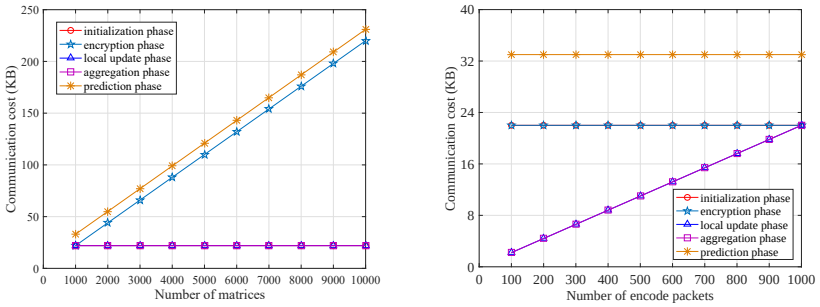
Federated learning is a fashionable collaborative learning method that can collaboratively update the weights or gradients to obtain the global model, and has been researched in different privacy-preserving



(a) Accuracy against rounds (b) Accuracy against epochs

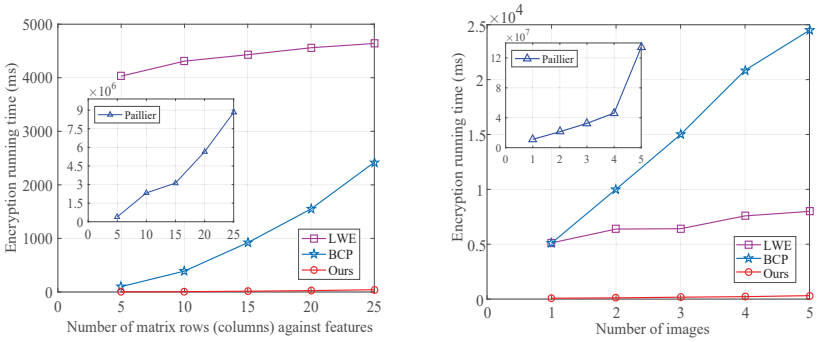
(c) Accuracy against epochs (d) Accuracy against rounds

Fig. 4: Classification Accuracy with rounds and epochs



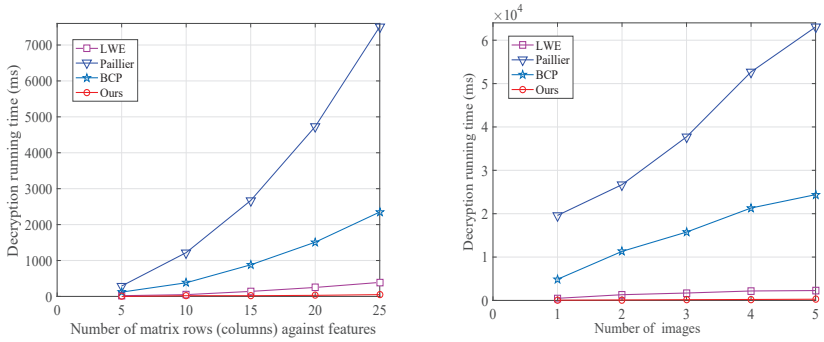
(a) Communication cost in different phases (b) Communication cost in different phases

Fig. 5: Communication cost



(a) Encryption time against features (b) Encryption time against images

Fig. 7: Encryption time with the number of features and images



(a) Decryption time against features (b) Decryption time against images

Fig. 8: Decryption time with the number of features and images

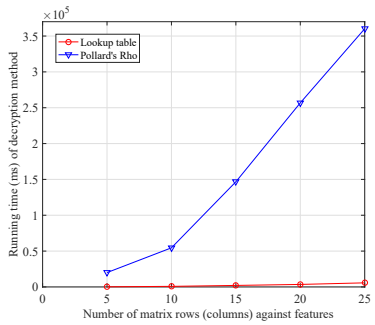


Fig. 9: The comparison of decryption time

ways owing to its privacy information leakage by weights. However, noninteractive and high-efficiency schemes in partially homomorphic cryptosystems have not been proposed. Therefore, we propose a practical and efficient privacy-preserving federated learning framework (PEPFL), which improves the LDEC scheme and presents a PSIMD parallel computing method, solving the distributed multi-key question for federated learning and improving the efficiency. In addition, users do not participate in the training by introducing trainers, which realizes users' noninteraction. The scheme can widely be applied in the scenarios of non-interaction, multi-key, or partially homomorphic cryptosystems. To the best of our knowledge, this is the first work on a partially homomorphic encryption scheme that supports parallel computing with PSIMD, which can also be applied in other homomorphic cryptosystems. In future work, we will focus on preventing participants' from cheating and providing incentive mechanisms in federated learning.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant Nos. U19B2021, 61972457, the Key Research and Development Program of Shaanxi under Grant No. 2020ZDLGY08-04, the Key Technologies R & D Program of He'nan Province under Grant Nos. 212102210084, and the Innovation Scientists and Technicians Troop Construction Projects of Henan Province.

