CFL: Cluster Federated Learning in Large-scale Peer-to-Peer Networks

Qian Chen1, Zilong Wang1, Yilin Zhou1, Jiawei Chen1, Dan Xiao1, and Xiaodong Lin2

1State Key Laboratory of Integrated Service Networks, School of Cyber Engineering, Xidian University, Xi’an 710071, China
{qchen_4, jiawei.chen98}@stu.xidian.edu.cn, zlwang@xidian.edu.cn,
{xduyilinzhou, xdudanxiao}@gmail.com
2School of Computer Science, University of Guelph, Guelph, Canada
xlin08@uoguelph.ca

Abstract. Federated learning (FL) has sparked extensive interest in exploiting the private data on clients’ local devices. However, the parameter server setting of FL not only has high bandwidth requirements, but also poses data privacy issues and a single point of failure. In this paper, we propose an efficient and privacy-preserving protocol, dubbed CFL, which is the first fine-grained global model training for FL in large-scale peer-to-peer (P2P) networks. Unlike previous FL in P2P networks, CFL aggregates local model update parameters hierarchically, which improves the communication efficiency facing large amounts of clients. Also, the aggregation in CFL is performed in a secure manner by introducing the authenticated encryption scheme, whose key is established through a random pairwise key scheme enhanced by a proposed voting-based key revocation mechanism. Rigorous analyses show that CFL guarantees the privacy and data integrity and authenticity of local model update parameters under two widespread threat models. More importantly, the proposed key revocation mechanism can effectively resist hijack attacks, thereby ensuring the confidentiality of the communication keys. Ingenious experiments on the Trec06p and Trec07 datasets show that the global model trained by CFL has good classification accuracy, model generalization, and rapid convergence rate, and the dropout-robustness of the system is achieved. Compared to the first global model training protocol for FL in P2P networks, PPT, CFL improves communication efficiency by 43.25%. Also, CFL outperforms PPT in terms of computational efficiency.

Keywords: Federated learning · Peer-to-peer network · Communication efficiency · Privacy-preserving

1 Introduction

Machine learning (ML) has injected vitality into many aspects of modern society, such as speech recognition [10], image recognition [33], natural language
processing [17], etc. Traditional ML is featured by centrally collecting raw data to train the model. However, gathering all data to a central database consumes high communication bandwidth, while the bandwidth is a rare resource. Besides, the centrally collected data has a high risk to be abused, which has drawn more and more attention from society, e.g., the General Data Protection Regulation [30] where the usage of user data is strictly limited. Under such circumstances, ML seems to meet a severe data usage dilemma [34].

To address communication efficiency and privacy concerns that centralized ML poses to data owners, Federated learning (FL) [24] has emerged as a state-of-the-art ML system to solve the dilemma, which allows clients to collaboratively reap the benefits of shared models without centrally storing data. In FL, clients train their personalized models locally, and then a central server aggregates local contributions, e.g., local models, local gradients, or local model update parameters, to update the global model. Such a distributed paradigm not only preserves the privacy of clients [5] but also improves the communication efficiency of the system [18], as only the local contributions leave clients’ local devices instead of the private and massive local data.

Although FL has been a great success, the parameter server setting relies too much on the central server which might not exist in the real world, such as Internet of Things (IoT) [20], Smart Home [28], and Ad Hoc [26]. Besides, the central server gives rise to several drawbacks: (1) the central server must be honest, which is difficult to guarantee in the real world [14]; (2) the central server has high computational costs and high bandwidth requirements [21]; (3) the central server can become a single point of failure [13,15]. As a result, how to deploy FL without the central server deserves deep research, which is referred to as the decentralized FL [15] or FL in peer-to-peer (P2P) networks [9].

Consequently, some decentralized FL works involving the protocol [9], algorithms [1,13,31], and frameworks [11,16,19,25] have been proposed successively. Also, decentralized vertical FL, which is a special decentralized FL paradigm, has been studied in [14,22,32]. However, all existing decentralized FL works are coarse-grained, which cannot guide the global model training in practice, except [9] (Under review by Computers & Security after a major reversion). Recently, Chen et al. [9] proposed a fine-grained PPT protocol, which was claimed as the first communication-efficient and privacy-preserving global model training protocol for FL in P2P networks. However, PPT has several shortcomings facing massive clients. First, PPT aggregates local model update parameters directly by a so-called cyclic transmission manner, which is inefficient for FL in large-scale P2P networks where massive clients are involved. Second, the key distribution scheme in PPT is vulnerable to hijacking attacks [10], which leaves a potential privacy leakage risk to the system. Third, the signature scheme in PPT requires extra storage space and heavy computational power. Therefore, an intuitive question is How to efficiently and securely achieves FL in large-scale P2P networks.

