

RESEARCH

Open Access

Privacy-preserving cloud-edge collaborative learning without trusted third-party coordinator



Xiaopeng Yu¹, Dianhua Tang^{1,2} and Wei Zhao^{1*}

Abstract

Cloud-edge collaborative learning has received considerable attention recently, which is an emerging distributed machine learning (ML) architecture for improving the performance of model training among cloud center and edge nodes. However, existing cloud-edge collaborative learning schemes cannot efficiently train high-performance models on large-scale sparse samples, and have the potential risk of revealing the privacy of sensitive data. In this paper, adopting homomorphic encryption (HE) cryptographic technique, we present a privacy-preserving cloud-edge collaborative learning over vertically partitioned data, which allows cloud center and edge node to securely train a shared model without a third-party coordinator, and thus greatly reduces the system complexity. Furthermore, the proposed scheme adopts the batching technique and single instruction multiple data (SIMD) to achieve parallel processing. Finally, the evaluation results show that the proposed scheme improves the model performance and reduces the training time compared with the existing methods; the security analysis indicates that our scheme can guarantee the security in semi-honest model.

Keywords Cloud computing, Edge computing, Collaborative learning, Parallel processing, Security and privacy, Homomorphic encryption

Introduction

With the significant advances of Internet of Things (IoT), large amounts of data are generated through terminal IoT devices at the edge of the network [1]. Edge computing [2], that extends the capabilities of cloud centre to locations close to terminal IoT devices, has been widely recognized as a potential big data analysis framework. To further mine the potential value of IoT big data, there is an increasing need to bring machine learning (ML) to edge computing, namely edge learning [3, 4]. However, if all data analysis and processing tasks are placed on the edge nodes, the quality of data analysis services is difficult to guarantee due to the limited capabilities of edge

nodes [5]. Moreover, in some scenarios, single edge node holds incomplete training data and wants to cooperate with the cloud center to accomplish the model training.

Unfortunately, the data collected by edge nodes from terminal IoT devices is generally valuable and sensitive, and is not able to be directly exchanged in real-world applications due to regulatory regulations and commercial competition [6]. That is, data is commonly stored locally and isolated among different data nodes, which is also called the “data isolation” problem [7]. Blockchain, as a decentralized, distributed, and immutable ledger structure, is considered an adaptable alternative for tackling trust-absence issue in a distributed environment. Consequently, the distributed structure of blockchain is suitable for building cloud-edge collaborative computing [8]. Blockchain technology has been used ensure the security and fairness of data processing. Recently, there is a growing focus on training machine learning models across distributed blockchain nodes without compromising privacy or violating regulations. The concept

*Correspondence:

Wei Zhao
zhaowei9801@163.com

¹ Science and Technology on Communication Security Laboratory, Chengdu 610041, China

² School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China



© The Author(s) 2023. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>. The Creative Commons Public Domain Dedication waiver (<http://creativecommons.org/publicdomain/zero/1.0/>) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

of collaborative learning (CL) has been introduced [9] to meet such requirements, which refers to the process where all parties keep training data locally and can collaboratively train a joint model without exchanging their raw data [10]. Recently, CL for logistic regression (CLLR) [11] has received considerable attention for its efficiency, simplicity and interpretability, which offloads the logistic regression (LR) training tasks of edge nodes to the cloud center. Based on privacy protection methods like homomorphic encryption (HE) [12], secure multi-party computation (MPC) [13], and differential privacy (DP) [14], several privacy-preserving CLLR schemes have been proposed. According to the distribution characteristics of training data, the existing works [15–28] are divided into two categories: horizontal CLLR (HCLLR) [15–19] and vertical CLLR (VCLLR) [20–28]. HCLLR is suitable for datasets held by different entities with the same feature dimensions but different sample spaces [15]. By contrast, VCLLR is applicable to the scenario where different features of the same set of objects are owned by different entities [20]. For the VCLLR, the features owned by a single entity are incomplete, the model training requires all entities to complete together, and thus the training process is more complicated [27].

However, for HE-based methods, the update of model parameters has the potential risk of additional private information leakage even in the semi-honest security setting [29]. For MPC-based methods, after performing secret sharing (SS) [30] on sparse samples, the sparse samples will become dense, and thus could not be handled efficiently. For DP-based methods, due to the model training process adds the noise, the speed of model convergence will be affected, and model accuracy degrades. To address these problems, we present a privacy-preserving cloud-edge CLLR (CECLLR) in this paper.

Our contributions

This work has the following main contributions:

- Firstly, using an approximate HE algorithm [31], we present a secure CECLLR on vertically partitioned data without the coordination of the third-party, which can train a shared model by combining the data from edge node and cloud centre without disclosing raw data and model information, and reduces the complexity of collaborative learning.
- Secondly, using the batching technique [31] and SIMD operation, the proposed CECLLR scheme enables the parallel computing, which greatly improves the computational efficiency and considerably reduces the communication complexity. Besides, the proposed scheme uses least squares to solve the

problem that HE cannot efficiently compute sigmoid function.

- Finally, on three available datasets [32], the performance comparisons demonstrate that, for the proposed CECLLR scheme, compared with related schemes [20, 26], the training time is reduced by nearly 3.6% – 83.0%; the accuracy, F1-score, and AUC improves almost 0.3% – 2.9%, 0.1% – 5.9%, and 0 – 0.02, respectively. Moreover, the security analysis indicates that our scheme can guarantee the security.

Organization

The remainder of this work is arranged as follows. Section [Related works](#) reviews the previous literature. In Section [Preliminaries](#), the preliminaries are discussed. Section [Proposed scheme](#) explains our scheme in detail. Section [Performance evaluation](#) describes the evaluation results. In Section [Security analysis](#), the security of our scheme is proved. Finally, Section [Conclusion](#) summarizes the work.

Related works

There have been several studies [15–28] training LR model while preserving the privacy of sensitive data. In general, existing works implement CLLR by using techniques such as HE [12], MPC [13], and DP [14]. According to the distribution characteristics of training data, existing works [15–28] are able to be divided into two categories: HCLLR [15–19] and VCLLR [20–28]. A review and summary of the existing works [15–28] are introduced below.

For the HCLLR schemes [15–19], using an additive HE algorithm [33] and a data aggregation protocol [34], Mandal et al. [15] proposed a secure horizontal federated LR, where each party is able to train its own model on local dataset and upload its updated model weights to a third-party coordinator that generates a global model weights by aggregating the received model weights. Adopting an additive SS [35, 36], Cock et al. [16] designed a privacy-preserving high-performance CLR training scheme with a TTP initializer. Wang et al. [17] proposed a CECLLR analysis method between cloud centre and edge nodes through multi-key fully HE [37]. By an additive HE [38], Zhu et al. [18] described a value-blind model update method for privacy-preserving LR analysis in a collaborative setting, which protects model privacy by sharing encrypted model parameters among the training parties. Ghavamipour et al. [19] introduced two distributed training methods for secure collaborative LR analysis, but each party needs to transmit multiple shares of its data to other parties separately, which

results in a high communication burden. However, in the existing HCLLR schemes [15–19], data communication among data owners and third-party coordinator increases the training complexity and the risk of privacy leakage. Moreover, it's hard to find a third-party that is trusted by any parties.