References

[1] S. Sharma, K. Chen, Image disguising for privacy-preserving deep learning, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018, ACM, 2018, pp. 2291–2293.

[2] J. Jeon, J. Kim, J. Kim, K. Kim, A. Mohaisen, J. Kim, Privacy-preserving deep learning computation for geo-distributed medical big-data platforms, CoRR abs/2001.02932.

[3] Y. Wang, H. Tan, Y. Wu, J. Peng, Hybrid electric vehicle energy management with computer vision and deep reinforcement learning, IEEE Trans. Ind. Informatics 17 (6) (2021) 3857–3868.

[4] S. Li, P. Zheng, L. Zheng, An ar-assisted deep learning-based approach for automatic inspection of aviation connectors, IEEE Trans. Ind. Informatics 17 (3) (2021) 1721–1731.

[5] W. Tang, B. Li, M. Barni, J. Li, J. Huang, An automatic cost learning framework for image steganography using deep reinforcement learning, IEEE Trans. Inf. Forensics Secur. 16 (2021) 952–967.

[6] B. Hitaj, G. Ateniese, F. Pérez-Cruz, Deep models under the GAN: information leakage from collaborative deep learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017, ACM, 2017, pp. 603–618.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA, Vol. 54, PMLR, 2017, pp. 1273–1282.

[8] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, S. Moriai, Privacy-preserving deep learning via additively homomorphic encryption, IEEE Trans. Inf. Forensics Secur. 13 (5) (2018) 1333–1345.

[9] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, R. Deng, Privacy-preserving federated deep learning with irregular users, IEEE Trans. Dependable Secur. Comput. (99) (2020) 1–1.

[10] G. Xu, H. Li, S. Liu, K. Yang, X. Lin, Verifynet: Secure and verifiable federated learning, IEEE Trans. Inf. Forensics Secur. 15 (2020) 911–926.

[11] T. Li, J. Li, X. Chen, Z. Liu, W. Lou, Y. T. Hou, Npmml: A framework for non-interactive privacy-preserving multi-party machine learning, IEEE Trans. Dependable Secur. Comput. (99) (2020) 1–14.

[12] P. Xie, B. Wu, G. Sun, BAYHENN: combining bayesian deep learning and homomorphic encryption for secure DNN inference, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, ijcai.org, pp. 4831–4837.

[13] J. Liu, M. Juuti, Y. Lu, N. Asokan, Oblivious neural network predictions via minionn transformations, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017, ACM, 2017, pp. 619–631.

[14] C. Juvekar, V. Vaikuntanathan, A. Chandrakasan, GAZELLE: A low latency framework for secure neural network inference, in: 27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018, USENIX Association, 2018, pp. 1651–1669.

[15] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, Y. Chen, Privacy-preserving collaborative deep learning with unreliable participants, IEEE Trans. Inf. Forensics Secur. 15 (2020) 1486–1500.

[16] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, H. Ludwig, Hybridalpha: An efficient approach for privacy-preserving federated learning, in: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2019, London, UK, November 15, 2019, ACM, 2019, pp. 13–23.

[17] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, Y. Zhou, A hybrid approach to privacy-preserving federated learning, in: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2019, London, UK, November 15, 2019, ACM, 2019, pp. 1–11.

[18] X. Liu, R. H. Deng, K. R. Choo, J. Weng, An efficient privacy-preserving outsourced calculation toolkit with multiple keys, IEEE Trans. Inf. Forensics Secur. 11 (11) (2016) 2401–2414.

[19] C. Li, W. Ma, Comments on "an efficient privacy-preserving outsourced calculation toolkit with multiple keys", IEEE Trans. Inf. Forensics Secur. 13 (10) (2018) 2668–2669.

[20] P. Li, J. Li, Z. Huang, T. Li, C. Gao, S. Yiu, K. Chen, Multi-key privacy-preserving deep learning in cloud computing, Future Gener. Comput. Syst. 74 (2017) 76–85.