Our response to this question is a Cluster Federated Learning (CFL) global model training protocol for FL in large-scale P2P networks. Specifically, we
expand the system model of PPT [9], where clients and a server loosely connected to a few of them are distributed in a P2P network, by involving large amounts of clients. To improve the communication efficiency of the system, CFL aggregates local model update parameters hierarchically, rather than directly aggregates all local contributions like PPT. Instead of leveraging the digital signature scheme, CFL guarantees security and computation efficiency through the authenticated encryption scheme, whose key is established by a random pairwise keys scheme enhanced by a proposed key revocation mechanism which improves the security against hijacking attacks [16].

The main contributions are as follows:

A global model training protocol for FL in large-scale P2P networks. We propose a fine-grained global model training protocol for FL in large-scale P2P networks which comprises a Secure Communication Key Establishment Protocol and an Inner-Cluster Model Aggregation Protocol. Compared to the PPT protocol, larger amounts of clients could train the global model in a higher communication and computation efficient manner with the deployment of the proposed CFL protocol.

A secure and privacy-preserving protocol. We analyze the security and privacy ability CFL achieves under the widespread internal semi-honest participants and external malicious adversaries threat models [4,5,9]. In detail, the proposed CFL protocol can protect the privacy of the client’s individual local contributions by a random noise, which is generated initially and eliminated ultimately. Besides, CFL guarantees the data integrity and authenticity of local contributions by leveraging the authenticated encryption scheme. Particularly, rigorous analyses show that CFL can guarantee the confidentiality of the communication key against hijacking attacks [16]. In addition, we analyze the network connectivity and formally give the threshold of the key ring size, which proves that the proposed CFL protocol can guarantee secure transmission.

Experimental evaluation. We conduct experiments on the Trec06p and Trec07 datasets, which demonstrate that the proposed CFL protocol can ensure classification accuracy, model generalization, and rapid convergence rate of the global model. Also, the dropout-robustness of the system is achieved by CFL. More importantly, CFL improves communication efficiency by 43.25% and computational efficiency by 0.87% compared to PPT [9].

The rest of the paper is organized as follows. Section 2 reviews related works. In Section 3, we formalize the system model, security requirements, and design goal. Section 4 formally describes some preliminaries. Section 5 devises the proposed CFL protocol followed by security analysis, network connectivity analysis, and experiments in Sections 6 and 7. We conclude this paper in Section 8.

2 Related Works

Existing decentralized FL works mainly focus on algorithms and frameworks. As for the algorithm, Baidu [13] proposed a bandwidth optimization algorithm, Ring AllReduce, which averages the gradient effectively on distributed devices. Yang
et al. [31] exploited the signal superposition property of wireless multiple-access channels to locally computing updates in FL by the over-the-air computation. Combining the PushSum gossip algorithm with stochastic gradient updates, As-sran et al. [1] proposed SGP and OSGP to accelerate distributed training of deep neural networks. As for the framework, Ramanan et al. [25] proposed a blockchain-based aggregator-free FL framework, BAFFLE, which achieves high scalability and computational efficiency in a private Ethereum network. Hu et al. [15] addressed the decentralized FL problem by aggregating the local model segments based on the gossip protocol. Dubey et al. [11] devised FEDUCB for FL in P2P networks to solve the contextual linear bandit problem. Li et al. [19] proposed SimFL to build decision trees with bounded errors, which leverages locality-sensitive hashing to collect similarity information. However, the above works propose advanced decentralized FL frameworks or algorithms in a coarse-grained manner. A formal description of decentralized FL settings and a general and fine-grained model training protocol that can guide model training for FL in P2P networks in practice are still absent.

Recently, Chen et al. [9] formalized the FL settings in P2P networks and proposed a fine-grained model training protocol, PPT, which aggregates clients’ local contributions in a privacy-preserving manner. However, the PPT protocol is inefficient facing a large number of clients, i.e., FL in large-scale P2P networks. Besides, the communication key establishment phase in PPT is vulnerable to hijacking attacks [16], which might give rise to a security risk to the system.

To meet the efficiency requirement of FL in large-scale P2P networks, the proposed CFL hierarchically aggregates local contributions through the multi cyclic transmission. Also, the proposed CFL resists hijacking attacks by a proposed voting-based key revocation mechanism when establishing communication keys. To the best of our knowledge, the proposed CFL protocol is the first fine-grained model training protocol for FL in large-scale P2P networks, which is efficient and privacy-preserving.

3 System Model, Security Requirements and Design Goal

In this section, we formalize the system model, propose the security requirements, and identify the design goal.

3.1 System Model

In this paper, we consider the FL in large-scale P2P networks, which is an expansion of the system model of PPT [9] in terms of client amount. Specifically, as shown in Fig. 1, a large group of potential clients $U = \{u_i | i = 1, 2, \cdots, Z\}$ with constant wireless communication ranges are distributed in a large-scale P2P network. The clients truly participate in the global model training process are target clients, denoted by $C = \{c_i | i = 1, 2, \cdots, N\}, C \subseteq U$. Each target client $c_i$ has a local dataset $D_i$ of size $|D_i|$ containing private training data. Besides, a server loosely connected with a few potential clients is responsible for coordinating
FL tasks, which means the server cannot aggregate clients’ local contributions directly. The goal of target clients is to collaboratively train a global ML model $W$ under the orchestration of the server, where $W$ is usually expressed as a $d$-dimensional vector in the ML community.