For the VCLLR schemes [20–28], based on an additive HE scheme [38], Hardy et al. [20] designed an approximation of Sigmoid to achieve federated LR, but their scheme degrades the model accuracy and requires third-party coordinator. Yang et al. [21] presented a vertical federated LR using quasi-Newton method and additive HE [38]. By an additive SS [39], Zhang et al. [22] introduced a secure CL framework for distributed features. Yang et al. [23] presented a parallel distributed vertical federated LR architecture based on an additive HE scheme [38], which does not require a third-party entity. Li et al. [24] described a vertical CL system for two-party LR based on an approximate HE scheme [31]. Using an additive HE algorithm [38], Wei et al. [25] designed a secure two-parties CLLR on vertically partitioned data. Combining an additive HE algorithm [38] and an secret sharing technique [40], on vertically distributed large-scale sparse training data, Chen et al. [26] presented a secure CLLR scheme by sharing model parameters between two parties. Based on an additive HE [38], He et al. [27] introduced a parallel solution for implementing secure vertical federated LR, which ensures the model accuracy by utilizing a piecewise function, but degrades the efficiency. By adopting an DP algorithm [41] and an HE scheme [38], Sun et al. [28] presented a federated learning algorithm for privacy-preserving vertical CLR, which removes the third-party entity. However, existing VCLLR schemes [20–28] have a low training efficiency and model accuracy.

Preliminaries

System model

As is shown in Fig. 1, the system model consists of two semi-trusted parties: cloud centre P_a and edge node P_b . The datasets D_a and D_b of P_a and P_b are vertically partitioned, namely, P_a has the labels and part of the features, P_b has another part of the features. D_a and D_b are isolated due to regulations and privacy concerns. P_a and P_b collaboratively obtain a joint LR model over their data without revealing the privacy of their sensitive data. The blockchain collects the iterative model parameters as audit records. The description of symbols in this paper are introduced in Table 1.

Homomorphic encryption

HE allows direct calculation on ciphertext without decryption, and can ensure that the calculation on ciphertext is consistent with that on plaintext. Cheon et al. [31] described an approximate HE based on ring learning with errors [42], which includes the operations as follows:

- $\{sk_i, pk_i, gk_i, rk_i\} \leftarrow \text{KeyGen}(N, Q)$: On input the parameters $\{N, Q\}$, it outputs secret key sk_i , public key pk_i , galois key gk_i , and relinearization key rk_i for party P_i .
- $\llbracket m_1 \rrbracket \leftarrow \text{Enc}(m_1, pk_i)$: On input the message vector m_1 and pk_i , it outputs the ciphertext $\llbracket m_1 \rrbracket$.
- $m_1 \leftarrow \text{Dec}(\llbracket m_1 \rrbracket, sk_i)$: On input the ciphertext $\llbracket m_1 \rrbracket$ and sk_i , it outputs the message vector m_1 .
- $\llbracket m_1 + m_2 \rrbracket \leftarrow \text{Add}(\llbracket m_1 \rrbracket, \llbracket m_2 \rrbracket)$: On input two ciphertexts $\llbracket m_1 \rrbracket$ and $\llbracket m_2 \rrbracket$, it outputs the ciphertext $\llbracket m_1 + m_2 \rrbracket = \llbracket m_1 \rrbracket + \llbracket m_2 \rrbracket$.
- $\llbracket m_1 + m_2 \rrbracket \leftarrow \text{Add_Plain}(\llbracket m_1 \rrbracket, m_2)$: On input the ciphertext $\llbracket m_1 \rrbracket$ and message vector m_2 , it outputs the ciphertext $\llbracket m_1 + m_2 \rrbracket = \llbracket m_1 \rrbracket + m_2$.

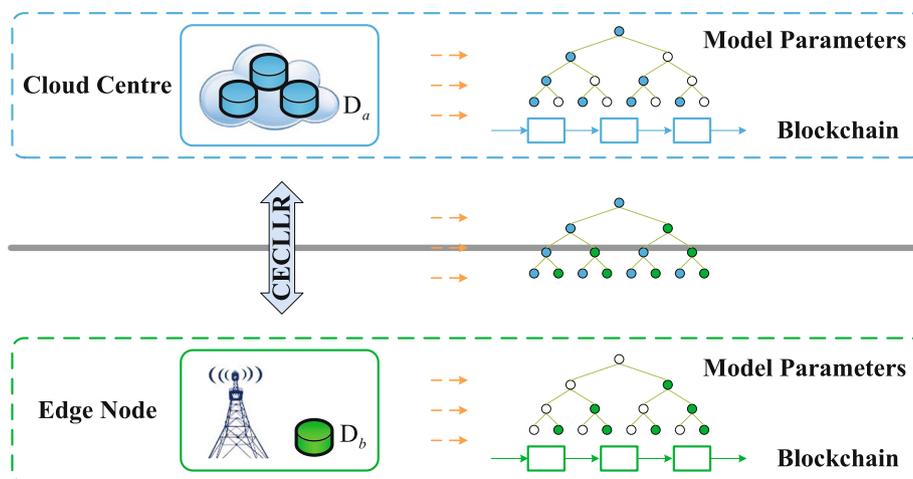


Fig. 1 System model

Table 1 The description of symbol

Symbols	Description
\mathbf{m}	A vector of messages $[m_0, \dots, m_{\frac{N}{2}-1}]$
$\mathbf{m}[j]$	The j -th message of \mathbf{m}
$\llbracket \mathbf{m} \rrbracket$	A ciphertext of \mathbf{m}
\mathbf{M}	A list of vectors $\{\mathbf{m}_0, \dots, \mathbf{m}_{n-1}\}$
$\mathbf{M}[j]$	The j -th vector of \mathbf{M}
$\mathbf{M}[i][j]$	The j -th message of the i -th vector of \mathbf{M}
\mathbf{M}	A set of lists $\{M_0, \dots, M_{n-1}\}$
$\mathbf{M}[j]$	The j -th list of \mathbf{M}
$\mathbf{M}[i][j]$	The j -th vector of the i -th list of \mathbf{M}
$\llbracket \mathbf{M} \rrbracket$	A list of ciphertexts $\{\llbracket \mathbf{m}_0 \rrbracket, \dots, \llbracket \mathbf{m}_{n-1} \rrbracket\}$
$\llbracket \mathbf{M}[j] \rrbracket$	The j -th ciphertext of $\llbracket \mathbf{M} \rrbracket$
$\llbracket \mathbf{M} \rrbracket$	A set of lists $\{\llbracket M_0 \rrbracket, \llbracket M_1 \rrbracket, \dots, \llbracket M_{n-1} \rrbracket\}$
$\llbracket \mathbf{M} \rrbracket [j]$	The j -th list of $\llbracket \mathbf{M} \rrbracket$
$\llbracket \mathbf{M} \rrbracket [i][j]$	The j -th ciphertext of the i -th list of $\llbracket \mathbf{M} \rrbracket$
$\mathbf{m}_0 + \mathbf{m}_1$	Component-wise addition
$\mathbf{m}_0 * \mathbf{m}_1$	Component-wise multiplication
$\mathbf{m}_0 - \mathbf{m}_1$	Component-wise subtraction

