Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6692061, 14 pages
https://doi.org/10.1155/2021/6692061

Research Article

Privacy-Preserving Federated Learning Framework with General Aggregation and Multiparty Entity Matching

Zhou Zhou,[1,2] **Youliang Tian**,[1,2] **and Changgen Peng**[1,2]

[1]*State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China*
[2]*Institute of Cryptography and Data Security, Guizhou University, Guiyang 550025, China*

Correspondence should be addressed to Youliang Tian; youliangtian@163.com

Received 3 January 2021; Revised 27 January 2021; Accepted 19 June 2021; Published 28 June 2021

Academic Editor: Zhipeng Cai

The requirement for data sharing and privacy has brought increasing attention to federated learning. However, the existing aggregation models are too specialized and deal less with users' withdrawal issue. Moreover, protocols for multiparty entity matching are rarely covered. Thus, there is no systematic framework to perform federated learning tasks. In this paper, we systematically propose a privacy-preserving federated learning framework (PFLF) where we first construct a general secure aggregation model in federated learning scenarios by combining the Shamir secret sharing with homomorphic cryptography to ensure that the aggregated value can be decrypted correctly only when the number of participants is greater than t. Furthermore, we propose a multiparty entity matching protocol by employing secure multiparty computing to solve the entity alignment problems and a logistic regression algorithm to achieve privacy-preserving model training and support the withdrawal of users in vertical federated learning (VFL) scenarios. Finally, the security analyses prove that PFLF preserves the data privacy in the honest-but-curious model, and the experimental evaluations show PFLF attains consistent accuracy with the original model and demonstrates the practical feasibility.

1. Introduction

In 2016, AlphaGo used 300,000 sets of flag games as training data and beat the world's top professional go players. Artificial intelligence (AI) has shown great potential and is expected to show itself in many fields and make important contributions [1]. In traditional AI, data processing needs to aggregate a large amount of data for model training. However, due to industry competition, privacy protection requirements, business management, and other issues, data of various industries forms islands, which are difficult to share. Therefore, data quality and availability is one of the constraints on AI development [2]. On the other hand, data privacy and security have become the focus of the world's attention [3] following the devastating losses caused by data leaks in recent years. The European Union recently introduced a new law—General Data Protection Regulations (GDPR) [4]—that shows the increasingly strict management of user data privacy and security will be the world trend. So the enactment of laws and regulations also brings new challenges to the traditional AI processing mode.

How to solve the problem of data isolation and data fusion on the premise of protecting users' privacy has become an urgent task for the development of AI. The federated learning (FL) framework, first proposed by Google in 2016 [5, 6], well meets those requirements. In the FL model, each participant keeps the local data training model, only transmits the parameters of each model to an aggregation server using the new cryptography technology, and the server returns the aggregation parameters to each participant for updating after the completion of parameter aggregation [7]. In the end, establishing the virtual common model, using the encryption mechanism to complete the parameter exchange is consistent with the optimal model trained from the data aggregation [8] under the traditional model. Recently, research of the federated learning has become a hot topic, and a lot of deep learning works focusing on privacy protecting have been done. In 2019, Yang et al.

systematically introduced the federated learning framework, application, and research direction [9], which helps us to control and understand federated learning as a whole. The federated learning framework has been applied and extended to Deep Neural Networks (DNN), eXtreme Gradient Boosting (XGBoost), Random Forest (RF), and other algorithms [2, 10, 11], of which the used techniques include secret sharing [12], differential privacy [13], and homomorphic cryptography [14].

At present, there are still the following problems about privacy-protecting FL. First, there are few protocols that involve multiple user entity matching [15] and ensuring their privacy in concurrent mode. In addition, too much interaction between users is required when encrypted gradients or parameters are passed to the server. Furthermore, the protocol only considers that all users participate online throughout the training cycle without going offline or that the recovery of correct data requires the assistance of other participants when one participant goes offline. Finally, the existing schemes are of poor generality and are often only for specific machine learning algorithms and application scenarios.

To solve the above problems, we propose a novel PFLF with general aggregation and multiparty entity matching. In this framework, we propose a general aggregation model (GAM) that can be used in many applications where aggregation is required and privacy is protected. Under our GAM, we construct a multiparty entity matching protocol (MEMP), which can complete the confirmation of the common user data without leaking any disjoint entity information. In addition, we design the vertical federated logistic regression (VFLR) algorithm while keeping the data in the local database. In summary, our contributions can be summarized as follows:

(i) We propose a PFLF that includes the GAM, MEMP, and VFLR to achieve multiscenario data aggregation, multiparty matching, and privacy protections

(ii) We exploit the homomorphic encryption and the improved Shamir secret reconstruction to ensure that only the aggregator receives messages from at least t participants; it can recover the secret and remove mask to acquire correct parameters or product. In the GAM, there is little interaction with other participants

(iii) We propose MEMP to confirm common entities based on GAM and the multiplicative homomorphism of RSA. It can enable participants with different data characteristics to determine common entities without leaking or inferring other useful information

(iv) We design a privacy-preserving VFLR by using Paillier homomorphic encryption and merging GAM and LR. It can train a secure VFLR and support the withdrawal of participants. In addition, the prediction accuracy of the model is not affected

(v) We give a comprehensive security analysis for our framework. We claim that the attackers will not acquire any useful information even if there is no more than t participant collusion. Besides, extensive experiments are operated to confirm that our framework is effective and efficient

The rest of the paper is organized as follows. In Section 2, we describe the preliminaries and the main technology. In Section 3, we describe the system architecture, security model, and problem description. In Section 4, we describe the algorithm details of our GAM with privacy protection and construct the secure MEMP and VFLR model. In Section 5, we demonstrate the security analysis of framework. Experimental evaluation and related work are discussed in Sections 6 and 7, respectively. Finally, we give the conclusion of the paper in Section 8.

2. Preliminary

2.1. Logistic Regression Algorithm. Consider a dataset $\{x^{(i)}, y^{(i)}\}_{i=1}^{m}$ with d dimension, in which $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \cdots, x_d^{(i)}) \in R^d$, $y^{x^{(i)}} \in (0, 1)$. The predicted value is mapped between 0 and 1 by the sigmoid function [16] $h_\theta(x) = 1/(1 + e^{-\theta^T x})$, where $\theta^T x = \sum_{j=0}^{d} \theta_j x_j$ and $x_0 = 1$. The objective function is defined as follows:

$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{d}\theta_j^2. \tag{1}$$

The gradient descent method is used to minimize the value of the objective function, and the model parameters are updated as follows:

$$\theta_j = \theta_j - \frac{\alpha}{m}\sum_{i=0}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} - \frac{\lambda}{m}\theta_j. \tag{2}$$

When given a new data x^{new}, the predictive value of logistic regression is set to

$$y^{\text{new}} = \begin{cases} 1 & h_\theta(x^{\text{new}}) \geq 0.5, \\ 0 & h_\theta(x^{\text{new}}) < 0.5. \end{cases} \tag{3}$$

2.2. Homomorphic Encryption. The Paillier scheme satisfying the additive homomorphism [17] is as follows: p, q are large primes of equal length chosen randomly; $n = pq, \varphi(n) = (p - 1)(q - 1)$ are calculated. Given the random number $g \in \mathbb{Z}_{n^2}^*$, then we have the public key $\text{pk} = (n, g)$ and the private key $\text{sk} = (\varphi(n), \varphi(n)^{-1} \bmod n)$. For the encryption, given the random number r satisfying $0 < r < n, r \in \mathbb{Z}_{n^2}^*$, we have the ciphertext $C = g^m r^n \bmod n^2$, where M is the plaintext. In the decryption phase, m is obtained by computing $m = L(c^{\varphi(n)} \bmod n^2)\varphi(n)^{-1}$, where $L(x) = (x - 1)/n$.

We mainly use the following properties of Paillier homomorphic encryption. Additivity can be indicated as $D(E(m1, r1)E(m2, r2) \bmod n^2) = m1 + m2$.

2.3. Secret Sharing. The secret sharing (SS) scheme [18] adopted in our scheme is used to mask the data ciphertext transmitted by participants, but ensures that the aggregator recovers the ciphertext product and it also helps the scheme support the withdrawal of participants. For (t, n) SS scheme, the secret s is split into n shares. s can be recovered only if at least t random shares are provided. The share generation algorithm is illustrated as $\text{SS.share}(s, t, n) = \{(u, s_u) \mid u \in U\}$, in which n represents the number of participants involved in SS and $U = \{1, 2, \cdots, n\}$ is the set of participants and s_u is the share for each user u. The secret can be recovered by at least t participants contained in $U'(U' \subseteq U)$ using Lagrange interpolation base $\text{SS.recon}(\{(u, s_u) \mid u \in U'\}, t)$ as follows:

$$s = \sum_{i=1}^{|U'|} s_i \iota_i(0), \tag{4}$$

where $\iota_i(0) = \prod_{j=1, i\neq j}^{|U'|} \text{id}_i/(\text{id}_j - \text{id}_i)$ is computed. Here, we can use id_i representing the identity of the ith participant.