In detail, the collaborative training process is an iteration. For clarity, we take the $t$-th round as an example to describe the iterative training process. The server first sends the global model $W^t$ to the potential clients connected to it directly, and these potential clients pass $W^t$ to other potential clients subsequently. After receiving the global model, each target client $c_i$ initializes the local model as the global model $W^t$, and trains it on $\mathcal{D}_i$ to obtain an updated local model $w^t_i$ through the stochastic gradient descent (SGD) algorithm, which is simply expressed as:

$$w^t_i \leftarrow \text{Train}(W^t, \mathcal{D}_i),$$

(1)

where $w^t_i$ is a $d$-dimensional vector. Then, each target client $c_i$ calculates its local model update parameters $x^t_i$, shown as:

$$x^t_i = w^t_i - W^t.$$  

(2)

Next, all target clients aggregate local model update parameters and upload the aggregation result to the server to update the global model, shown as:

$$W^{t+1} = W^t + \sum_{i=1}^{N} p_i x^t_i,$$

(3)

where $p_i = \frac{|\mathcal{D}_i|}{\sum_{i=1}^{N} |\mathcal{D}_i|}$ is the aggregation weight of $c_i$, and $W^{t+1}$ is the updated global model. All participants under our consideration perform the above training process periodically until the global model converges to an optimal result, i.e., the final converged global model $W^*$.  

Communication model. Inherited from the communication characteristics of P2P networks, all data in the system is transmitted through public communication channels, which means the transmitted data could be eavesdropped by any
client. In addition, all target clients can only communicate with their single-hop neighbor clients within their constant wireless communication ranges, and only a few clients could communicate with the server directly.

3.2 Security Requirements

In the context of FL in large-scale P2P networks, security and privacy are the most important concerns. Thus, we consider two widely used threat models in FL [13,19], i.e., the internal semi-honest participants threat model and the external malicious adversaries threat model. We further consider hijacking attacks [16] from external malicious adversaries when establishing communication keys, which is thoughtless in PPT [9].

In the internal semi-honest (also named as honest-but-curious) participants threat model, the server and target clients perform prescribed operations honestly but are curious about others’ local model update parameters, which means clients could infer private information from others’ local model update parameters by executing model inversion attacks [12]. In particular, honest-but-curious potential clients may execute eavesdropping attacks [27] to obtain the private local model update parameters from target clients, thereby executing model inversion attacks [12] to threaten the privacy of local data.

As shown in Fig. 1, in the external malicious adversaries threat model, a malicious adversary $A$ could execute tampering attacks [27] and impersonation attacks [27] to threaten the data integrity and the authenticity of aggregated local contributions. More seriously, $A$ could execute hijacking attacks, where $A$ hijacks honest participants to obtain the communication keys, and hence tampers with the aggregated model update parameters when aggregating local contributions, resulting in serious security risks. As a result, the proposed protocol should meet the following security requirements:

- **Privacy preservation.** Neither an honest-but-curious server nor clients can obtain others’ precise local model update parameters. Also, the aggregated model update parameters should be only obtained by the sender and receiver but others during the transmission process.

- **Data integrity and authenticity.** Ensure that local model update parameters are indeed sent by a legitimate client and have not been tampered with during transmission. That is, active attacks executed by $A$ should be resisted.

- **Confidentiality of the communication keys.** The communication key used to protect the aggregated model update parameters during transmission can only be obtained by the two corresponding target clients. In other words, hijacking attacks executed by $A$ should be detected and resisted.

3.3 Design Goal

Under the system model and security requirements, our design goal is to propose an efficient and privacy-preserving global model training protocol for FL in large-scale P2P networks. Specifically, the following objectives should be achieved in the proposed protocol:
The security requirements should be guaranteed in the proposed protocol. As mentioned in the security requirements, the proposed protocol should achieve privacy preservation, data integrity and authenticity, and the communication key confidentiality simultaneously under the internal semi-honest participants threat model and the external malicious adversaries threat model.

Communication and computational efficiency should be guaranteed in the proposed protocol. Although more clients are involved in FL in large-scale P2P networks, the proposed protocol should ensure communication efficiency. In addition, the proposed protocol should reduce the computational cost of a single client to ensure the computational efficiency.