- $\llbracket \mathbf{m}_1 + \dots + \mathbf{m}_n \rrbracket \leftarrow \text{Add_Many}(\llbracket \mathbf{M} \rrbracket)$: On input the ciphertext list $\llbracket \mathbf{M} \rrbracket = \{\llbracket \mathbf{m}_1 \rrbracket, \dots, \llbracket \mathbf{m}_n \rrbracket\}$, it outputs the ciphertext $\llbracket \mathbf{m}_1 + \dots + \mathbf{m}_n \rrbracket = \llbracket \mathbf{m}_1 \rrbracket + \dots + \llbracket \mathbf{m}_n \rrbracket$.
- $\llbracket \mathbf{m}_1 - \mathbf{m}_2 \rrbracket \leftarrow \text{Sub}(\llbracket \mathbf{m}_1 \rrbracket, \llbracket \mathbf{m}_2 \rrbracket)$: On input two ciphertexts $\llbracket \mathbf{m}_1 \rrbracket$ and $\llbracket \mathbf{m}_2 \rrbracket$, it outputs the ciphertext $\llbracket \mathbf{m}_1 - \mathbf{m}_2 \rrbracket = \llbracket \mathbf{m}_1 \rrbracket - \llbracket \mathbf{m}_2 \rrbracket$.
- $\llbracket \mathbf{m}_1 - \mathbf{m}_2 \rrbracket \leftarrow \text{Sub_Plain}(\llbracket \mathbf{m}_1 \rrbracket, \mathbf{m}_2)$: On input the ciphertext $\llbracket \mathbf{m}_1 \rrbracket$ and message vector \mathbf{m}_2 , it outputs the ciphertext $\llbracket \mathbf{m}_1 - \mathbf{m}_2 \rrbracket = \llbracket \mathbf{m}_1 \rrbracket - \mathbf{m}_2$.
- $\llbracket \mathbf{m}_1 * \mathbf{m}_2 \rrbracket \leftarrow \text{Mul}(\llbracket \mathbf{m}_1 \rrbracket, \llbracket \mathbf{m}_2 \rrbracket, \text{rk}_i)$: On input two ciphertexts $\llbracket \mathbf{m}_1 \rrbracket, \llbracket \mathbf{m}_2 \rrbracket$ and relinearization key rk_i , it outputs the ciphertext $\llbracket \mathbf{m}_1 * \mathbf{m}_2 \rrbracket = \llbracket \mathbf{m}_1 \rrbracket * \llbracket \mathbf{m}_2 \rrbracket$.
- $\llbracket \mathbf{m}_1 * \mathbf{m}_2 \rrbracket \leftarrow \text{Mul_Plain}(\llbracket \mathbf{m}_1 \rrbracket, \mathbf{m}_2, \text{rk}_i)$: On input the ciphertext $\llbracket \mathbf{m}_1 \rrbracket$, message vector \mathbf{m}_2 , and relinearization key rk_i , it outputs the ciphertext $\llbracket \mathbf{m}_1 * \mathbf{m}_2 \rrbracket = \llbracket \mathbf{m}_1 \rrbracket * \mathbf{m}_2$.
- $\llbracket \mathbf{m}_2 \rrbracket \leftarrow \text{Rotate_Vector}(\llbracket \mathbf{m}_1 \rrbracket, k, \text{gk}_i)$: On input the ciphertext $\llbracket \mathbf{m}_1 \rrbracket = \llbracket [m_{1,0}, m_{1,1}, \dots, m_{1, \frac{N}{2}-1}] \rrbracket$, k , and gk_i , it rotates $\llbracket \mathbf{m}_1 \rrbracket$ left by k , and outputs the ciphertext $\llbracket \mathbf{m}_2 \rrbracket = \llbracket [m_{1,k}, \dots, m_{1, \frac{N}{2}-1}, m_{1,0}, \dots, m_{1, k-1}] \rrbracket$.

Sigmoid approximation

The main idea of binary LR is to output weights $\mathbf{w} = \{w_0, \dots, w_n\}$ that minimizes the loss function

$$J(\mathbf{w}) = -\frac{1}{m} \sum_{i=1}^m ((1 - y_i) \cdot (1 - \log(\sigma(\mathbf{x}_i \cdot \mathbf{w}))) + y_i \cdot \log(\sigma(\mathbf{x}_i \cdot \mathbf{w}))),$$

where $\mathbf{x}_i = \{1, x_{i,1}, \dots, x_{i,n}\}$ and $y_i \in \{0, 1\}$. The following gradient descent (GD)

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \frac{\alpha^{(k)}}{m} \sum_{i=1}^m ((\sigma(\mathbf{x}_i \cdot \mathbf{w}^{(k)}) - y_i) \cdot \mathbf{x}_i),$$

can be employed to obtain the extremum of $J(\mathbf{w})$, where $\alpha^{(k)}$ and $\mathbf{w}^{(k)}$ are the learning rate and the weights at the k -th iteration. Since approximate HE algorithm [31] is unable to perform sigmoid operations $\sigma(x) = \frac{1}{1+e^{-x}}$ effectively, the evaluation of $\sigma(x)$ is a barrier to the implementation of CECLR. By the method of least squares, we obtain a polynomial function $f(x) = w_0 + w_1x - w_3x^3 + w_5x^5 - w_7x^7$ of 7-degree to approximate sigmoid function $\sigma(x)$, where $w_0 = \frac{1}{2}, w_1 = \frac{1.73496}{8}, w_3 = \frac{4.19407}{8^3}, w_5 = \frac{5.43402}{8^5}, w_7 = \frac{2.50739}{8^7}$. The maximum error value between $g(x)$ and $\sigma(x)$ is about 0.032 over the domain $[-8, 8]$.

Proposed scheme

We present a privacy-preserving CECLR on vertically partitioned datasets D_a and D_b , where D_a contains m samples $\{y_i, x_{i,1}, \dots, x_{i,n_1}\}$ with $y_i \in \{0, 1\}$, D_b contains m samples $\{x_{i,n_1+1}, \dots, x_{i,n_1+n_2}\}$, and $i \in [m]$. The first column and other columns of D_a represent the label and features, respectively. Each column of D_b represents a feature. For ease of description, we assume $m = l \cdot \frac{N}{2}$ for $l \in \mathbb{Z}^*$, if the constraint do not meet, the parties pad 0. Moreover, we define the Algorithm 1, 2, which are able to be found in Appendix A. Using the batching techniques from RLWE-based approximate HE [31], our scheme is able to pack and encrypt a message vector containing $\frac{N}{2}$ messages into one ciphertext, and thus reduces the time of model training by parallel processing based on SIMD. We describe our CECLR scheme as follows.

Preprocessing

- P_b computes $l = \lceil \frac{2m}{N} \rceil$, produces $\{\text{sk}_b, \text{pk}_b, \text{rk}_b, \text{gk}_b\} \leftarrow \text{KeyGen}(N, Q)$, encrypts D_b into $(l \times n_2)$ ciphertexts

$$\llbracket \mathbf{x}_{i,n_1+j} \rrbracket = \text{Enc}(\{x_{i, \frac{N}{2}, n_1+j}, \dots, x_{(i+1), \frac{N}{2}-1, n_1+j}\}, \text{pk}_b), \tag{1}$$

where $i = 0, 1, \dots, l - 1, j = 1, 2, \dots, n_2$, encrypts the initial weights $\{w_{n_1+1}^{(0)}, \dots, w_{n_1+n_2}^{(0)}\}$ into n_2 ciphertexts

$$\llbracket w_{n_1+j}^{(0)} \rrbracket = \text{Enc}(\llbracket w_{n_1+j}^{(0)} \rrbracket, \dots, w_{n_1+j}^{(0)} \rrbracket_{1 \times \frac{N}{2}}, \text{pk}_a), \quad (2)$$

where $j = 1, 2, \dots, n_2$, and sends $\{N, Q\}, \text{pk}_b, \text{rk}_b, \text{gk}_b, e, \{\llbracket x_{i,n_1+j} \rrbracket : i = 0, 1, \dots, l - 1, j = 1, 2, \dots, n_2\}, \{\llbracket w_{n_1+j}^{(0)} \rrbracket : j = 1, 2, \dots, n_2\}$ to P_a .