3. System Architecture

In this section, we introduce the system architecture, illustrated in Figure 1. Some frequently used notations of the paper are listed in Table 1.

3.1. System Model. Our framework involves three types of participating entities: a key generation center (KGC), a center server (CS), and a set of average participants (AP). Details are presented as follows.

Key Generation Center. KGC primarily performs key generation and distribution. Its main purpose is to initialize the system, generate public and private keys for homomorphic encryption, generate subsecrets based on Shamir secret sharing, assign corresponding public and private keys to CS, and distribute subsecrets to each general participant. Afterwards, it will go offline.

Center Server. CS is often the initiator of a federated learning mission. It is the one who has data labels, coordinating the execution of the entire process. It aggregates the parameters uploaded by all online participants. In MEMP, it calculates the user intersection and returns the common entity. In VFLR, it returns the calculated sample error. In the process, we hope that CS can infer nothing except the uploaded ciphertext and the final result.

Average Participants. AP refers to participants who participate in model training without tags. It involves multiple average participants, namely, $\text{AP} = \{U_1, U_2, \cdots, U_n\}$. In the aggregation framework, they are usually done with local encryption and send values for aggregation.

3.2. Security Model. Based on [19], in our scheme, we assume that the interaction channels through CS and AP are secure and not subject to risks such as tampering, and all participants except KGC are considered to be honest-but-curious. KGC is a trusted party which always performs its tasks honestly and does not collude with any entity. CS and AP honestly follow the agreed process, but may try to learn all possible information that is of interest to them from their received messages. We define a threat model with an honest-but-curious adversary $\mathcal{A}$ who can corrupt at most $t - 1$ parties and obtain their inputs or other private information. In the entity matching protocol, what $\mathcal{A}$ wants to know is users' information and CS's private key. In the model building and prediction phase, $\mathcal{A}$ makes full use of the information it holds to learn about the data including data characteristics and weights of other honest parties. Our model needs to meet the following security requirements.

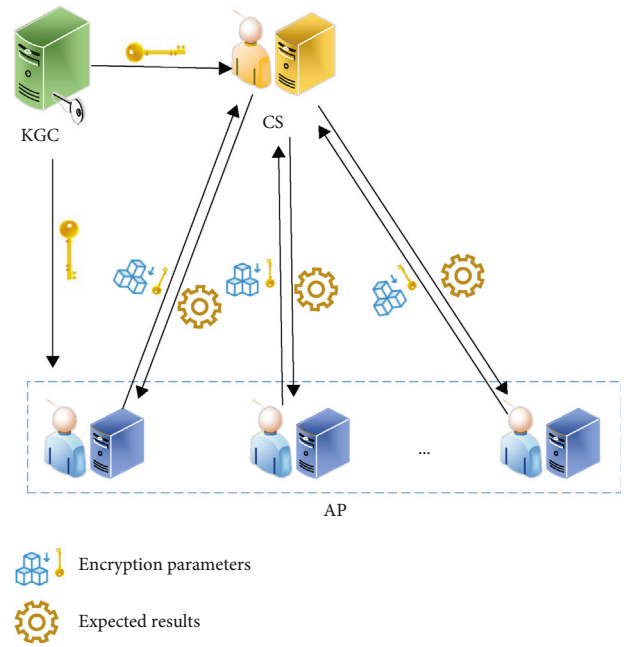

Figure 1: System architecture.

Data Privacy. CS cannot recognize any private data uploaded by U_i, and other U_j $(j \neq i)$ cannot infer the private data of others. For example, mark matrix and model parameters must not be exposed.

Secure Withdrawal. CS and U_i cannot continue to use the information of the exiting participants for subsequent calculations, and the process of recovering the aggregated value cannot reveal the parameters of the exiting participants if any participant drops in a round. There should be a safe way to deal with delayed transmissions and not be mistaken for offline.

3.3. Problem Description. In order to achieve a GAM that can enable aggregated messages to be decrypted only if they come from at least t participants, while ensuring that individual parameter ciphertext is not exposed, we introduce some cryptographic tools. For example, the transformed Shamir secret sharing scheme helps achieve threshold aggregation and cover the homomorphic ciphertext of each participant. Homomorphic encryption features facilitate obtaining the product or sum of parameter plaintext through ciphertext

Table 1: Notation table.

Notation	Description
E_{k_p}	The homomorphic encryption with public key k_p
e, d	The public key and private key for RSA
$\mathbb{Z}_P^*$	A cyclic group of order P, primitive g
s_i	The secret share distributed to the participants U_i
X_k^i	The kth sample of the ith participant
Θ_i	The characteristic weight of the ith participant
$H()$	A hash function
$h_\theta()$	Function prediction of logistic regression

aggregation. Furthermore, the same entity between different participants needs to be determined for multiple participants with different characteristics in the vertical federated mode. Firstly, we should design a MEMP with privacy protection, through which multiple participants obtain their overlapping entity IDs without exposing their respective data. After then, we use these common entities' data with different characteristics to train the learning model while ensuring local data privacy. To achieve these two goals, under our aggregation model, we use RSA blind signature to generate data identity libraries, record the matching results by token matrix, and use RSA and Paillier as homomorphic encryption for specific functional requirements. In particular, the prediction accuracy of VFLR realized by the framework is unchanged, and it can support the withdrawal of participants.

4. Construction of PFLF

Our PFLF implements a systematic FL process, including three main functions. Firstly, it can realize multiparty data aggregation without data leakage. Secondly, it can find the common set of entities of multiple participants without revealing useful information. For VFL scenarios, subsequent joint training can only be completed if the common entities are identified. When using logistic regression algorithm in VFL scenario, secure data aggregation is necessary after entity matching is completed. So, thirdly, we design the VFLR. In particular, the aggregation in our framework is generic, not only for a specific machine learning algorithm but also for all application scenarios based on thresholds. In this section, we present the details of our GAM and its role in constructing MEMP and VFLR.

4.1. A Novel GAM. A common aggregation model is suitable for such application scenarios where the aggregation server CS can decrypt and obtain the desired results through homomorphic encryption only when messages received are from at least t participants. Through this model, the participants' private data is fully protected in the process of achieving the interaction purpose according to the protocol, and when there are participants offline, the aggregated messages that do not involve the offline information can be recovered quickly. Here, firstly, based on the mentioned SS scheme, we can make the following transformation.

Each user U_i chooses a random number β_{ik}, where $i = \{1, 2, \cdots, t\}$ is the number of participants who reconstructed the secret and k is the number of samples. For the secret reconstruction formula $s = \sum_{i=1}^{|U'|} s_i \iota_i(0)$, let us multiply both sides of this equation by random numbers β_{ik} and the equation is transformed into the following form:

$$\begin{aligned}
\beta_{1k} s &= \beta_{1k}(s_1\iota_1(0) + s_2\iota_2(0) + \cdots + s_t\iota_t(0)),\\
\beta_{2k} s &= \beta_{2k}(s_1\iota_1(0) + s_2\iota_2(0) + \cdots + s_t\iota_t(0)),\\
&\cdots\\
\beta_{tk} s &= \beta_{tk}(s_1\iota_1(0) + s_2\iota_2(0) + \cdots + s_t\iota_t(0)).
\end{aligned} \tag{5}$$

We can sum both sides of this equation and get

$$\sum_{i=1}^{t} \beta_{ik} s = \sum_{i=1}^{t} \beta_{ik} s_1 \iota_1(0) + \sum_{i=1}^{t} \beta_{ik} s_2 \iota_2(0) + \cdots + \sum_{i=1}^{t} \beta_{ik} \beta_{ik} s_t \iota_t(0). \tag{6}$$

When used for threshold encryption, it converts to

$$\begin{aligned}
&\left\{ g^{\sum_{i=1}^{t} \beta_{ik} s_1 \iota_1(0)} E_k(m_{1k}) \right\} \left\{ g^{\sum_{i=1}^{t} \beta_{ik} s_2 \iota_2(0)} E_k(m_{2k}) \right\} \cdots \left\{ g^{\sum_{i=1}^{t} \beta_{ik} s_t \iota_t(0)} E_k(m_{tk}) \right\}\\
&= \left\{ g^{\sum_{i=1}^{t} \beta_{ik} s_1 \iota_1(0) + \sum_{i=1}^{t} \beta_{ik} s_2 \iota_2(0) + \cdots + \sum_{i=1}^{t} \beta_{ik} s_t \iota_t(0)} \right\} \prod_{i=1}^{t} E_k(m_{ik})\\
&= \left\{ g^{\sum_{i=1}^{t} \beta_{ik} s} \right\} \prod_{i=1}^{t} E_k(m_{ik}).
\end{aligned} \tag{7}$$

We assume that in such a scenario, each participant U_i having a message m_{ik} needs to ask CS to help calculate $\sum_{i=1}^{n} m_{ik}$, but does not want to disclose m_{ik} to others and also does not want CS to infer some private data through the message sent by themselves to carry out various possible calculations. Figure 2 shows its workflow. In this way, they can do it like this: U_i first computes $g^{\sum_{i=1}^{t} \beta_{ik} s_i \iota_i(0)} E_k(m_{ik})$ and sends it to CS according to the above equation, the receiver with the private key k_s can decrypt and get $\sum_{i=1}^{t} m_{ik}$ when each participant makes public the value $g^{\beta_{ik}}$ corresponding to β_{ik}, and there exists the secret $s = k_p$ and the public key $k = k_p$. Besides, $E_k()$ satisfies homomorphic encryption which refers to multiplicative homomorphism in our entity matching protocol and additive homomorphism in our joint model training. How the model supports users' withdrawal is described in detail in Section 4.3. If the model is used for horizontal federated learning, m_{ik} can be gradient and other important parameters. Later in the VFLR section, we will focus on using logistic regression to explain the framework.