4 Preliminaries

4.1 Random Pairwise Keys Scheme

The random pairwise keys scheme [8] is initially deployed in distributed networks to establish communication keys. The basic idea is to associate two potential clients with a pairwise key based on their identities. The detailed steps of the random pairwise keys scheme are described as follows:

- **Step-1.** All potential clients generate their unique identities, denoted by $ID = \{ID_i| i=1, 2, \cdots, Z\}$.
- **Step-2.** Each potential client randomly matches its identity with $M$ other identities, and each client pair obtains a pair-wise $k$ from a key pool $K$.
- **Step-3.** The pair-wise $k$ and the identity of the matching client are stored in both clients’ key rings, denoted by $R_i = \{(ID_{i\alpha}, k_{i\alpha})|\alpha=1, 2, \cdots, M\}$.
- **Step-4.** Through the challenge-response mechanism, each potential client uses its key ring to establish communication keys with other potential clients that have common keys.

4.2 Authenticated Encryption

Authenticated encryption [2] is an encryption scheme that provides both security and authentication, which comprises three algorithms: the key generation algorithm $AE.Gen(\cdot)$, the encryption algorithm $AE.Enc(\cdot)$, and the decryption algorithm $AE.Dec(\cdot)$. The detailed steps of the authenticated encryption are as follows:

1. $AE.Gen(\cdot)$ generates a symmetric key $K$, which is simply expressed as:
   
   $$K \leftarrow AE.Gen(1^\kappa),$$

   where $\kappa$ is a security parameter, i.e. the length of the symmetric key which is consistent with the length of each dimension of the local model.

2. $AE.Enc(\cdot)$ inputs the plaintext $X$ and the key $K$ and outputs the ciphertext $Y$ and an authentication tag $\sigma$ in the form of a message authentication code (MAC), which is simply expressed as:

   $$(Y||\sigma) \leftarrow AE.Enc(X, K),$$
where $||$ represents the string concatenation operator.

$\text{AE.Dec}(\cdot)$ inputs the ciphertext $Y$ and the corresponding key $K$ and outputs the plaintext $X$, which is simply expressed as:

$$X \leftarrow \text{AE.Dec}((Y||σ), K).$$

If the $σ$ fails authentication, $\text{AE.Dec}(\cdot)$ outputs an unauthenticated symbol.

5 System Design

In this section, we propose an efficient and privacy-preserving Cluster Federated Learning (CFL) global model training protocol for FL in large-scale P2P networks. The proposed CFL protocol mainly consists of the following five parts: cluster division, communication key establishment, local model training, aggregation within a single cluster, and aggregation and update across clusters.

5.1 Cluster Division

There are large amounts of clients distributed in a large-scale P2P network under our consideration, where the complex connections among clients may bring huge communication latency if aggregating all local contributions directly. Therefore, we first divide all potential clients into several clusters $L = \{L_h | h = 1, 2, \cdots, λ\}$ based on their constant wireless communication ranges, shown in Fig. 1. Note that, since only a few potential clients are directly connected to the server in each cluster, the potential clients and their connections within the cluster can also be regarded as a P2P network. Thus, the local contribution can be aggregated hierarchically, i.e., aggregation within a single cluster and aggregation across clusters subsequently. We stress that such a hierarchical aggregation process is named as multi cyclic transmission, since the aggregation processes within several clusters are performed simultaneously, which is more communication-efficient than cyclic transmission of the PPT protocol [9], where all participants aggregate local contributions in a cyclic manner directly.

For gravity, we regularize the symbols in a specific cluster $L_h \in L$. The potential clients in $L_h$ are denoted by $U_h = \{u_{h,i} | i = 1, 2, \cdots, z_h\}$, and all $U_h (h = 1, 2, \cdots, λ)$ make up $U$, i.e., $U = \bigcup_{h=1}^{λ} U_h$. The clients that actually participate in a specific round of global model training are target clients, denoted by $C_h = \{c_{h,i} | i = 1, 2, \cdots, n_h\}$, $C_h \subseteq U_h$, and all $C_h (h = 1, 2, \cdots, λ)$ make up $C$, i.e., $C = \bigcup_{h=1}^{λ} C_h$.

5.2 Communication Key Establishment

To enhance the security while transmitting local model update parameters, we propose a Secure Communication Key Establishment Protocol in Fig. 2 to establish communication keys for potential clients within a single cluster,
Secure Communication Key Establishment Protocol