- P_a computes $l = \lceil \frac{2m}{N} \rceil$, sets D_a into $(l \times (n_1 + 1))$ message vectors

$$x_{i,j} = [x_{i, \frac{N}{2}, j}, \dots, x_{(i+1), \frac{N}{2}-1, j}], \quad (3)$$

$$y_i = [y_{i, \frac{N}{2}}, \dots, y_{(i+1), \frac{N}{2}-1}], \quad (4)$$

where $i = 0, 1, \dots, l - 1, j = 1, 2, \dots, n_1$, sets the message vectors

$$x_{i,0} = [1, \dots, 1]_{1 \times \frac{N}{2}}, \quad (5)$$

where $i = 0, 1, \dots, l - 1$, encrypts the initial weights $\{w_0^{(0)}, \dots, w_{n_1}^{(0)}\}$ into $(n_1 + 1)$ ciphertexts

$$\llbracket w_j^{(0)} \rrbracket = \text{Enc}(\llbracket w_j^{(0)} \rrbracket, \dots, w_j^{(0)} \rrbracket_{1 \times \frac{N}{2}}, \text{pk}_b), \quad (6)$$

where $j = 1, 2, \dots, n_1$, sets the message vectors

$$\frac{\alpha}{m} = [\frac{\alpha}{m}, 0, \dots, 0]_{1 \times \frac{N}{2}}, \quad (7)$$

$$a_i = [a_i, \dots, a_i]_{1 \times \frac{N}{2}}, \quad (8)$$

where $i = 0, 1, 3, 5, 7$, and sets the lists

$$X'[i] = \{x_{i,0}, \dots, x_{i,n_1}\}, \quad (9)$$

$$\llbracket X'' \rrbracket [i] = \{\llbracket x_{i,n_1+1} \rrbracket, \dots, \llbracket x_{i,n_1+n_2} \rrbracket\}, \quad (10)$$

where $i = 0, 1, \dots, l - 1$,

$$\llbracket Y \rrbracket = \{\llbracket y_0 \rrbracket, \dots, \llbracket y_{l-1} \rrbracket\}, \quad (11)$$

$$\llbracket W^{(0)} \rrbracket = \{\llbracket w_0^{(0)} \rrbracket, \dots, \llbracket w_{n_1+n_2}^{(0)} \rrbracket\}. \quad (12)$$

Training

P_a and P_b jointly perform the Algorithm 3.

Input: $\llbracket m_0 \rrbracket, a_0, a_1, a_3, a_5, a_7, \text{rk}_b$

Output: $\llbracket f(m_0) \rrbracket = \llbracket m_{14} \rrbracket$

- 1: $\llbracket m_1 \rrbracket = \text{Mul}(\llbracket m_0 \rrbracket, \llbracket m_0 \rrbracket, \text{rk}_b); \llbracket m_2 \rrbracket = \text{Mul}(\llbracket m_1 \rrbracket, \llbracket m_1 \rrbracket, \text{rk}_b)$
- 2: $\llbracket m_3 \rrbracket = \text{Mul}(\llbracket m_2 \rrbracket, \llbracket m_1 \rrbracket, \text{rk}_b); \llbracket m_4 \rrbracket = \text{Mul_Plain}(\llbracket m_0 \rrbracket, a_7, \text{rk}_b)$
- 3: $\llbracket m_5 \rrbracket = \text{Mul}(\llbracket m_3 \rrbracket, \llbracket m_4 \rrbracket, \text{rk}_b); \llbracket m_6 \rrbracket = \text{Mul_Plain}(\llbracket m_0 \rrbracket, a_5, \text{rk}_b)$
- 4: $\llbracket m_7 \rrbracket = \text{Mul}(\llbracket m_2 \rrbracket, \llbracket m_6 \rrbracket, \text{rk}_b); \llbracket m_8 \rrbracket = \text{Mul_Plain}(\llbracket m_0 \rrbracket, a_3, \text{rk}_b)$
- 5: $\llbracket m_9 \rrbracket = \text{Mul}(\llbracket m_1 \rrbracket, \llbracket m_8 \rrbracket, \text{rk}_b); \llbracket m_{10} \rrbracket = \text{Mul_Plain}(\llbracket m_0 \rrbracket, a_1, \text{rk}_b)$
- 6: $\llbracket m_{11} \rrbracket = \text{Add_Plain}(\llbracket m_{10} \rrbracket, a_0); \llbracket m_{12} \rrbracket = \text{Sub}(\llbracket m_{11} \rrbracket, \llbracket m_9 \rrbracket)$
- 7: $\llbracket m_{13} \rrbracket = \text{Add}(\llbracket m_{12} \rrbracket, \llbracket m_7 \rrbracket); \llbracket m_{14} \rrbracket = \text{Sub}(\llbracket m_{13} \rrbracket, \llbracket m_5 \rrbracket)$
- 8: **return** $\llbracket f(m_0) \rrbracket = \llbracket m_{14} \rrbracket$

Algorithm 1 Training

Reconstructing

$$\{w_0^{(e)}, \dots, w_{n_1}^{(e)}\} = \{P^{(e)}[0][0] + O^{(e)}[0][0], \dots, P^{(e)}[n_1][0] + O^{(e)}[n_1][0]\} \quad (14)$$

- P_a sends $\{O^{(e)}[n_1 + 1][0], \dots, O^{(e)}[n_1 + n_2][0]\}$ to P_b , P_b computes

$$\begin{aligned} & \{w_{n_1+1}^{(e)}, \dots, w_{n_1+n_2}^{(e)}\} \\ & = \{P^{(e)}[n_1 + 1][0] + O^{(e)}[n_1 + 1][0], \dots, P^{(e)}[n_1 + n_2][0] + O^{(e)}[n_1 + n_2][0]\} \end{aligned} \quad (13)$$

- P_b sends $\{P^{(e)}[0][0], P^{(e)}[1][0], \dots, P^{(e)}[n_1][0]\}$ to P_a , P_a computes

Performance evaluation

We compare our CECLR and related schemes [20, 26]. All evaluations are performed on a 64-bits desktop with Linux system, 16 GB memory and i7 CPU. We choose initial weights $\{w_0^{(0)}, \dots, w_{n_1}^{(0)}\} = \{0, \dots, 0\}$, $\{w_{n_1+1}^{(0)}, \dots, w_{n_1+n_2}^{(0)}\} = \{0, \dots, 0\}$, learning rate $\varphi = 0.15$, and maximum number of iterations $e = 20$. For the schemes [20, 26], Paillier cryptosystem [38] is chosen

Table 2 Performance comparisons

D_a	D_b	Scheme	Training Time	Accuracy	F1-score	AUC	No TTP
$\Phi_1\{l, f_1 - f_4\}$	$\Phi_1\{f_5 - f_8\}$	[20]	1.28 min	74.1%	85.1%	0.57	×
		[26]	2.31 min	74.1%	85.1%	0.56	✓
		Ours	1.23 min	74.4%	85.2%	0.58	✓
$\Phi_2\{l, f_1 - f_5\}$	$\Phi_2\{f_6 - f_9\}$	[20]	2.23 min	91.3%	77.5%	0.96	×
		[26]	3.33 min	90.9%	75.3%	0.96	✓
		Ours	2.15 min	91.7%	77.9%	0.96	✓
$\Phi_3\{l, f_1 - f_8\}$	$\Phi_3\{f_9 - f_{15}\}$	[20]	9.09 min	84.6%	55.6%	0.89	×
		[26]	14.40 min	82.7%	60.1%	0.89	✓
		Ours	2.45 min	85.6%	61.5%	0.90	✓