4.2. Secure Multiparty Entity Matching. As shown in Algorithm 1, the secure multiparty entity matching protocol

completes the confirmation of the common entity of multiple participants under the premise of protecting privacy. In the protocol, there is a CS with data sample identity u_{sk} and a set of average participants $U = \{U_1, U_2, \cdots, U_n\}$ with their own data sample u_{ik}, which represents the identity of the k th sample for the ith participant. Note that we briefly describe the situation of sending a message M from A to B as $A \Rightarrow B : M$. The protocol workflow is shown in Figure 3. The process of the protocol is shown as below.

In the initial parameter setting phase, the CS sets the penalty term, coefficient λ, and maximum iteration number ma $x_i$ter of the model. T generates a public-private key (k_p, k_s) for homomorphic encryption of the later model building and predicting. The algorithm introduces an RSA encryption with a blind factor to mask confidential information, so the public-private key $((N, e), d)$, $((N, e_i), d_i)$, $(i \in (1, 2, \cdots, n))$ are generated by T. In the following, we omit the modulus N for RSA. In addition, T also generates subshares s_i and s_i' of the public key k_p and e for U_i based on the identity of U_i using SS.share(k_p, t, n) or SS.share(e, t, n).

In the exchange of information phase, each participant U_i chooses a random number r_i, computes $v_{ik} = (r_i)^e (H(u_{ik})^{d_i})$, and sends it to CS. CS chooses a random number r_{s_i} to reflect the randomness of the interaction and uses the private key d for signature $(v_{ik})^d r_{cs}$ and sends disturbed $c_{ik} = r_i (H(u_{ik}))^{d_i d} r_{cs}$ to U_i. Disturbing the order of $c_{i1}, c_{i2}, \cdots, c_{ik}$ can eliminate the correspondence between c_{ik} and $u_i k$ so that U_i cannot lock the IDs corresponding to the intersection of U_i and CS in the comparison phase. Furthermore, CS calculates random identifiers of each entity for different participants: $d_{ij} = H(H(u_{sj})^d r_{s_i}^{e_i})$. Then, CS sends $[d_{11}, d_{12}, \cdots, d_{1j}], \cdots, [d_{n1}, d_{n2}, \cdots, d_{nj}]$ to corresponding participants $U_1, U_2, \cdots, U_{n-1}$ for comparison.

In the comparison phase, each participant U_i eliminates the blind factor and opens its signature for c_{ik} to obtain $(H(u_{ik}))^d r_{s_i}^{e_i}$ and uses the hash function to compute $g_{ik} = H((H(u_{ik}))^d r_{s_i}^{e_i})$. By comparing them with d_{ij}, CS gets an l -dimensional matrix $[b_{i1}, b_{i2}, \cdots, b_{ij}]$. $b_{ij} = 1$ if d_{ij} belongs to g_{ik}. If not, $b_{ij} = a_{ij}$, where a_{ij} is a random number and $a_{ij} > 1$. In this way, each participant creates a comparison matrix.

In the solution phase, each participant U_i chooses a set of random number p_{ij} (i.e., $\{p_{11}, p_{12}, \cdots, p_{1j},\}, \cdots, \{p_{n1}, p_{n2}, \cdots, p_{nj},\}$) and encrypts its matrix by computing $g^{s_{i'} \iota_i(0) \sum_{i=1}^{n} p_{ij}} (b_{ij})^e$ with RSA.

After then, U_i sends their encrypted matrix to CS for aggregation. At last, CS aggregates matrix values of n participants by computing $\prod_{i=1,j}^{n,j} g^{s_{i'} \iota_i(0) \sum_{i=1}^{n} p_{ij}} (b_{ij})^e$ and uses its private key d to decrypt, obtaining $\prod_{i=1,j}^{n,j} b_{ij}$. Because CS can recover the secret e by computing $\sum_{i=1}^{s_i'} \iota_i(0)$. Specific analysis can refer to Secure Model Building.

In the identification phase, CS finds the corresponding entity by the value of $\prod_{i=1,j}^{n,j} b_{ij}$. Because each d_{ij} comes from CS, then if it is 1, it means that each participant has a corresponding d_{ij}, and if not, it means that at least one participant does not have d_{ij}. Therefore, CS can find the right entity u_{sj} based on $\prod_{i=1,j}^{n,j} b_{ij}$. In the end, CS broadcasts common entity IDs to other participants for following model training.

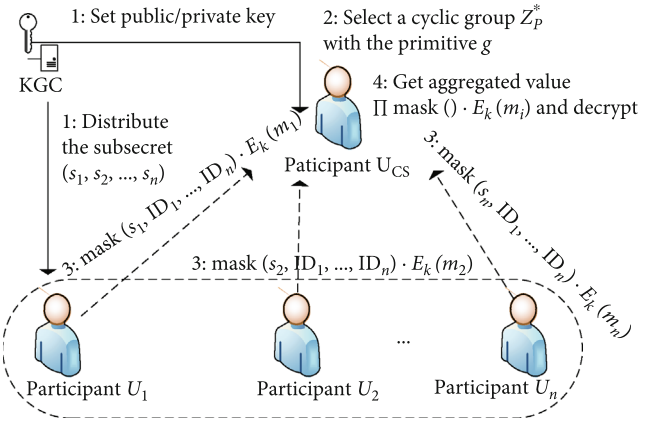

Figure 2: Workflow of the GAM.

4.3. Secure Model Building

4.3.1. Secure Training. For logistic regression to find the better model with gradient descent method, the part that needs to be computed jointly is error using the predicted value and the sample label. It is better for CS to do the aggregation and calculation since sample labels are mastered by CS. The workflow is shown in Figure 4. To protect the confidentiality of each participant's data, the aggregated data is received in ciphertext. Consequently, we chose to use Paillier homomorphic encryption to do the computation. However, each participant's data cannot be decrypted separately by CS; for this reason, we apply Shamir secret sharing scheme such as Formula (7) to ensure that CS could decrypt only after the aggregation operation was completed. The detailed process is shown in Algorithm 2.

The number of common entities of all participants is m, the average participants in the joint modeling are $U_i (U_i \in U)$, and each average participant secretly keeps a subsecret as s_i. Now given a cyclic group G and its primitive g, $U_i (i \in [1, n])$ computes $D_k^i = \Theta_i X_k^i$, in which X_k^i is the kth sample for the ith participant U_i. With subsecret s_i and identities, U_i can compute its own $s_i \iota_i(0)$. In order to keep the subsecret s_i dynamic, for each sample, U_i adds a random factor or the time stamp β_{ik} to calculate $C_k^i = g^{s_i \iota_i(0) \sum_{i=1}^{n} \beta_{ik}} E_{k_p}(D_k^i)$ and sends it to CS after other participants release their $g^{\beta_{ik}}$. At this point, although CS gets the ciphertext of each participant, he cannot decrypt it because he cannot get the subsecret s_i. Only when messages are received from at least t participants can aggregation $\prod_{i=1}^{n} C_k^i$ that can be generated, i.e.,

$$\prod_{i=1}^{n} C_k^i = \prod_{i=1}^{n} g^{s_i \iota_i(0) \sum_{i=1}^{n} \beta_{ik}} E_{k_p}(D_k^i) = g^{\sum_{i=1}^{n} \left\{ s_i \iota_i(0) \sum_{i=1}^{n} \beta_{ik} \right\}} E_{k_p}\left(\sum_{i=1}^{n} D_k^i \right), \tag{8}$$