- **Communication key establishment:**
  - Each potential client \( u_{h,i} \) generates its identity \( ID_{h,i} \).
  - Each identity is matched with \( m_h \) other randomly selected identities.
  - Each client pair associated with two identities obtains a pairwise key from a key pool \( K \).
  - All identities and the corresponding keys make up \( u_{h,i} \)'s key ring
    \[ R_{h,i} = \{ (ID_{h,i,\alpha}, k_{h,i,\alpha}) | \alpha = 1, 2, \ldots, m_h \} \]
  - \( u_{h,i} \) broadcasts a plaintext \( a_{h,i} \) and \( m_h \) encrypted messages \( A_{h,i,\alpha} \)
    using \( k_{h,i,\alpha} \) in \( R_{h,i} \), shown as \( A_{h,i,\alpha} \leftarrow CK.Enc(k_{h,i,\alpha}, a_{h,i}) \).
  - \( u_{h,j} \), \( j \neq i \) tries to decrypt \( A_{h,i,\alpha} \) using \( k_{h,j,\alpha} \) in \( R_{h,j} \), which is shown as \( a'_{h,j} \leftarrow CK.Dec(k_{h,j,\alpha}, A_{h,i,\alpha}) \).
  - \( u_{h,i} \) compares \( a'_{h,j} \) with the plaintext \( a_{h,i} \), seeking for shared keys.
  - \( u_{h,i} \) and \( u_{h,j} \) establish the communication key \( K_{h_i,h_j} \) by XOR.

- **Key revocation:**
  - \( u_{h,i} \) is assigned \( m_h \) voting keys \( \{ v_{h,i,\alpha} | \alpha = 1, 2, \ldots, m_h \} \) and the hash values of the voting keys of \( m_h - 1 \) other voting members \( H(v_{h,i,\beta}) \).
    where \( H(\cdot) \) is the hash function, \( \beta \neq \alpha \), and \( 1 \leq \beta \leq m_h \).
  - When \( u_{h,\alpha}, u'_{\alpha} \) votes against \( A' \), it broadcasts \( v_{h,\alpha',\alpha} \) in forms of plaintext.
  - \( m_h - 1 \) other voting members verify the vote \( v_{h,\alpha',\alpha} \), shown as
    \[ \{0, 1\} \leftarrow Verify(H(v_{h,\alpha'},\alpha), H'(v_{h,\alpha',\alpha})). \]
  - Once validated, the vote is marked, shown as \( H(v_{h,\alpha',\alpha}) \leftarrow (v_{h,\alpha',\alpha}, \nu) \),
    where \( \nu \) denotes the vote is marked.
  - If the number of marked votes is not less than \( l_h \), \( A' \) will be revoked.

- **Re-keying:**
  - The affected clients restart the Communication key establishment part to build new communication keys.

Fig. 2: Secure Communication Key Establishment Protocol.

which is an important part of the proposed CFL protocol and comprises three parts, i.e., Communication key establishment, Key revocation, and Re-keying.

- **Communication key establishment.** Without loss of generality, we take the cluster \( L_h \) as an example. Each potential client \( u_{h,i} \) first generates a unique identity \( ID_{h,i} \). Then, \( u_{h,i} \) randomly matches its identity with \( m_h \) other identities, and each client pair obtains a pairwise key from a key pool \( K \) containing sufficient keys. Thus, \( m_h \) sets of identities and the corresponding pairwise keys make up the key ring
  \[ R_{h,i} = \{ (ID_{h,i,\alpha}, k_{h,i,\alpha}) | \alpha = 1, 2, \ldots, m_h \} \]
  which is stored locally on \( u_{h,i} \).
  Note that the above offline operations do not consume communication resources.

  Afterwards, all potential clients discover other potential clients that have shared keys with them through the challenge-response mechanism. Specifically, \( u_{h,i} \) broadcasts a plaintext \( a_{h,i} \) and \( m_h \) messages \( \{ A_{h,i,\alpha} | \alpha = 1, 2, \ldots, m_h \} \) encrypted by \( m_h \) pairwise keys \( \{ k_{h,i,\alpha} | \alpha = 1, 2, \ldots, m_h \} \) in \( R_{h,i} \), shown as:

  \[ A_{h,i,\alpha} \leftarrow CK.Enc(k_{h,i,\alpha}, a_{h,i}), \quad (4) \]
where $\text{CK.Enc}(\cdot)$ is the symmetric encryption algorithm. After that, every other potential clients $u_{h,j}$ ($j \neq i$) attempts to decrypt the ciphertexts successively using the locally stored $m_h$ pairwise keys in its keyring, shown as:

$$a'_{h,i} \leftarrow \text{CK.Dec}(k_{h,j\alpha}, A_{h,i\alpha}),$$

(5)

where $a'_{h,i}$ is the decryption result. Consequently, $u_{h,j}$ compares $a'_{h,i}$ with the plaintext $a_{h,i}$, seeking for shared keys. Thus, $u_{h,i}$ and $u_{h,j}$ execute XOR operations for all shared keys to obtain a communication key $K_{h,i,h,j}$.

- **Key revocation.** In the external malicious threat model under our consideration, an adversary $A$ can hijack honest potential clients to obtain the communication keys, thereby threatening the security of the system. Thus, we design a compact and efficient key revocation mechanism based on public voting. In detail, once a potential client observes active attacks from the client $A'$ hijacked by $A$, it broadcasts a negative vote. We define the voting members as $\{u_{h,A'\alpha} | \alpha = 1, 2, \ldots, m_h\}$ who are single-hop neighbors of $A'$. Once $A'$ is voted against more than $l_h$, all voting members would cut off all connections to $A'$ for revocation, thereby eliminating the $A'$'s influence on the system.