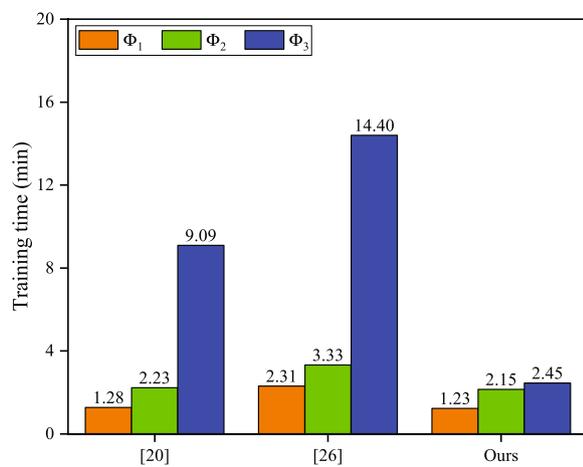


Fig. 2 Training time

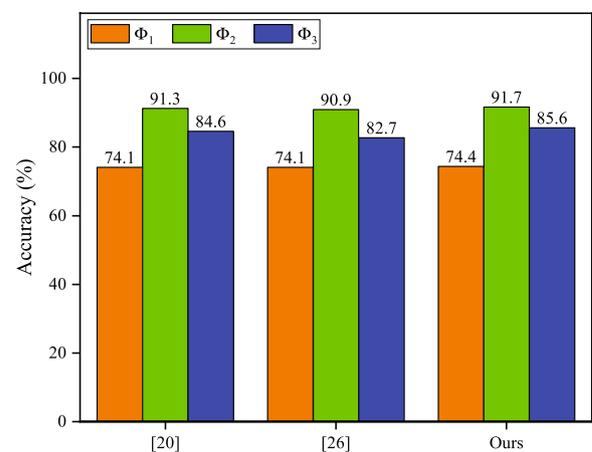


Fig. 3 Accuracy

to support HE computations. For our scheme, we utilize the SEAL library [43] to instantiate HE computations [31]. To ensure the security with $\kappa = 80$ bits, we select prime number $p, q = 512$ bits and $n = 1024$ bits in schemes [20, 26]. In our scheme, the rescaling operation in Mul and Mul_Plain needs the modulus reduction of $6q_0$ and $4q_1$ bits, the ciphertext modulus of the output in each iteration q_2 needs to be larger than q_0 to achieve the correctness and accuracy. Let $q_0 = 50, q_1 = 40$ and $q_2 = 60$, we obtain $Q = 6q_0 + 4q_1 + q_2 = 520$ bits, $N = 2^{15} \geq \frac{\kappa + 110}{7.2} \cdot \log Q$. On three datasets [32]: Φ_1 - uis, Φ_2 - Edinburgh, and Φ_3 - nhanes3, we compare our CECLR with related schemes [20, 26] in terms of training time, accuracy, F1-score, AUC. Φ_1 has 8 features and 575 samples; Φ_2 has 9 features and 1253 samples; Φ_3 has 15 features and 15649 samples. P_a has the labels and the first 4 features $\{l, f_1 - f_4\}$ of Φ_1 , P_b has the last 4 features $\{f_5 - f_8\}$ of Φ_1 ; P_a has the labels and the first 5 features $\{l, f_1 - f_5\}$ of Φ_2 , P_b has the last 4 features $\{f_6 - f_9\}$ of Φ_2 ; P_a has the labels and the first 8 features $\{l, f_1 - f_8\}$ of Φ_3 , P_b has the last 7 features $\{f_9 - f_{15}\}$ of Φ_3 . All presented

results are the average of 10 tests, the validity of results is obtained by 5-fold cross-validation. The performance comparisons among our CECLR scheme and the schemes [20, 26] are shown in Table 2, where “✓” means “Satisfied”, “×” denotes “Not satisfied”. From Table 2, we obtain that our CECLR is able to train a model in less time without degrading the model performance than the schemes [20, 26], and does not need a third-party coordinator.

From Fig. 2, we see that, compared with [20, 26], the training time in our method (1.23 min) reduces nearly 3.9% and 46.8% over Φ_1 ; the training time in our method (2.15 min) reduces almost 3.6% and 35.4% over Φ_2 ; the training time in our method (2.45 min) reduces about 73.0% and 83.0% over Φ_3 .

From Fig. 3, we see that, compared with [20, 26], the accuracy in our method (74.4%) is nearly 0.3% and 0.3% increase over Φ_1 ; the accuracy in our method (91.7%) is almost 0.4% and 0.8% increase over Φ_2 ; the accuracy in our method (85.6%) is about 1.0% and 2.9% increase over Φ_3 .

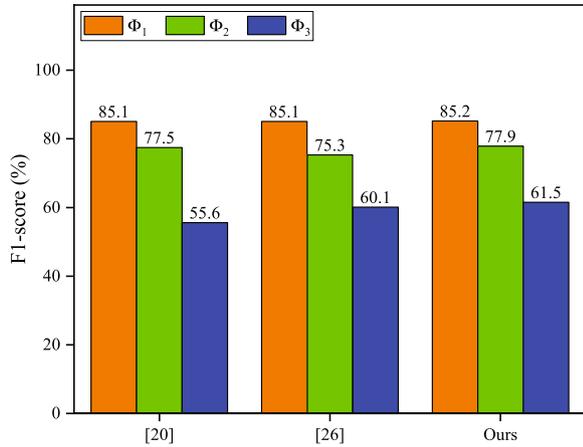


Fig. 4 F1-score

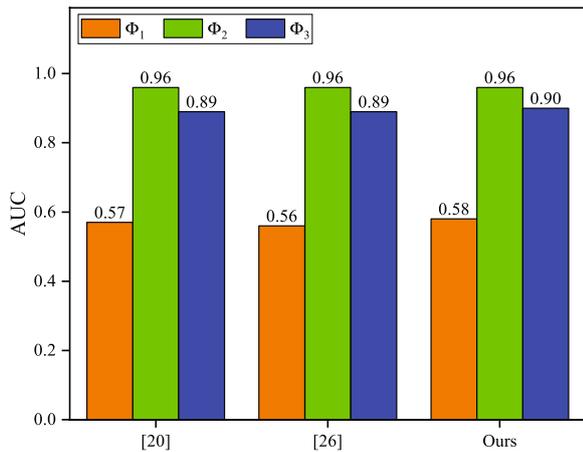


Fig. 5 AUC

From Fig. 4, we see that, compared with [20, 26], the F1-score in our method (85.2%) has nearly 0.1% and 0.1% increase over Φ_1 ; the F1-score in our method (77.9%) has almost 0.4% and 2.6% increase over Φ_2 ; the F1-score in our method (61.5%) has about 5.9% and 1.4% increase over Φ_3 .

From Fig. 5, we see that, compared with [20, 26], the AUC in our method (0.58) is nearly 0.01 and 0.02 increase over Φ_1 ; the AUC in our method (0.96) is the same over Φ_2 ; the AUC in our method (0.90) is about 0.01 and 0.01 increase over Φ_3 .

Security Analysis

Let P_a and P_b learn pk_b, rk_b, gk_b , and only P_b gets sk_b , our CECLR is to compute an objective functionality $f = \{f_a, f_b\}$ in semi-honest model [44]. For inputs

$\{m_a, m_b\}$ of $\{P_a, P_b\}$, the outputs $\{f_a(m_a, m_b), f_b(m_a, m_b)\}$ for $\{P_a, P_b\}$ are random, and neither party can infer private data beyond its output. Next, we demonstrate the security of our CECLR by means of simulation-based security [45].

Definition 1 Let Π represent a protocol to calculate deterministic functionality $f = \{f_a, f_b\}$. Given inputs $\{m_a, m_b\}$ of $\{P_a, P_b\}$, and security parameter κ , the views for P_a and P_b in Π are represented as $\mathcal{V}_a(\kappa, m_a, m_b) = \{pk_b, rk_b, gk_b, m_a, o_a\}$ and $\mathcal{V}_b(\kappa, m_a, m_b) = \{sk_b, pk_b, rk_b, gk_b, m_b, o_b\}$, where m_a denotes P_a 's input, m_b denotes P_b 's input, o_a represents the output for P_a , and o_b represents the output for P_b . Π has the ability to calculate f securely in the semi-honest model if there are two simulators \mathcal{S}_a and \mathcal{S}_b with probabilistic polynomial-time, such that

$$\{\mathcal{S}_a(1^\kappa, m_a, f_a(m_a, m_b))\}_{\kappa, m_a, m_b} \cong \{\mathcal{V}_a(\kappa, m_a, m_b)\}_{\kappa, m_a, m_b},$$

$$\{\mathcal{S}_b(1^\kappa, m_b, f_b(m_a, m_b))\}_{\kappa, m_a, m_b} \cong \{\mathcal{V}_b(\kappa, m_a, m_b)\}_{\kappa, m_a, m_b}.$$

Theorem 1 Suppose that P_a and P_b do not collude, and approximate HE algorithm [31] meets the semantic security, the proposed CECLR could achieve the security under semi-honest model.