Input: a central server CS, a set of participants $U = \{U_1, U_2, \cdots, U_n\}$, and a trusted party T.
Output: common entity IDs.
1: CS sets the parameters for model training penalty, λ, $\max_i$ter.
2: T generates a public-private key (k_p, k_s) for homomorphic encryption, a public-private key $((N, e), d)$ for RSA encryption for CS, also generates average participants' public-private keys $((N, e_i), d_i)$ and subshares of the public key k_p and e based on the identity of the participants, i.e., $T \Rightarrow U : \{(u, s_u) \mid u \in U\} = \text{SS.share}(k_p, t, n)$. Here, $U_1, U_2, \cdots, U_n$ get subshares $\{s_1, s_2, \cdots, s_n\}$ for k_p and subshares $\{s_1', s_2', \cdots, s_n'\}$ for e.
3: Each participant U_i chooses a random number r_i, computes $v_{ik} = (r_i)^e (H(u_{ik})^{d_i})$, and operates $U \Rightarrow CS : v_{ik}$.
4: CS chooses a random number r_{s_i}, uses the private key for signature $(v_{ik})^d r_{s_i}$, gets each $c_{ik} = r_i (H(u_{ik}))^{d_i d} r_{s_i}$, and returns them to U_i after disturbing the order of c_{ik}.
5: **for** $i = 1 \longrightarrow n$ **do**
6: **for** $j = 1 \longrightarrow l$ **do**
7: CS computes for the l entities: $d_{ij} = H(H(u_{sj})^d r_{s_i}^{e_i})$.
8: CS sends $[d_{11}, d_{12}, \cdots, d_{1j}], \cdots, [d_{n1}, d_{n2}, \cdots, d_{nj}]$ to corresponding participants $U_1, U_2, \cdots, U_{n-1}$.
9: **for** $i = 1 \longrightarrow n$ **do**
10: Each participant U_i eliminates the blind factor and d_i for c_{ik}, obtains $(H(u_{ik}))^d r_{s_i}^{e_i}$, and computes their hash values $g_{ik} = H((H(u_{ik}))^d r_{s_i}^{e_i})$.
11: **for** $j = 1 \longrightarrow l$ **do**
12: Each participant generates its own l-dimensional matrix $[b_{i1}, b_{i2}, \cdots, b_{ij}]$ by determining whether d_{ij} belongs to its g_{ik}.
13: **if** $d_{ij} \in g_{ik}$ **then**
14: $b_{ij} = 1$.
15: **else**
16: $b_{ij} = a_{ij}$, where a_{ij} is a random number and $a_{ij} > 1$.
17: Each participant U_i chooses a set of random number p_{ij} and encrypts its matrix with $g^{s_{i'} \iota_i(0) \sum_{i=1}^{n} p_{ij}} (b_{ij})^e$, operating $U_i \Rightarrow$ CS : encryptedmatrix.
18: CS aggregates matrix values by computing $\prod_{i=1,j}^{n,j} g^{s_{i'} \iota_i(0) \sum_{i=1}^{n} p_{ij}} (b_{ij})^e$ and obtains $\prod_{i=1,j}^{n,j} b_{ij}$ by decrypting.
19: **if** $\prod_{i=1,j}^{n,j} b_{ij} == 1$ **then**
20: CS finds the corresponding u_{sj}.
21: CS broadcasts u_{sj} to other participants $U_1, U_2, \cdots, U_{n-1}$.
22: **return** common entity IDs: u_{sj}.

Algorithm 1: MEMP.

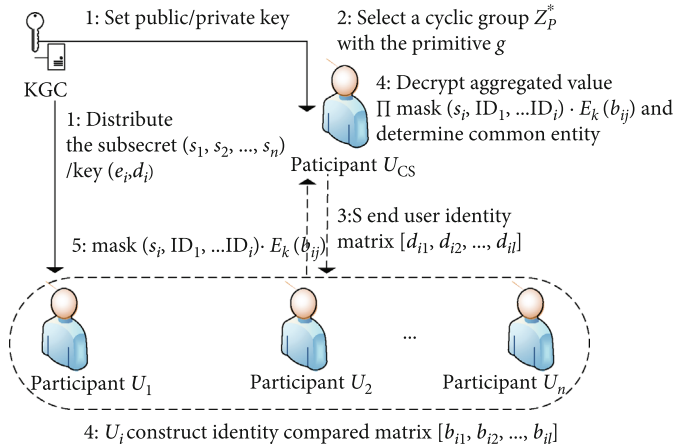

Figure 3: Workflow of the MEMP.

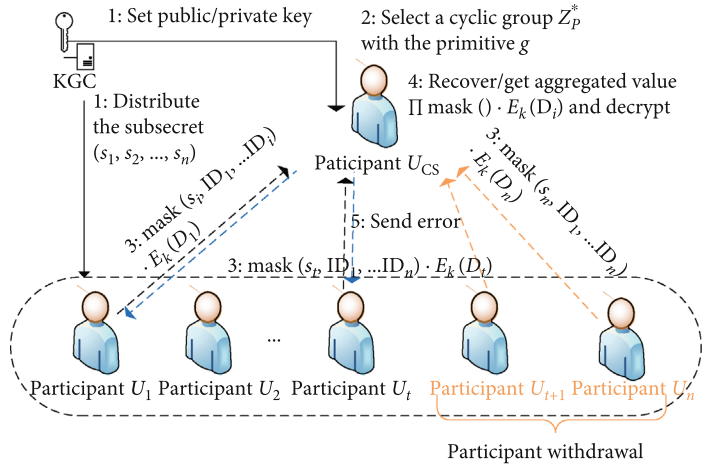

Figure 4: Workflow of the VFLR.

Input: a central server CS, a set of participants $U = \{U_1, U_2, \cdots, U_n\}$, instance space of m samples of each participants, subshares $\{s_1, s_2, \cdots, s_n\}$, cyclic group G, and its primitive g.
Output: federated logistic regression model.
1: **for** $k = 1 \longrightarrow m$ **do**
2: **for** $i = 1 \longrightarrow n$ **do**
3: U_i computes $D_k^i = \Theta_i X_k^i$.
4: U_i chooses random number β_{ik} and makes public its $g^{\beta_{ik}}$.
5: U_i uses others' $g^{\beta_{ik}}$ to compute $C_k^i = g^{s_i \iota_i(0) \sum_{i=1}^n \beta_{ik}} E_{k_p}(D_k^i)$ and sends it to CS.
6: **for** $k = 1 \longrightarrow m$ **do**
7: **if** someone exits **then**
8: CS eliminates the value involving information of quitters in $\iota_i(0)$.
9: CS performs the aggregation $\prod_{i=1}^n C_k^i$ and decrypts to get $\sum_{i=1}^n \Theta_i X_k^i$.
10: CS computes $\text{error}_k = y^{(k)} - 1/(1 + e^{-(\sum_{i=1}^n \Theta_i X_k^i + \Theta_s X_k^s)})$.
11: broadcasts error_k.
12: Each participant and CS can update weight parameter by computing $\Theta_i = \Theta_i + (\alpha/m) \sum_{k=1}^m \text{error}_k X_k^i - (\lambda/m)\Theta_i$.
13: Repeat all until reaching the termination condition.
14: **return** built model.

Algorithm 2: VFLR.

where $g^{\sum_{i=1}^n \{s_i \iota_i(0) \sum_{i=1}^n \beta_{ik}\}} = g^{\sum_{i=1}^n \beta_{ik} k_p}$. Because k_p and $g^{\sum_{i=1}^n \beta_{ik}}$ are public to CS, CS can decrypt and get $\sum_{i=1}^n \Theta_i X_k^i$ to compute error_k. The next step, CS will broadcast error_k to current participants $U'(U' \subseteq U)$. Each participant and CS can update weight parameter by computing $\Theta_i = \Theta_i + (\alpha/m) \sum_{k=1}^m \text{error}_k X_k^i - (\lambda/m)\Theta_i$. All steps are repeated until the maximum number of iterations is reached.

4.3.2. Withdrawal of Participants. Some participants may withdraw from federated learning, such as being unwilling to contribute models or dropping offline. In order to deal with the above situation, we can make a contract to reduce the occurrence of withdrawal. It is assumed that each average participant U_i will be paid a certain amount based on their contribution in each iteration. The total expenditure and maximum number of iterations set by CS are Amounts and t. The contract signed by all participants is as follows: (1) n average participants submit a deposit to CS, and the deposit is Amounts/t. (2) In the FL, if CS receives all the messages within the maximum allowable period, the errors will be returned normally to all participants according to the protocol. (3) Else, CS will send withdrawal confirmation request to participants who did not send the message. If they report it is delayed, the delayed messages can still be used to compute the aggregated value. But these delayed participants will not get paid this round. (4) Once the participant reports withdrawal, the deposit will be distributed to other online participants, including CS. Rational participants generally do not withdraw in order to maximize their own interests. (5) Upon completion of the FL, deposits will be returned to all the online participants. Particularly, if some participants withdraw during training, CS requires each participant to resend the message without the identity of the quitters in order to decrypt. Because of the randomness of C_k^i, CS cannot use the message sent twice to perform comparison calculation and get useful data. The reason is that CS still cannot get the subsecret to reconstruct the polynomial.

Input: federated logistic regression model, results inquirer R, and its instance space.
Output: predicted results.
1: **for** $i = 1 \longrightarrow n$ **do**
2: R, respectively, encrypts the data $E_{k_p'}(X_k^i)$ belonging to the characteristics of different participants including CS.
3: Each participant computes $(E_{k_p'}(X_k^i))^{\Theta_i}$ and communicates it to CS.
4: CS does an aggregate operation $(E_{k_p'}(X_k^s))^{\Theta_s}\prod_{i=1}^{n}(E_{k_p'}(X_k^i))^{\Theta_i}$ and returns it to R.
5: R decrypts and gets $\Theta_i X_k^i$.
6: R gets predicted results h by $h_\theta(x) = 1/(1 + e^{-\theta^T x})$.
7: **return** result h.