In particular, the neighborhood broadcast [3] mechanism is used for every public vote in the network. The mechanism adopts broadcasts between neighbors to hierarchically make a consensus in the network. We stipulate that all voting members need to broadcast any public votes received to maximize the probability of successful transmission to adjacent voting members. All broadcasts are in the form of plaintext, as the voters' identities have no need to be protected. The detailed voting process is demonstrated in Fig. 2.

- **Re-keying.** To ensure the long-term security of the system, the communication keys used for a period of time need to be updated. When the keys expire, the Re-keying part will start. That is, all affected clients restart the Communication key establishment part to rebuild new communication keys.

### 5.3 Local Model Training

After establishing communication keys, all potential clients collaboratively train the global model. Taking the $t$-th round as an example, the server first distributes a global model $W^t$ to the potential clients directly connected to it. These potential clients subsequently pass $W^t$ to other potential clients within the same cluster. After receiving $W^t$, each target client $c_{h,i}$ initializes the local model as $W^t$, and trains it on its local dataset $D_{h,i}$, shown as:

$$w_{h,i}^t \leftarrow \text{Train}(W^t, D_{h,i}).$$

(6)

Then, $c_{h,i}$ calculates the local model update parameters, shown as:

$$x_{h,i}^t = w_{h,i}^t - W^t.$$  

(7)

Afterwards, $c_{h,i}$ calculates its weighted local model update parameters $X_{h,i}^t = p_{h,i} x_{h,i}^t$, where $p_{h,i} = \frac{|D_{h,i}|}{\sum_{i=1}^{\frac{|D_{h,i}|}{|D_{h,i}|}}}$. 

5.4 Aggregation within a Single Cluster

| Inner-Cluster Model Aggregation Protocol |
|------------------------------------------|

- **Local Model Training:**
  - \(c_{h,1}\) performs local model training and holds weighted local model update parameters \(X_{h,1}^t\) to be uploaded.
  - \(c_{h,1}\) broadcasts that it would participate in the global model update.

- **Model Disturbance:**
  - The server randomly chooses a leader client \(c_{h,1}\).
  - \(c_{h,1}\) generates a random noise \(s_h^t\) to disturb \(X_{h,1}^t\), shown as \(\hat{X}_{h,1}^t = X_{h,1}^t + s_h^t\).

- **Transmission and Aggregation:**
  - \(c_{h,1}\) chooses a neighbor \(c_{h,2}\) and broadcasts the choice.
  - \(c_{h,1}\) encrypts \(\hat{X}_{h,1}^t\), shown as \((Y_{h,1}^t||\sigma_{h,1}^t) \leftarrow \text{AE.Enc}\left((\hat{X}_{h,1}^t||\tau_{h,1}^t), K_{h1,h2}\right)\).
  - \(c_{h,1}\) sends \(Y_{h,1}^t||\sigma_{h,1}^t\) to \(c_{h,2}\).
  - \(c_{h,2}\) decrypts \(Y_{h,1}^t||\sigma_{h,1}^t\), shown as \((\hat{X}_{h,1}^t||\tau_{h,1}^t) \leftarrow \text{AE.Dec}\left((Y_{h,1}^t||\sigma_{h,1}^t), K_{h1,h2}\right)\).
  - \(c_{h,2}\) calculates \(X_{h,2}^t = \hat{X}_{h,2}^t + X_{h,2}^t\).
  - \(c_{h,2}\) performs the broadcast, encryption, and send operations as \(c_{h,1}\) does.
  - The remaining target clients successively perform the same operations as \(c_{h,2}\) does until \(c_{h,n_k}\) receives the aggregated result.
  - \(c_{h,n_k}\) aggregates its local contributions, shown as \(\hat{X}_{h,n_k}^t = \sum_{i=1}^{n_k} X_{h,i}^t + s_h^t\).

- **Cluster model Aggregation:**
  - \(c_{h,n_k}\) sends \(\hat{X}_{h,n_k}^t\) back to \(c_{h,1}\).
  - \(c_{h,1}\) subtracts \(s_h^t\) and calculates \(\text{sum}_h^t = \sum_{i=1}^{n_k} X_{h,i}^t\).

Fig. 3: Inner-Cluster Model Aggregation Protocol.

After local training, all target clients aggregate their local model update parameters within their clusters respectively. The server randomly selects a target client directly connected to it as the leader client, namely \(c_{h,1}\). Then, \(c_{h,1}\) generates a random noise \(s_h^t\) to disturb its weighted local model update parameters \(X_{h,1}^t\), shown as:

\[
\hat{X}_{h,1}^t = X_{h,1}^t + s_h^t,
\]

where \(s_h^t\) is a \(d\)-dimensional vector. Subsequently, \(c_{h,1}\) attaches a timestamp \(\tau_{h,1}^t\) to the disturbed model update parameters, which is represented as \(\hat{X}_{h,1}^t||\tau_{h,1}^t\).