Proof

Following simulation-based security [45], we demonstrate that we could construct $\mathcal{S}_{\mathcal{A}_a}$ and $\mathcal{S}_{\mathcal{A}_b}$, such that

$$\{\mathcal{S}_{\mathcal{A}_a}(1^\kappa, \llbracket c \rrbracket, \mathbf{o}_a)\}_{\kappa, \llbracket c \rrbracket, sk_b} \cong \{\mathcal{V}_{\mathcal{A}_a}(\kappa, \llbracket c \rrbracket, sk_b)\}_{\kappa, \llbracket c \rrbracket, sk_b},$$

$$\{\mathcal{S}_{\mathcal{A}_b}(1^\kappa, \mathbf{o}_b, sk_b)\}_{\kappa, \llbracket c \rrbracket, sk_b} \cong \{\mathcal{V}_{\mathcal{A}_b}(\kappa, \llbracket c \rrbracket, sk_b)\}_{\kappa, \llbracket c \rrbracket, sk_b},$$

where $\mathcal{V}_{\mathcal{A}_a}$ and $\mathcal{V}_{\mathcal{A}_b}$ represent the views of corrupted parties \mathcal{A}_a and \mathcal{A}_b , respectively. We prove that \mathcal{A}_a and \mathcal{A}_b cannot distinguish the above two equations.

Against corrupted \mathcal{A}_a : Given κ , input $\llbracket c \rrbracket$ and output \mathbf{o}_a of \mathcal{A}_a , we build a $\mathcal{S}_{\mathcal{A}_a}$ to simulate the view of \mathcal{A}_a . To do this, we analyze $\mathcal{V}_{\mathcal{A}_a}(\kappa, \llbracket c \rrbracket, sk_b)$. Since \mathcal{A}_a does not get other message from \mathcal{A}_b , $\mathcal{V}_{\mathcal{A}_a}(\kappa, \llbracket c \rrbracket, sk_b)$ consists of $\llbracket c \rrbracket$ and random message r_a . Given $\kappa, \llbracket c \rrbracket$, and \mathbf{o}_a , $\mathcal{S}_{\mathcal{A}_a}$ produces the simulation $(\llbracket c \rrbracket, r_a)$ of $\mathcal{V}_{\mathcal{A}_a}(\kappa, \llbracket c \rrbracket, sk_b)$. Therefore, we obtain:

$$\mathcal{V}_{\mathcal{A}_a}(\kappa, \llbracket c \rrbracket, sk_b) = (\llbracket c \rrbracket, r_a), \mathcal{S}_{\mathcal{A}_a}(1^\kappa, \llbracket c \rrbracket, \mathbf{o}_a) = (\llbracket c \rrbracket, r_a).$$

According to the above analysis, we could obtain that $\mathcal{S}_{\mathcal{A}_a}$'s output and \mathcal{A}_a 's view is indistinguishable. Therefore, our CECLR is secure against semi-honest \mathcal{A}_a .

Against corrupted \mathcal{A}_b : Given κ , input sk_b and output \mathbf{o}_b of \mathcal{A}_b , we build a $\mathcal{S}_{\mathcal{A}_b}$ to simulate the view of \mathcal{A}_b . With regards to this, we analyze $\mathcal{V}_{\mathcal{A}_b}(\kappa, \llbracket \mathbf{c} \rrbracket, sk_b)$. Since the only data received by \mathcal{A}_b is $\llbracket \mathbf{c} \rrbracket$, $\mathcal{V}_{\mathcal{A}_b}(\kappa, \llbracket \mathbf{c} \rrbracket, sk_b)$ consists of sk_b of \mathcal{A}_b , random messenger r_b and ciphertext $\llbracket \mathbf{o}_b \rrbracket$. Given κ , sk_b , and \mathbf{o}_b , $\mathcal{S}_{\mathcal{A}_b}$ produces the simulation of $\mathcal{V}_{\mathcal{A}_b}(\kappa, \llbracket \mathbf{c} \rrbracket, sk_b)$. $\mathcal{S}_{\mathcal{A}_b}$ encrypts \mathbf{o}_b with pk_b into $\llbracket \mathbf{o}'_b \rrbracket$, and produces the output $(sk_b, r_b, \llbracket \mathbf{o}'_b \rrbracket)$. Therefore, we obtain:

$$\mathcal{V}_{\mathcal{A}_b}(\kappa, \llbracket \mathbf{c} \rrbracket, sk_b) = (sk_b, r_b, \llbracket \mathbf{o}_b \rrbracket), \mathcal{S}_{\mathcal{A}_b}(1^\kappa, sk_b, \mathbf{o}_b) = (sk_b, r_b, \llbracket \mathbf{o}'_b \rrbracket).$$

From the above analysis, we could obtain that $\mathcal{S}_{\mathcal{A}_b}$'s output and \mathcal{A}_b 's view is indistinguishable. Therefore, our CECLR is secure against semi-honest \mathcal{A}_b .

Conclusion

In this paper, using an approximate HE algorithm, we present a CECLR without third-party coordinator while achieving the security of the raw data and model information for all parties. The proposed scheme adopts the batching and SIMD achieves a parallel processing. The performance evaluations based on the experiments show that, compared with existing methods, our scheme has lower training time and higher model performance. In future work, we will improve our scheme to support complex model training and datasets, and optimize our scheme to resist attacks by malicious parties.

Appendix A

Input: $\llbracket \mathbf{m}_0 \rrbracket = \llbracket [m_{0,0}, \dots, m_{0, \frac{N}{2}-1}] \rrbracket, \frac{N}{2}, \text{gk}_b$
Output: $\llbracket \mathbf{m}_1 \rrbracket = \llbracket [\sum_{i=0}^{\frac{N}{2}-1} m_{0,i}, \dots, \sum_{i=0}^{\frac{N}{2}-1} m_{0,i}]_{1 \times \frac{N}{2}} \rrbracket$

- 1: $\llbracket \mathbf{m}_1 \rrbracket = \llbracket \mathbf{m}_0 \rrbracket$
- 2: **for** ($k = \frac{N}{2}; k > 0; k = \frac{k}{2}$) **do**
- 3: $\llbracket \mathbf{m}_2 \rrbracket = \text{Rotate_Vector}(\llbracket \mathbf{m}_1 \rrbracket, k, \text{gk}_b)$
- 4: $\llbracket \mathbf{m}_1 \rrbracket = \text{Add}(\llbracket \mathbf{m}_1 \rrbracket, \llbracket \mathbf{m}_2 \rrbracket)$
- 5: **end for**
- 6: **return** $\llbracket \mathbf{m}_1 \rrbracket$

Algorithm 2 $\llbracket f(\mathbf{m}_0) \rrbracket = \text{Approx_Sigmoid}(\llbracket \mathbf{m}_0 \rrbracket, \mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_3, \mathbf{a}_5, \mathbf{a}_7, \text{rk}_b)$

Input: \mathbf{X}' , $\llbracket \mathbf{X}'' \rrbracket$, $\llbracket \mathbf{Y} \rrbracket$ and $\llbracket \mathbf{W}^{(0)} \rrbracket$
Output: $\mathbf{O}^{(e)}$ for P_a and $\mathbf{P}^{(e)}$ for P_b