Algorithm 3: Secure predicting.

4.4. Secure Predicting. Secure prediction should ensure that user data privacy is not compromised and that model parameters are not exposed. As described by Algorithm 3, results inquirer R intends to provide a set of data for prediction without privacy leakage and all data characteristics correspond to all participants in the current model. First, R, respectively, encrypts the data with $E_{k_p'}(X_k^i)$, for example, (X_k^i) belongs to a characteristic that corresponds to U_i. Each participant computes $(E_{k_p'}(X_k^i))^{\Theta_i}$ through R's public key k_p $'$ and communicates it to CS. The aggregation operation is still done by CS to protect the parameters Θ_i from being exposed. Then, CS computes $(E_{k_p'}(X_k^s))^{\Theta_s}\prod_{i=1}^{n}(E_{k_p'}(X_k^i))^{\Theta_i}$ and returns it to R. The joint value $\Theta_i X_k^i$ could be get with private key k_s' by R, which computes predicted results h using $h_\theta(x) = 1/(1 + e^{-\theta^T x})$.

5. Security Analysis

In this section, we prove that our scheme is secure based on the simulator under the honest-but-curious setting. Recall that the involved parties are n participants $U_1, U_2, \cdots, U_n$ and the CS. We assume an honest-but-curious adversary $\mathcal{A}$ who can corrupt participants but at most $t - 1$. According to [20], we define the security of our framework by comparing the real interaction and ideal interaction. In the real interaction, there is an environment $\mathcal{Z}$ which chooses inputs and receives outputs of uncorrupted participants. The adversary $\mathcal{A}$ who can interact arbitrarily with the environment $\mathcal{Z}$ forwards all received messages to $\mathcal{Z}$ and acts as instructed by $\mathcal{Z}$. We let REAL$[\mathcal{Z}, \mathcal{A}, \pi]$ represent the view of $\mathcal{A}$. Similarly, we let IDEAL$[\mathcal{Z}, \mathcal{S}, \mathcal{F}]$ represent the view of $\mathcal{S}$ where adversary $\mathcal{S}$ and honest participants interact with the environment $\mathcal{Z}$ running the dummy protocol in the presence of functionality $\mathcal{F}$.

Definition 1. A protocol π is secure if for every admissible adversary $\mathcal{A}$ attacks the real interaction, there exists a simulator $\mathcal{S} = [\mathcal{S}_u, \mathcal{S}_{cs}]$ attacking the ideal interaction, such that the environment $\mathcal{Z}$ cannot distinguish between the ideal view of $\mathcal{S}$ and the real view of $\mathcal{A}$.

5.1. Security of GAM. In the GAM, although CS can collude with at most $t - 1$ participants to obtain the privacy of honest participants, they get nothing but aggregated results. Since each participant's data is encrypted as $g^{\sum_{i=1}^{t}\beta_{ik}s_1 t_i(0)}E_k(m_{ik})$, based on the security of Shamir secret sharing and homomorphic encryption, only the privacy of m_{ik} is discussed here.

Theorem 2. *For secure aggregated framework, there exists a PPT simulator $\mathcal{S}_u$ or $\mathcal{S}_{cs}$ that can simulate the ideal view IDEAL$[\mathcal{Z}, \mathcal{S}, \mathcal{F}_{gam}]$ which is computationally indistinguishable from the real view of $\mathcal{A}_u$ or $\mathcal{A}_{CS}$. $\mathcal{F}_{gam}$ is illustrated as Table 2, where a subset $W \subseteq U \cup CS$ represents joint attackers.*

According to whether CS is involved in collusion, the discussion is divided into two situations.

Case 1. Excluding CS from W.

Proof. Since CS is not compromised, the view constructed by simulator $\mathcal{S}_u$ is independent of the input of CS. So the simulator $\mathcal{S}_u$ can execute a simulation by asking $\mathcal{F}_{gam}$ to generate fake data as inputs of the honest users, but the true inputs for honest-but-curious users. When sending $b(r_i)f_k(x_i)$, the simulator utilizes random number instead of true data. As aggregating and decrypting, CS returns aggregated result that does not indicate which special participants' $b(r_i)f_k(x_i)$ are aggregated. Hence, the ideal view simulated by $\mathcal{S}_u$ is indistinguishable from the real view of $\mathcal{A}_u$ since it is impossible to determine that $\sum_{i=1}^{|U\supseteq W|}x_i$ is obtained from real data. □

Case 2. Including CS in W.

Proof. To prove the indistinguishability of the views in Case 2, the simulator gradually makes some improvements to the protocol. There exists hyb1, hyb2 that imply secure modification to the original protocol, ensuring the indistinguishability of the changed protocol from the original protocol, in our hybrid argument.

hyb1: in this hybrid, the simulator $\mathcal{S}_{CS}$ generates the masked input for honest participants as below:

$$y_i = r_i E_k(m_{ik}), \tag{9}$$

Table 2: Definition of $\mathcal{F}_{\text{gam}}$.

Give function $f_k(x_1, x_2, \cdots x_n) = k^{-1}\prod_{i=1}^{n} b(r_i) f_k(x_i)$ with homomorphic encryption function f_k and $\prod_{i=1}^{n} b(r_i) = k$ when $i \geq t$.
$\mathcal{F}_{\text{gam}}$'s operation is as follows:
(1) On input (Input, *sid*, m_i) from U_i, set $x_i = m_i$ and send (Input, *sid*, U_i, $\lvert m_i \rvert$) to adversary $\mathcal{A}$.
(2) On input (Compute, *sid*, U_i) from U_i, choose $r_i \in_R Z_P^*$ randomly, compute $b(r_i) f_k(x_i)$, and send them to CS if $i \geq t$. If $U_i \in W \subseteq U$ is corrupted, send (x_i, r_i) to it.
(3) On input (Aggregate, *sid*, CS), compute f_k^{-1} $\left(k^{-1}\prod_{i=1}^{n} \boxtimes b(r_i) f_k(x_i)\right)$ and send it to CS.

instead of utilizing

$$y_i = g^{\sum_{i=1}^{t} \beta_{ik} s_i t_i(0)} E_k(m_{ik}). \tag{10}$$

Because $g^{\sum_{i=1}^{t} \beta_{ik}}$ is a random number, we can get $g^{\sum_{i=1}^{t} \beta_{ik} s_i t_i(0)}$ is also a random value. The DDH assumption ensures that it is easy to infer they are indistinguishable.

hyb2: in this hybrid, the simulator generates encrypted $E_k(\mathrm{t}_i)$ by replacing m_{ik} with a random number t_i. The security of the encryption algorithm ensures the indistinguishability of the two ciphertexts. Therefore, the simulator submits

$$y_i = r_i t_i, \tag{11}$$

instead of sending

$$y_i = r_i E_k(m_{ik}). \tag{12}$$

Accordingly, the simulation has been completed since $\mathcal{S}_{\text{CS}}$ successfully simulates the real view without acquiring $x_i(m_{ik})$ and subsecret s_i and we can infer that the output of this hybrid is indistinguished from the real one. □

5.2. Security of MEMP. In the process of confirming the identity of the common entity, no entity information other than a common identity is available between participants. Thus, not only can the true identity of the entity not be exposed but its hash value cannot be either. The reason is that some honest-but-curious participants can calculate the hash value of a possible identifier to determine whether it belongs to U_i. It is necessary to cover the confidential information with a random factor similar to a blind signature. We prove the following two theorems by constructing two separate simulators $\mathcal{S}_{\text{u}}$ and $\mathcal{S}_{\text{cs}}$ to show that the real views and ideal views are computationally indistinguishable.

Theorem 3. *For group entity matching, there exists a PPT simulator $\mathcal{S}_u$ or $\mathcal{S}_{cs}$ that can simulate the ideal view IDEAL$[\mathcal{Z}, \mathcal{S}, \mathcal{F}_{em}]$ which is computationally indistinguishable from the real view of $\mathcal{A}_u$ or $\mathcal{A}_{cs}$.*

Table 3: Definition of $\mathcal{F}_{\text{em1}}$.