Next, \(c_{h,1}\) randomly selects a neighbor client \(c_{h,2}\) and encrypts \(\hat{X}_{h,1}^t||\tau_{h,1}^t\) using their communication key \(K_{h1,h2}\), shown as:

\[
(Y_{h,1}^t||\sigma_{h,1}^t) \leftarrow \text{AE.Enc}\left((\hat{X}_{h,1}^t||\tau_{h,1}^t), K_{h1,h2}\right).
\]

Then, \(c_{h,1}\) sends \(Y_{h,1}^t||\sigma_{h,1}^t\) to \(c_{h,2}\), and \(c_{h,2}\) recovers it subsequently, shown as:

\[
(\hat{X}_{h,1}^t||\tau_{h,1}^t) \leftarrow \text{AE.Dec}\left((Y_{h,1}^t||\sigma_{h,1}^t), K_{h1,h2}\right).
\]
After validating the timestamp $\tau_{t,1}^h$, $c_{h,2}$ aggregates its own weighted model update parameters, shown as:

$$\hat{X}_{t,2}^h = \hat{X}_{t,1}^h + X_{t,2}^h.$$  \hfill (11)

Subsequently, $c_{h,2}$ selects a new neighbor client $c_{h,3}$, and performs the encryption and transmission operations as $c_{h,1}$ does. The remaining target clients perform the same decryption, aggregation, encryption, and transmission operations successively until all the target clients are traversed. Note that the aggregation route follows the depth-first traversal algorithm in the graph [29], where each target client prefers to select the target clients that have not yet participated in the aggregation. Finally, the final client $c_{h,n_h}$ sends the aggregated model update parameters back to $c_{h,1}$. $c_{h,1}$ subtracts the noise $s_t^h$ to obtain the aggregation result within its cluster, denoted by $\text{sum}_t^h$ and defined as:

$$\text{sum}_t^h = \sum_{i=1}^{n_h} X_{t,i}^h.$$ \hfill (12)

To make a consensus on the aggregation route, we also adopt the neighborhood broadcast [3] mechanism during transmission, where each target client hierarchically broadcasts its behaviors to all clients. With the deployment of the neighborhood broadcast mechanism, dropout-robustness is guaranteed.

For gravity, we illustrate the model aggregation process within a single cluster as the Inner-Cluster Model Aggregation Protocol in Fig. 3.

### 5.5 Aggregation and Update Across Clusters

After finishing the model aggregation within each single cluster, each leader client $c_{h,1}$ could obtain the inner-cluster aggregation result $\text{sum}_t^h$. Next, each leader client uploads $\text{sum}_t^h$ directly to the server, and the server aggregates all $\text{sum}_t^h, h = 1, \cdots, \lambda$, shown as:

$$\text{SUM}_t^t = \sum_{h=1}^\lambda q_h \text{sum}_t^h,$$ \hfill (13)

where $q_h = \frac{\sum_{D_i \in L_h} |D_i|}{\sum_{i=1}^{L} |D_i|}$ is the aggregation weight of the cluster $L_h$. Consequently, the server updates the global model, shown as:

$$W_{t+1} = W_t + \text{SUM}_t^t.$$ \hfill (14)

Finally, the server distributes the updated global model to all potential clients, and the target clients iteratively train and aggregate until the global model converges to $W^*$. For clarity of description, we illustrate the integrated CFL protocol from a high-level view in Fig. 4.
CFL for FL in Large-Scale P2P Networks (high-level view)

Step 1. All potential clients distributed in the P2P network are divided into several clusters according to their wireless communication range.

Step 2. The potential clients in each cluster establish communication keys according to the Secure Communication Key Establishment Protocol.

Step 3. All target clients in every cluster perform local model training.

Step 4. All target clients within each single cluster perform aggregation according to the Inner-Cluster Model Aggregation Protocol.

Step 5. All target clients perform aggregation across clusters to obtain the global model update parameters.

Step 6. The server updates the global model.

Step 7. The server distributes the updated global model to all potential clients.

Step 8. The server determines whether the global model converges:
   - If so, stop training; Otherwise, go back to step 3.

Note:
1. The neighborhood broadcast mechanism is adopted to record the behaviors of all clients.
2. A newly joined client $u_{\text{new}}$ will perform step 2-8.

Fig. 4: High-level view of the CFL protocol.

6 Analysis

In this section, we analyze the security properties of the proposed CFL protocol under the internal semi-honest participants threat model and the external malicious adversaries threat model. Particularly, we analyze the network connectivity, which is critical for secure communication among clients.

6.1 Security Analysis

In the internal semi-honest participants threat model, we mainly focus on the privacy requirements facing the honest-but-curious server, target clients, and potential clients.