- 1: **for** ($k = 0$ to $(e - 1)$) **do**
- 2: **for** ($i = 0$ to $(l - 1)$) **do**
- 3: **for** ($j = 0$ to n_1) **do**
- 4: P_a computes $\llbracket \mathbf{C} \rrbracket[i][j] = \text{Mul_Plain}(\llbracket \mathbf{W}^{(k)} \rrbracket[j], \mathbf{X}'[i][j], \text{rk}_b)$
- 5: **end for**
- 6: **for** ($j = 1$ to n_2) **do**
- 7: P_a computes $\llbracket \mathbf{C} \rrbracket[i][n_1 + j] = \text{Mul}(\llbracket \mathbf{W}^{(k)} \rrbracket[n_1 + j], \llbracket \mathbf{X}'' \rrbracket[i][j], \text{rk}_b)$
- 8: **end for**
- 9: P_a computes $\llbracket \mathbf{E} \rrbracket[i] = \text{Add_Many}(\llbracket \mathbf{C} \rrbracket[i])$
- 10: P_a computes $\llbracket \mathbf{F} \rrbracket[i] = \text{Approx_Sigmoid}(\llbracket \mathbf{E} \rrbracket[i], \mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_3, \mathbf{a}_5, \mathbf{a}_7, \text{rk}_b)$
- 11: P_a computes $\llbracket \mathbf{G} \rrbracket[i] = \text{Sub}(\llbracket \mathbf{F} \rrbracket[i], \mathbf{Y}[i])$
- 12: **for** ($j = 0$ to n_1) **do**
- 13: P_a computes $\llbracket \mathbf{H} \rrbracket[i][j] = \text{Mul_Plain}(\llbracket \mathbf{G} \rrbracket[i], \mathbf{X}'[i][j], \text{rk}_b)$
- 14: **end for**
- 15: **for** ($j = 1$ to n_2) **do**
- 16: P_a computes $\llbracket \mathbf{H} \rrbracket[i][n_1 + j] = \text{Mul}(\llbracket \mathbf{G} \rrbracket[i], \llbracket \mathbf{X}'' \rrbracket[i][j], \text{rk}_b)$
- 17: **end for**
- 18: **end for**
- 19: **for** ($j = 0$ to $(n_1 + n_2)$) **do**
- 20: P_a computes $\llbracket \mathbf{I} \rrbracket[j] = \text{Add_Many}(\{\llbracket \mathbf{H} \rrbracket[0][j], \dots, \llbracket \mathbf{H} \rrbracket[l - 1][j]\})$
- 21: P_a computes $\llbracket \mathbf{J} \rrbracket[j] = \text{Rot_Sum}(\llbracket \mathbf{I} \rrbracket[j], \frac{N}{2}, \text{gk}_b)$
- 22: P_a computes $\llbracket \mathbf{K} \rrbracket[j] = \text{Mul_Plain}(\llbracket \mathbf{J} \rrbracket[j], \frac{\alpha}{m}, \text{rk}_b)$
- 23: P_a computes $\llbracket \mathbf{W}^{(k+1)} \rrbracket[j] = \text{Sub}(\llbracket \mathbf{W}^{(k)} \rrbracket[j], \llbracket \mathbf{K} \rrbracket[j])$
- 24: P_a randomly chooses $\mathbf{O}^{(k+1)}[j] = [o_{j,0}^{(k+1)}, \dots, o_{j,\frac{N}{2}-1}^{(k+1)}]$
- 25: P_a computes $\llbracket \mathbf{P}^{(k+1)} \rrbracket[j] = \text{Sub_Plain}(\llbracket \mathbf{W}^{(k+1)} \rrbracket[j], \mathbf{O}^{(k+1)}[j])$
- 26: **end for**
- 27: P_a sends $\llbracket \mathbf{P}^{(k+1)} \rrbracket$ to P_b
- 28: **for** ($j = 0$ to $(n_1 + n_2)$) **do**
- 29: P_b computes $\mathbf{P}^{(k+1)}[j] = \text{Dec}(\llbracket \mathbf{P}^{(k+1)} \rrbracket[j], \text{sk}_b)$
- 30: P_b computes $\llbracket \mathbf{Q}^{(k+1)} \rrbracket[j] = \text{Enc}(\mathbf{P}^{(k+1)}[j][0], \dots, \mathbf{P}^{(k+1)}[j][0], \text{pk}_b)$
- 31: **end for**
- 32: P_b sends $\llbracket \mathbf{Q}^{(k+1)} \rrbracket$ to P_a
- 33: **for** ($j = 0$ to $(n_1 + n_2)$) **do**
- 34: P_a sets $\mathbf{R}^{(k+1)}[j] = [\mathbf{O}^{(k+1)}[j][0], \dots, \mathbf{O}^{(k+1)}[j][0]]$
- 35: P_a computes $\llbracket \mathbf{W}^{(k+1)} \rrbracket[j] = \text{Add_Plain}(\llbracket \mathbf{Q}^{(k+1)} \rrbracket[j], \mathbf{R}^{(k+1)}[j])$
- 36: **end for**
- 37: **end for**
- 38: **return** $\mathbf{O}^{(e)}$ for P_a and $\mathbf{P}^{(e)}$ for P_b

Algorithm 3 $\llbracket \mathbf{m}_1 \rrbracket = \text{Rot_Sum}(\llbracket \mathbf{m}_0 \rrbracket, \frac{N}{2}, \text{gk}_b)$

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No. U19B2021) and Stability Program of Science and Technology on Communication Security Laboratory (Grant No. 2022).

Authors' contributions

Xiaopeng Yu proposed the idea, conducted the experiments, and wrote the paper; Dianhua Tang reviewed the experiments and revised the paper; Wei Zhao analyzed the data and revised the paper. All authors reviewed the manuscript. The author(s) read and approved the final manuscript.

Funding

This work is supported by the National Natural Science Foundation of China (Grant No. U19B2021) and Stability Program of Science and Technology on Communication Security Laboratory (Grant No. 2022).

Availability of data and materials

The supporting data can be provided on request.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare no conflicts of interest.