$\mathcal{F}_{\text{em1}}$'s operation is as follows:
(1) On input (Init, *sid*, id_i, $(\mathrm{ID}_{i1}, \mathrm{ID}_{i2}, \cdots, \mathrm{ID}_{ik})$) from U_i, generate random numbers $(h_{i1}, h_{i2}, \cdots, h_{ik})$ for the corresponding ID_{i1}, I $\mathrm{D}_{i2}, \cdots, \mathrm{ID}_{ik}$, store $(\mathrm{id}_i, (\mathrm{ID}_{i1}, h_{i1}), (\mathrm{ID}_{i2}, h_{i2}), \ldots)$, and send $(\mathrm{id}_i, (\mathrm{ID}_{i1}, h_{i1}), (\mathrm{ID}_{i2}, h_{i2}), \ldots)$ to corrupted adversary A.
(2) On input (Init, *sid*, CS, $(\mathrm{ID}_{s1}, \mathrm{ID}_{s2}, \cdots, \mathrm{ID}_{sl})$) from CS, generate l random numbers $(h_{s1}, h_{s2}, \cdots, h_{sl})$.
Once the IDs of the CS and U_i are equal, there sets $h_{sj} = h_{it}$, $(j \in [1, k], t \in [1, l])$.
For corrupted CS, send $(\mathrm{id}_s, (\mathrm{ID}_{s1}, h_{s1}), (\mathrm{ID}_{s2}, h_{s2}), \cdots)$ to it. Store them.
(3) On input (signature, *sid*, U_i, d_i) from U_i, choose $r_i \in_R Z_P^*$ randomly, and compute sign $(d_i, r^{ee_i} h_{it})$.
Send them to CS and store (U_i, sid, r_i).
(4) On input (Bsignature, sid, CS, d, sign $(d_i, r^{ee_i} h_{it})$) from CS, compute sign $(d, \text{sign } (d_i, r^{ee_i} h_{it}))$.
Send r_{s_i} sign $(d, \text{sign } (d_i, r^{ee_i} h_{it}))$ to U_i, where r_{s_i} are random numbers.
(5) On input (Open, *sid*, U_i, sign $(d, \text{sign } (d_i, r^{ee_i} h_{it}))$) from U_i. Check whether there exists (U_i, sid, r_i).
If exiting, $h_{it}^d r_{s_i}^{e_i}$ can be got, or abandon it. Generate random H_{it} and store $(U_i, \mathit{sid}, \left(h_{it}^d r_{s_i}^{e_i}, H_{it}\right))$.
(6) On input (Sign, *sid*, *CS*, U_i, d), compute (sign $\left(d, h_{sj} r_{s_i}^{ee_i}\right)$). Generate random H_{sj} for every sign $\left(d, h_{sj} r_{s_i}^{ee_i}\right)$.
If existing sign $\left(d, h_{sj} r_{s_i}^{ee_i}\right) = \left(h_{it}^d r_{s_i}^{e_i}\right)$, setting $H_{sj} = H_{it}$ and storing (CS, *sid*, $\left(h_{it}^d r_{s_i}^{e_i}, H_{sj}\right)$). Send (CS, *sid*, H_{sj}) to the corresponding U_i.
(7) On input (Compare, U_i, *sid*, CS), generate the l – dimensional comparison matrix using 1 and random numbers, where there exist H_{sj} in $(U_i, \mathit{sid}, \left(h_{it}^d r_{s_i}^{e_i}, H_{it}\right))$, 1 will be set, or a random number will be set.

Let us divide $\mathcal{F}_{\text{em}}$ into two parts, such as $\mathcal{F}_{\text{em1}}$ and $\mathcal{F}_{\text{em2}}$. $\mathcal{F}_{\text{em1}}$ is the process from the beginning to the establishment of the comparison matrix, and $\mathcal{F}_{\text{em2}}$ means the process from the encrypted comparison matrix to the end, similar to the previous $\mathcal{F}_{\text{gam}}$. The description of $\mathcal{F}_{\text{em1}}$ is shown in Table 3.

Here, we only complete the proof of the ideal view IDEAL$[\mathcal{Z}, \mathcal{S}, \mathcal{F}_{\text{em1}}]$ which is computationally indistinguishable from the real view of $\mathcal{A}_{\text{u}}$ or $\mathcal{A}_{\text{cs}}$. There are still two cases to prove.

Case 1 (excluding CS from W). Just consider the average participants U_i that are compromised.

Proof. Suppose a group of participants is corrupted in the beginning. Because the views of corrupted participants and the inputs of honest participants are irrelevant and the values of all honest participants are meaningless for corrupted participants according to the real protocol π_{em}, so the simulator $\mathcal{S}_{\text{u}}$ first can run the protocol with the true inputs while using dummy data as the inputs of honest participants, namely, $\mathcal{S}_{\text{u}}$ asks $\mathcal{F}_{\text{em1}}$ to generate random numbers as the inputs of honest particpants. After the blind signature is executed, the value returned from CS is added with a random number of

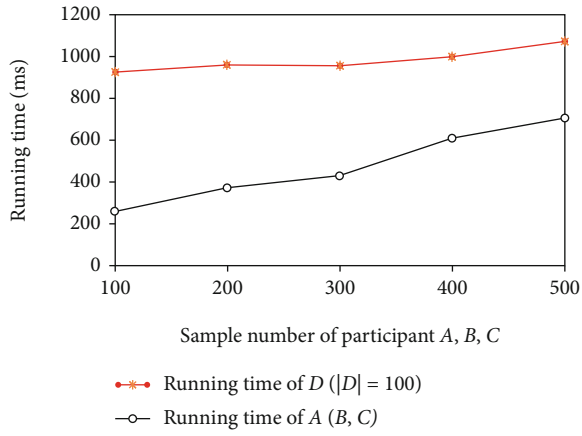

Figure 5: Running time with different sample sizes in U_i.

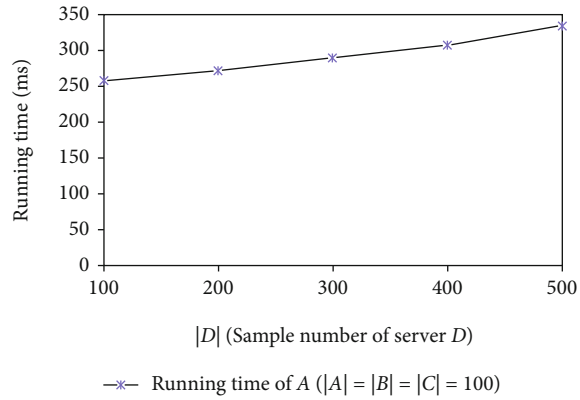

Figure 6: Running time of A with increasing samples in D.

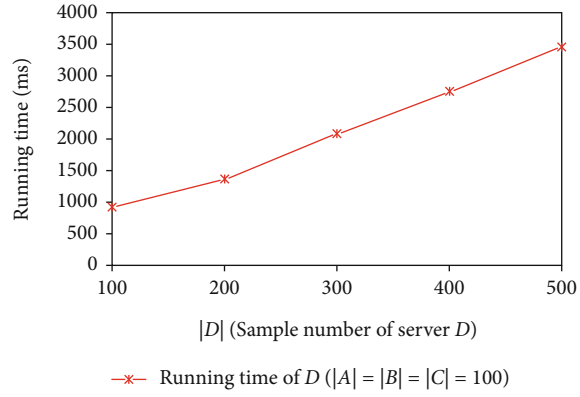

Figure 7: Running time of D with increasing samples in D.

CS so that $\mathcal{S}_u$ cannot distinguish whether the values are generated from real data. Then, an l-dimensional matrix is generated by comparing dummy data with a set of generated comparative values from CS. Since the participant's entity data does not leave the local, the generated matrix is derived from random numbers. Hence, the simulated l-dimensional matrix is indistinguishable from the one that is using true data for comparison. □

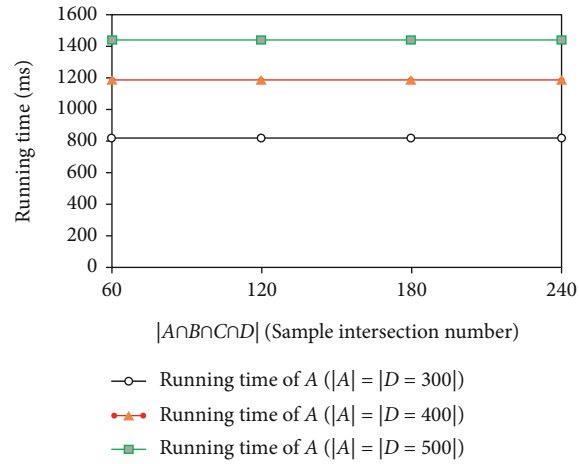

Figure 8: Running time of A with different amounts of intersection.

Case 2 (including CS in W). Namely, consider the corrupted CS and U_i.

Proof. For the corrupted CS and $\mathcal{S}_u$, denote the views of CS and participants U_i as $V_{CS} = \text{view}_{CS}$ and $V_u = \{\text{view}_{u_1}, \text{vie w}_{u_2}, \cdots, \text{view}_{u_n}\}$. Based on the process of the MEMP, we can derive $\text{view}_{CS} = \{v_{ik}, c_{ik}, d_{ij}\}$ and $\text{view}_{u_i} = \{v_{ik}, c_{ik}, g_{ik}\}$, where $i \in [1, n]$, $j \in [1, l]$ and ik refers to the kth entity's identity of the ith participants. It can be found that the elements belonging to view_{CS} and view_{u_i} can be treated as random values. Therefore, we can infer that view_{CS} and view_{u_i} are simulatable for $\mathcal{S}_{CS}$ and $\mathcal{S}_u$, and the simulated views cannot be distinguished computationally by the adversary.