[21] X. Ma, J. Ma, L. Hui, J. Qi, G. Sheng, Pdlm: Privacy-preserving deep learning model on cloud with multiple keys, IEEE Trans. Serv. Comput. (2018) 1–1.

[22] H. Chen, W. Dai, M. Kim, Y. Song, Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019, ACM, 2019, pp. 395–412.

[23] S. Zhang, A. Choromanska, Y. LeCun, Deep learning with elastic averaging SGD, in: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, 2015, pp. 685–693.

[24] E. Ghadimi, H. R. Feyzmahdavian, M. Johansson, Global convergence of the heavy-ball method for convex optimization, in: 14th European Control Conference, ECC 2015, Linz, Austria, July 15-17, 2015, IEEE, 2015, pp. 310–315.

[25] J. Wang, V. Tantia, N. Ballas, M. G. Rabbat, Slowmo: Improving communication-efficient distributed SGD with slow momentum, in: 8th International Conference on Learning Representations, ICLR 2020, April 26-30, 2020, Addis Ababa, Ethiopia, OpenReview.net, 2020, pp. 1–25.

[26] W. Liu, L. Chen, Y. Chen, W. Zhang, Accelerating federated learning via momentum gradient descent, IEEE Trans. Parallel Distributed Syst. 31 (8) (2020) 1754–1766.

[27] A. Fu, X. Zhang, N. Xiong, Y. Gao, H. Wang, VFL: A verifiable federated learning with privacy-preserving for big data in industrial iot, CoRR abs/2007.13585.

[28] A. C. Yao, Protocols for secure computations (extended abstract), in: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982, IEEE Computer Society, 1982, pp. 160–164.

[29] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, in: Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings, Vol. 196 of Lecture Notes in Computer Science, Springer, 1984, pp. 10–18.

[30] J. Pollard, Monte carlo method for index computation (mod p), Mathematics of Computation 32 (143) (1978) 918–924.

[31] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Secure distributed key generation for discrete-log based cryptosystems, J. Cryptol. 20 (1) (2007) 51–83.

[32] X. Yi, F. Rao, E. Bertino, A. Bouguettaya, Privacy-preserving association rule mining in cloud computing, in: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14-17, 2015, ACM, 2015, pp. 439–450.

[33] D. Chaum, C. Crépeau, I. Damgård, Multiparty unconditionally secure protocols (abstract), in: Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings, Vol. 293, Springer, 1987, p. 462.

[34] X. Yang, Y. Feng, W. Fang, J. Shao, X. Tang, S.-T. Xia, R. Lu, An accuracy-lossless perturbation method for defending privacy attacks in federated learning (2021). arXiv:2002.09843.

[35] N. Qian, On the momentum term in gradient descent learning algorithms, Neural Networks 12 (1) (1999) 145–151.

[36] T. Ogilvie, R. Player, J. Rowell, Improved privacy-preserving training using fixed-hessian minimisation, IACR Cryptol. ePrint Arch. 2020 (2020) 1514.

[37] D. Chai, L. Wang, K. Chen, Q. Yang, Secure federated matrix factorization, CoRR abs/1906.05108.

[38] D. Chaum, C. Crépeau, I. Damgård, Multiparty unconditionally secure protocols (extended abstract), in: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA, ACM, 1988, pp. 11–19.

[39] E. Hesamifard, H. Takabi, M. Ghasemi, Cryptodl: Deep neural networks over encrypted data, CoRR abs/1711.05189.

[40] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, R. Sharma, Cryptflow2: Practical 2-party secure inference, in: CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020, ACM, 2020, pp. 325–342.

[41] R. Canetti, M. Varia, Decisional diffie-hellman problem, in: Encyclopedia of Cryptography and Security, 2nd Ed, Springer, 2011, pp. 316–319.

[42] Y. Desmedt, Y. Frankel, Threshold cryptosystems, in: Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings, Vol. 435, Springer, 1989, pp. 307–315.

[43] R. Canetti, Universally composable security: A new paradigm for cryptographic protocols, in: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA, IEEE Computer Society, 2001, pp. 136–145.

[44] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding, Vol. 1592, Springer, 1999, pp. 223–238.

[45] E. Bresson, D. Catalano, D. Pointcheval, A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications, in: Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings, Vol. 2894, Springer, 2003, pp. 37–54.