Received: 25 October 2022 Accepted: 17 January 2023

Published online: 10 February 2023

References

- Feng J, Liu L, Pei Q, Li K (2022) Min-max cost optimization for efficient hierarchical federated learning in wireless edge networks. *IEEE Trans Parallel Distrib Syst* 33(11):2687–2700
- Shi W, Jie C, Quan Z, Li Y, Xu L (2016) Edge computing: Vision and challenges. *IEEE Internet Things J* 3(5):637–646
- Feng J, Zhang W, Pei Q, Wu J, Lin X (2022) Heterogeneous computation and resource allocation for wireless powered federated edge learning systems. *IEEE Trans Commun* 70(5):3220–3233
- Liu L, Zhang J, Song SH, Letaief KB (2020) Client-edge-cloud hierarchical federated learning. In: *IEEE International Conference on Communications*. Piscataway, IEEE, pp 1–6
- Feng J, Liu L, Pei Q, Hou F, Wu J (2021) Service characteristics-oriented joint optimization of radio and computing resource allocation in mobile-edge computing. *IEEE Internet Things J* 8(11):9407–9421
- Mohassel P, Zhang Y (2017) Secureml: A system for scalable privacy-preserving machine learning. In: *IEEE Symposium on Security and Privacy*. Piscataway, IEEE, pp 19–38
- Yang Q, Liu Y, Chen T, Tong Y (2019) Federated machine learning: concept and applications. *ACM Trans Intell Syst Technol* 10(2):1–19
- Du J, Cheng W, Lu G, Cao H, Chu X, Zhang Z, Wang J (2022) Resource pricing and allocation in mec enabled blockchain systems: An a3c deep reinforcement learning approach. *IEEE Trans Netw Sci Eng* 9(1):33–44
- Konenčný J, McMahan HB, Yu FX, Richtárik P, Suresh AT, Bacon D (2016) Federated learning: Strategies for improving communication efficiency. Preprint at <https://arxiv.org/abs/1610.05492>
- Du J, Yu FR, Lu G, Wang J, Jiang J, Chu X (2020) Mec-assisted immersive vr video streaming over terahertz wireless networks: A deep reinforcement learning approach. *IEEE Internet Things J* 7(10):9517–9529
- Ding C, Zhou A, Liu Y, Chang RN, Hsu CH, Wang S (2022) A cloud-edge collaboration framework for cognitive service. *IEEE Trans Cloud Comput* 10(3):1489–1499
- Rivest RL, Adleman LM, Dertouzos ML (1978) On data banks and privacy homomorphisms. *Found Secure Comput* 76(4):169–179
- Yao AC (1982) Protocols for secure computations. In: *23rd Annual IEEE Symposium on Foundations of Computer Science*. Piscataway, IEEE, pp 1–5
- Dwork C, Kenthapadi K, Mcsherry F, Mironov I, Naor M (2006) Our data, ourselves: privacy via distributed noise generation. In: *Advances in Cryptology - EUROCRYPT 2006: International Conference on the Theory and Application of Cryptographic Techniques*. Springer, Heidelberg, pp 486–503
- Mandal K, Gong G (2019) Privfl: Practical privacy-preserving federated regressions on high-dimensional data over mobile networks. In: *10th ACM SIGSAC Conference on Cloud Computing Security Workshop*. ACM, New York, pp 57–68
- Cock MD, Dowsley R, Nascimento ACA, Railsback D, Shen JW, Todoki A (2021) High performance logistic regression for privacy-preserving genome analysis. *BMC Med Genomics* 14(1):1–18
- Wang C, Xu J, Yin L (2021) A secure cloud-edge collaborative logistic regression model. In: *IEEE Congress on Cybermatics/14th IEEE International Conference on Internet of Things/14th IEEE International Conference on Cyber, Physical and Social Computing/17th IEEE International Conference on Green Computing and Communications/7th IEEE International Conference on Smart Data*. Piscataway, IEEE, pp 244–253
- Zhu R, Jiang C, Wang X, Wang S, Zheng H, Tang H (2020) Privacy-preserving construction of generalized linear mixed model for biomedical computation. *Bioinformatics* 36(Supplement_1):128–135
- Ghavamiipour AR, Turkmen F, Jian X (2022) Privacy-preserving logistic regression with secret sharing. *BMC Med Inform Decis Mak* 22(89):1–11
- Hardy S, Henecka W, Ivey-Law H, Nock R, Patrini G, Smith G, Thorne B (2017) Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. Preprint at <https://arxiv.org/abs/1711.10677>
- Yang K, Fan T, Chen T, Shi Y, Yang Q (2019) A quasi-newton method based vertical federated learning framework for logistic regression. Preprint at <https://arxiv.org/abs/1912.00513>
- Zhang Y, Bai G, Li X, Curtis C, Ko RKL (2020) Privcoll: Practical privacy-preserving collaborative machine learning. In: *25th European Symposium on Research in Computer Security*. Springer, Heidelberg, pp 399–418
- Yang S, Ren B, Zhou X, Liu L (2019) Parallel distributed logistic regression for vertical federated learning without third-party coordinator. Preprint at <https://arxiv.org/abs/1911.09824>
- Li Q, Huang Z, Lu WJ, Hong C, Qu H, He H, Zhang W (2020) Homopai: A secure collaborative machine learning platform based on homomorphic encryption. In: *IEEE 36th International Conference on Data Engineering*. Piscataway, IEEE, pp 1713–1717
- Wei QJ, Li Q, Zhou ZP, Ge ZQ, Zhang YG (2020) Privacy-preserving two-parties logistic regression on vertically partitioned data using asynchronous gradient sharing. *Peer Peer Netw Appl* 14(3):1379–1387
- Chen C, Zhou J, Wang L, Wu X, Fang W, Tan J, Wang L, Ji X, Liu A, Wang H (2021) When homomorphic encryption marries secret sharing: secure large-scale sparse logistic regression and applications in risk control. In: *27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, pp 2652–2662
- He D, Du R, Zhu S, Zhang M, Liang K, Chan S (2022) Secure logistic regression for vertical federated learning. *IEEE Internet Comput* 26(2):61–68
- Sun H, Wang Z, Huang Y, Ye J (2022) Privacy-preserving vertical federated logistic regression without trusted third-party coordinator. In: *6th International Conference on Machine Learning and Soft Computing*. ACM, New York, pp 132–138
- Li Z, Huang Z, Chen C, Hong C (2019) Quantification of the leakage in federated learning. Preprint at <https://arxiv.org/abs/1910.05467>
- Shamir A (1979) How to share a secret. *Commun ACM* 22(11):612–613
- Cheon JH, Kim A, Kim M, Song Y (2017) Homomorphic encryption for arithmetic of approximate numbers. In: *Advances in Cryptology - ASIA-CRYPT 2017: 23rd International Conference on the Theory and Application of Cryptology and Information Security*. Springer, Heidelberg, pp 409–437
- Kim A, Song Y, Kim M, Lee K, Cheon JH (2018) Logistic regression model training based on the approximate homomorphic encryption. *BMC Med Genomics* 83(Supplement 4):23–31

33. Joye M, Libert B (2013) Efficient cryptosystems from \mathcal{R} -th power residue symbols. In: *Advances in Cryptology - EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Heidelberg, pp 76–92
34. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, Seth K (2017) Practical secure aggregation for privacy-preserving machine learning. In: *2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, New York, pp 1175–1191
35. Cock MD, Dowsley R, Horst C, Katti R, Nascimento A, Poon WS, Truex S (2019) Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation. *IEEE Trans Dependable Secure Comput* 16(2):217–230
36. Reich D, Todoki A, Dowsley R, Cock MD, Nascimento A (2008) Privacy-preserving classification of personal text messages with secure multi-party computation: an application to hate-speech detection. In: *33rd International Conference on Neural Information Processing Systems*. Springer, Heidelberg, pp 3757–3769
37. Chen H, Dai W, Kim M, Song Y (2019) Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In: *ACM SIGSAC Conference on Computer and Communications Security*. ACM, New York, pp 395–412
38. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: *Advances in Cryptology - EUROCRYPT 1999: International Conference on the Theory and Application of Cryptographic techniques*. Springer, Heidelberg, pp 223–238
39. Dan B, Laur S, Willemson J (2008) Sharemind: A framework for fast privacy-preserving computations. In: *13th European Symposium on Research in Computer Security*. Springer, Heidelberg, pp 192–206
40. Boyle E, Gilboa N, Ishai Y (2015) Function secret sharing. In: *Advances in Cryptology - EUROCRYPT: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Heidelberg, pp 337–367
41. Dwork C, McSherry F, Nissim K, Smith A (2006) Calibrating noise to sensitivity in private data analysis. In: *Theory of Cryptography*. Springer, Heidelberg, pp 265–284
42. Lyubashevsky V, Peikert C, Regev O (2010) On ideal lattices and learning with errors over rings. In: *Advances in Cryptology - EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Heidelberg, pp 1–23
43. SEAL Microsoft SEAL (release 4.0). Microsoft Research, Redmond. <https://github.com/Microsoft/SEAL>. Accessed Mar 2022.
44. Goldreich O (2006) *Foundations of Cryptography: Volume I, Basic Applications*. Cambridge University Press, Cambridge
45. Küsters R, Datta A, Mitchell JC, Ramanathan A (2008) On the relationships between notions of simulation-based security. *J Cryptol* 21:492–546

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)