The proof of the second part about $\mathcal{F}_{em2}$ is similar to Theorem 2, so we will not go into details.

5.3. Security of VFLR. In the model training, CS needs to get messages sent by at least t participants to decrypt the correct value so that the data for each participant cannot be retrieved and messages retrieved by CS can only be aggregated values without revealing anything else due to the combination of homomorphic encryption and threshold methods. The following theorem will be proved to show the security of the VFLR model.

Theorem 4. *For secure VFLR model, there exists a PPT simulator $\mathcal{S}_u$ or $\mathcal{S}_{cs}$ that can simulate the ideal view IDEAL$[\mathcal{I}, \mathcal{S}, \mathcal{F}_{ml}]$ which is computationally indistinguishable from the real view of $\mathcal{A}_u$ or $\mathcal{A}_{cs}$.*

Our VFLR calls the previous aggregation framework, and its security is based on the proof of Theorem 2. Since Theorem 2 has been proved, here we only do a simple description for VFLR.

Proof. Similar to the proof of Theorem 2, for a group of corrupted participants, $\mathcal{S}_u$ can run them using their local data while for honest participants $\mathcal{S}_u$ simulate them with dummy data. Therefore, $\mathcal{S}_u$ run $\mathcal{F}_{ml}$ to generate random values to

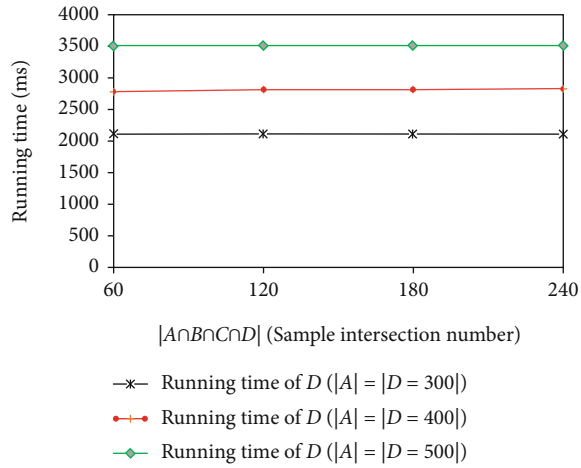

Figure 9: Running time of D with different amounts of intersection.

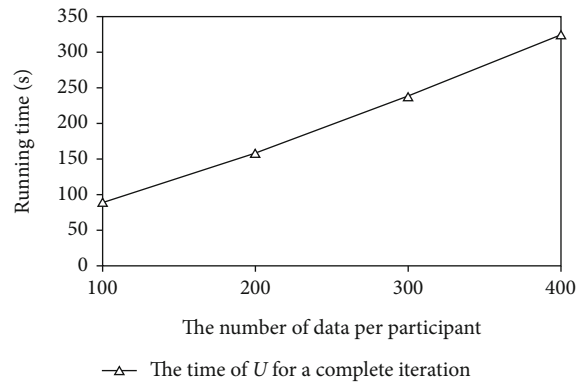

Figure 10: Running time of U_i with different amounts of data.

replace the masked $\Theta_i X_k^i$ as the inputs of honest participants. In the model building, what the honest-but-curious participants get is the errors rather than some information that reflects real data and they cannot identify whether the aggregated values used to calculate the errors are based on real data. Therefore, the view of $\mathcal{S}_u$ is indistinguishable from a real one. □

Considering the corrupted CS, $\mathcal{S}_{CS}$ run $\mathcal{F}_{ml}$ to generate dummy labels, with which $\mathcal{F}_{ml}$ computes the errors after obtaining the aggregation. Because local data are within honest participants, the outputs cannot reveal specific information. Therefore, there exist a PPT simulator $\mathcal{S}_{cs}$ that can simulate the ideal view which is computationally indistinguishable from the real view of $\mathcal{A}_{cs}$.

6. Performance Evaluation

In this section, the effectiveness and efficiency of the experiment are presented.

6.1. Experiment Configuration. Four clients A, B, C, and D are built to simulate the feasibility and performance, in which A, B, and C refer to the average participants and D means the aggregator with sample labels. We carry out our experiments

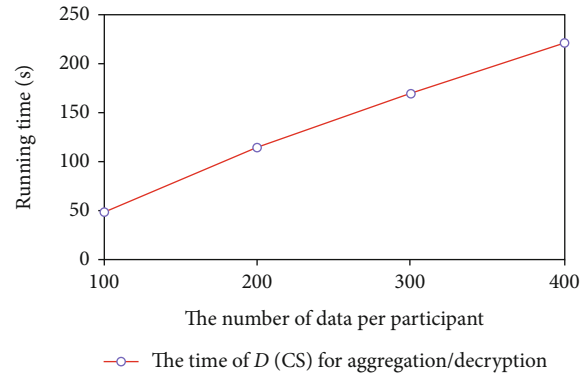

Figure 11: Running time of CS with different amounts of data.

Table 4: Comparison of functionality.

Scheme	GAM	M-p	EM	WP
Cheng et al.'s scheme [30]	×	√	×	×
Fu et al.'s scheme [31]	√	√	×	×
Our scheme	√	√	√	√

on the device with CPU i7-6700, 3.2 GHz, and Memory 24 G. The programs in the experiments are implemented in Python, and the length of p and q are set to 512 bits for RSA and Paillier. We adopt a finite field $\mathbb{Z}_P^*$ with $P = 2^{11}$ and the standard Shamir's $(t - n)$ secret sharing to generate the shares of secret.

In the MEMP, we generated different random values between 52252800000000000000 and 522528000000000000550 as identification of user samples for A, B, C, and D, respectively, satisfying that the number of intersection of A, B, C, and D is 60, 120, 180, and 240. In VFLR, we selected 300 digits from the handwritten digits dataset (handwritten digit dataset: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html), which contains 64 features. A, B, and C, respectively, hold 20 features in these samples, while D contains 4 features and 1 label. Since the dataset is multiclassified, we set 0 to 4 as category 1 and 5 to 9 as category 2.

6.2. Performance Analysis of the MEMP. For the MEMP, in the simulations, we get the execution time of U_i (A, B, C) and the CS (D) when increasing the amount of data from 100 to 500. Note that the time to initialize the system ignored in all experiments and $|A|$ represents the size of samples in A, the same thing for B, C, D. U_i (A, B, C) first calculates the median value for comparison through CS, the computation of which is related to the amount of their own data, and then the size of the calculated comparison matrix is related to the amount of CS's data. When the number of samples' identification of A, B, and C increases, the increase of computation time of A, B, or C is mainly reflected in generating sample identification library, and the most time-consuming in D is the blind signature. Figure 5 shows the calculation time of A and D when the number of samples in D is 100 and the samples in U_i (A, B, C) fluctuate from 100 to 500. Figures 6 and 7 show the changing trend of running time of A and D

Table 5: Comparison of computation and communication.

Scheme	Participant	Aggregator	Comm-rounds	Mask number
Xu et al.'s scheme [25]	$1R + 2SC$	tSC	2	$n \cdot t + 1$
Fu et al.'s scheme [31]	2SC + 2CR	1CR	1	n
Our scheme	$1R + 1HE + 1P$	$1HE + 1P$	1	n

with the increase of $|D|$ when the amounts of A, B, and C are all 100. As the sample amount of D is increasing, the encryption time is proportional to the data volume of D. Because the time of aggregation and decryption is related to its own data volume, its identification volume affects the size of an identification matrix, thereby affecting the aggregation volume, resulting in a linear increase in the time of aggregation and decryption. However, changes in the intersection of A, B, C, and D will not affect the respective calculation time that is only related to the amount of data of all parties, as is shown in Figures 8 and 9.

6.3. Performance Analysis of the VFLR. In the VFLR, there is no approximation algorithm applied here so that the updated parameters in our VFLR are exactly the same as those in the traditional logistic regression. Therefore, the training accuracy is also consistent. The main observation here is the running time of the VFLR. The extra cost in training is mainly from power exponent and homomorphism. In the training phase, t_e represents the time of a complete encryption and t_s represents the time of an aggregation and decryption for n paticipants. When the sample size is m, complexity time of encryption and aggregation, respectively, is $o(t_e \cdot m)$ and $o(t_s \cdot m)$. We selected 400 samples and set the number of features at each end to 20. When the number of data amount is 100, 200, 300, and 400, respectively, the time for a single end to complete an iteration and the time for D to complete aggregation and decryption are captured. (See Figures 10 and 11.) As the figures show, we gradually adjust the data volume from 100 to 400, and the running time increases approximately linearly.

6.4. Communication Overhead. Since the establishment of the user identification library and the generation of the comparison matrix can be done offline in the MEMP, we discuss the communication overhead of the MEMP from the perspective of the proposed aggregation model. The same is true for the VFLR, because its execution process is completely consistent with the GAM. Let the length of the ciphertext of each participant be cl and the length of the $H()$ and error, respectively, be $|H|$ and $|\text{error}|$ where $cl > |H|$ and $cl > |\text{error}|$. In the MEMP, it only takes one turn to complete the comparison and identify the common entity. If CS sends l entity mask, the communication load for each participant is $l \cdot cl$. In order for participants to complete the comparison, the CS needs to send corresponding entity mask for n participants so that the communication load of CS is $|H| \cdot l \cdot n + n \cdot |\text{ID}|$. In the VFLR, we just consider the communication load for one iteration. Each participant sends ciphertext of size $cl \cdot m$. The CS just needs to return errors of m samples, so the communication load of CS is $m \cdot |\text{error}|$. In this way, the total space complexity of MEMP and VFLR can be expressed as $o(n \cdot l \cdot cl)$ and $o(n \cdot m \cdot cl)$. We can see that as the number of participants or samples increases, the total communication overhead also grows linearly.

7. Related Work

Many privacy-preserving models for specific machine learning algorithms have emerged. There mainly exist two kinds of technologies adopted in the privacy-preserving training, i.e., differential privacy [21] and cryptography-based approaches. Differential privacy applied to FL can prevent clients from trying to reconstruct the private data of other clients by exploiting the global model, as done in [13, 22]. It adds noise to the original dataset or gradient parameters while ensuring the availability of the data. But it brings low accuracy. The cryptographic technologies can provide privacy protection while ensuring accuracy. Secure multiparty computation, secret sharing, and homomorphic encryption are the common methods. For example, Aono et al. [23] used homomorphic encryption to improve the logistic regression algorithm ensuring the security of the training and predicting data. Liu et al. [24] propose a secret sharing-based federated extreme boosting learning framework to achieve privacy-preserving model training for mobile crowdsensing. Xu et al. [25] proposed a privacy-preserving and verifiable federated learning framework based on homomorphic hash functions, in which clients can verify whether the result returned by cloud server is correct.

Some previous works with privacy preserving over vertical data partition are discussed in [26, 27]. However, there exist potential privacy risks as a result of revealing class distribution over the given attributes. Research on VFL is first proposed in [28], where a federated logistic regression scheme is designed through an additively homomorphic scheme. Nock et al. [29] then provide a formal assessment of how errors in entity resolution impact learning. Cheng et al. [30] propose a novel lossless privacy-preserving tree-boosting system, which conducts a learning process on multiple parties with partially common user samples but different feature sets. Fu et al. [31] combines Lagrange interpolation and Chinese remainder theorem to realize the secure aggregation of gradients. But some works assume sample entities already being matched or they only deal with two VFL participants. The proposed framework is more advantages than those approaches as it can support multiparticipant VFL by taking into account entity matching and model training with withdrawal of participants. Table 4 shows the functional comparison between our framework and existing two main works from GAM, multiparticipants (M-p), entity matching (EM), withdrawal of participants

(WP). Table 5 shows the comparison of computation and communication mainly for secure aggregation from the participant and aggregator's main function operation, communication rounds (Comm-rounds), and mask number (i.e., the number of message received by the aggregator), where SC represents the times of secret reconstruction, CR represents the times of calculation of Chinese residual theorem, HE represents the times of homomorphic encryption, R represents the times of pseudorandom generator, and P represents the times of calculation of the power exponent. It can be seen from Table 5 that our scheme has advantages over reference [25] in terms of calculation and communication overhead. Compared with [25, 31], although our scheme applies the principle of secret sharing, it does not need to spend the overhead of secret reconstruction.

8. Conclusion

For privacy protection of data aggregation and joint training in federated learning, as well as entity matching, we designed a PFLF where we proposed a general aggregation model and designed a multiparty entity matching protocol, which can find the common entity of multiple participants without data disclosure. In addition, our GAM was used to improve logistic regression algorithm to ensure the confidentiality of data samples during training and support the withdrawal of participants over VFL. The security analysis of the scheme was given based on the simulator, and the performance of the system was tested with the experimental data. The next research will focus on optimizing the operating load of the system and considering cases where the participants are malicious to construct a verifiable federated learning framework and design incentives to facilitate federated learning.

Data Availability

The data used to support the finding of this study are included in the article.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61662009 and 61772008), Science and Technology Major Support Program of Guizhou Province (Grant No. 20183001), Key Program of the National Natural Science Union Foundation of China (Grant No. U1836205), Science and Technology Program of Guizhou Province (Grant No. [2019]1098), Project of Innovative Group in Guizhou Education Department (Grant No. [2013]09), Project of High-level Innovative Talents of Guizhou Province (Grant No. [2020]6008), and Science and Technology Program of Guiyang (Grant No. [2021]1-5).

References

[1] D. Silver, A. Huang, C. J. Maddison et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] P. Kairouz, H. B. McMahan, B. Avent et al., "Advances and open problems in federated learning," 2019, https://arxiv.org/abs/1912.04977.

[3] Y. Tian, Z. Wang, J. Xiong, and J. Ma, "A blockchain-based secure key management scheme with trustworthiness in dwsns," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6193–6202, 2020.

[4] P. Voigt and A. Von dem Bussche, "The EU general data protection regulation (GDPR)," in *A Practical Guide*, Springer International Publishing, Cham, 1st edition, 2017.

[5] J. Konecny, H. B. McMahan, D. Ramage, and P. Richtarik, "Federated optimization: distributed machine learning for on-device intelligence," 2016, https://arxiv.org/abs/1610.02527.

[6] H. B. McMahan, E. Moore, D. Ramage, and B. A. Y. Arcas, *Federated Learning of Deep Networks Using Model Averaging*, 2016.

[7] K. Bonawitz, V. Ivanov, B. Kreuter et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, Dallas, TX, USA, October 2017.

[8] R. Xiong, R. Ma, L. Chen et al., "A personalized privacy protection framework for mobile crowdsensing in iiot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4231–4241, 2019.

[9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.

[10] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in Neural Information Processing Systems*, pp. 4424–4434, 2017.

[11] Y. Liu, Z. Ma, X. Liu, Z. Wang, S. Ma, and K. Ren, "Revocable federated learning: a benchmark of federated forest," 2019, https://arxiv.org/abs/1911.03242.

[12] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.

[13] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," 2017, https://arxiv.org/abs/1710.06963.

[14] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[15] G. Liang and S. S. Chawathe, "Privacy-preserving inter-database operations," in *International Conference on Intelligence and Security Informatics*, pp. 66–82, Springer, 2004.

[16] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," *Advances in neural information processing systems*, pp. 289–296, 2009.

[17] C. Jost, H. Lam, A. Maximov, and B. J. Smeets, "Encryption performance improvements of the paillier cryptosystem," *IACR Cryptology ePrint Archive*, vol. 2015, article 864, 2015.

[18] D. Bogdanov, *Foundations and Properties of Shamirs Secret Sharing Scheme Research Seminar in Cryptography*, vol. 1, University of Tartu, Institute of Computer Science, 2007.

[19] P. Mohassel and Y. Zhang, "Secureml: a system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, San Jose, CA, USA, May 2017.

[20] R. Canetti, Y. Dodis, R. Pass, and S. Walfish, "Universally composable security with global setup," in *Theory of Cryptography Conference*, pp. 61–85, Springer, 2007.

[21] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: the sulq framework," in *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '05*, pp. 128–138, Baltimore, Maryland, USA, 2005.

[22] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," *International Conference on Artificial Intelligence and Statistics*, pp. 473–481, 2018.

[23] Y. Aono, T. Hayashi, L. Trieu Phong, and L. Wang, "Scalable and secure logistic regression via homomorphic encryption," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pp. 142–144, New Orleans, LA, USA, March 2016.

[24] Y. Liu, Z. Ma, X. Liu, S. Ma, S. Nepal, and R. Deng, "Boosting privately: privacy-preserving federated extreme boosting for mobile crowdsensing," 2019, https://arxiv.org/abs/1907.10218.

[25] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.

[26] J. Vaidya and C. Clifton, "Privacy-preserving decision trees over vertically partitioned datan," in *IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 139–152, Springer, 2005.

[27] J. Vaidya, "A survey of privacy-preserving methods across vertically partitioned data," in *Privacy-preserving data mining*, pp. 337–358, Springer, 2008.

[28] S. Hardy, W. Henecka, H. Ivey-Law et al., "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017, https://arxiv.org/abs/1711.10677.

[29] R. Nock, S. Hardy, W. Henecka et al., "Entity resolution and federated learning get a federated resolution," 2018, https://arxiv.org/abs/1803.04035.

[30] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "Secureboost: a lossless federated learning framework," 2019, https://arxiv.org/abs/1901.08755.

[31] A. Fu, X. Zhang, N. Xiong, Y. Gao, H. Wang, and J. Zhang, "VFL: a verifiable federated learning with privacy-preserving for big data in industrial IoT," *IEEE Transactions on Industrial Informatics*, p. 1, 2020.