The proposed CFL protocol is privacy-preserving facing an honest-but-curious server. An honest-but-curious server may infer the private local training data from clients’ local contributions by executing model inversion attacks [12]. To defend against such inferences, the proposed CFL protocol aggregates local contributions in a privacy-preserving manner, where the server can only obtain the aggregation result rather than the private local model update parameters.

The proposed CFL protocol is privacy-preserving facing honest-but-curious target clients. An honest-but-curious target client may also execute model inversion attacks [12] to threaten the privacy contained in the local datasets. To resist such attacks, a random noise is introduced to disturb the aggregated model update parameters. Therefore, all target clients can only get disturbed local contributions or the intermediate aggregation results, rather than the precise local model update parameters.
The proposed CFL protocol is privacy-preserving facing honest-but-curious potential clients. An honest-but-curious potential client may execute eavesdropping attacks [27] to obtain the intermediate aggregation result. Thus, the proposed CFL protocol adopts the authenticated encryption to protect aggregated model update parameters. Therefore, an honest-but-curious potential client cannot eavesdrop the plaintext of other’s local model update parameters.

Furthermore, in the external malicious threat models, we discuss the data integrity and authenticity of the aggregated model update parameters and the confidentiality of the communication keys. The proposed CFL protocol guarantees the data integrity and authenticity of the aggregated model update parameters. A malicious adversary can execute tampering attacks [27] and impersonation attacks [27] to threaten the data integrity and authenticity of the aggregated model update parameters. To defend against such attacks, the proposed CFL protocol adopts authenticated encryption to protect aggregated model update parameters, where the authentication tag can effectively resist unauthorized tampering of messages and verify whether the sender is a trusted source.

The proposed CFL protocol guarantees the confidentiality of the communication keys. More seriously, a malicious adversary can hijack honest participants, which may lead to the leakage of the communication keys, resulting in various attacks to threaten the security of the system. Thus, the Secure Communication Key Establishment Protocol is enhanced by a voting-based key revocation mechanism to remove the contaminated keys, thereby eliminating the possibility of malicious adversaries stealing communication keys. Therefore, the confidentiality of the communication key is ensured.

### 6.2 Network Connectivity Analysis

In the proposed CFL protocol, we use the Secure Communication Key Establishment Protocol to establish communication keys for potential clients. We analyze the network connectivity based on the established communication keys in Theorem 1 which is the basis of secure transmission.

**Theorem 1.** For a cluster $L_h$, the size of the key ring for each potential client only needs to be $m_h$ to ensure a high network connectivity probability.

**Proof.** We first formalize the network topology in cluster $L_h$ as a random graph $G(n_h, r_h)$, where $n_h$ is the amount of potential clients in $L_h$, and $r_h \in [0, 1]$ is the probability that any two potential clients are connected.

To reflect the desired connectivity probability $P_c$ of $G(n_h, r_h)$, we recall Lemma 1.

**Lemma 1 ([27]).** Given a desired connectivity probability $P_c$ for a graph $G(n_h, r_h)$, the threshold function $r_h$ is defined by:

$$P_c = \lim_{n_h \to \infty} \Pr[G(n_h, r_h) \text{ is connected}] = e^{-e^{-c}}, \quad (15)$$

where $r_h = \frac{\ln(n_h)}{n_h} + \frac{c}{n_h}$, and $c$ is any real constant.
According to Lemma 1, given a high connectivity probability (e.g., 0.999), we can calculate $r_h$ as follows:

$$r_h = \frac{\ln(n_h)}{n_h} + \frac{-\ln(-\ln(P_c))}{n_h}.$$

(16)

Thus, we can infer that a graph with $n_h$ vertices has at most $\binom{n_h}{2}$ edges. Assume that $G(n_h, r_h)$ has $e$ edges, we can calculate $r_h$ as follows:

$$r_h = \frac{e}{\binom{n_h}{2}}.$$

(17)

Next, we can calculate the key ring size $m_h$ of the potential client, i.e., the expected degree of $G(n_h, r_h)$, shown as:

$$m_h = 2e/n_h.$$

(18)

Therefore, according to the equations (17) and (18), we can figure out the relationship between $m_h$ and $r_h$ as:

$$m_h = r_h \times (n_h - 1) \sim r_h \times n_h,$$

(19)

where $\sim$ represents an equivalence symbol.

As a result, according to the equation (16), we can calculate the key ring size $m_h$ as:

$$m_h = \ln(n_h) - \ln(-\ln(P_c)),$$

(20)

which complete the proof.

Theorem 1 indicates that the size of the key ring of each potential client only needs to be $m_h = \ln(n_h) - \ln(-\ln(P_c))$ for a high network connectivity probability. In other words, given the size of the key ring $m_h$, all aggregation results can be transmitted in ciphertext with a high probability.

7 Experiments and Evaluation

In this section, we conduct the experiments in a spam classification scenario. All experiments are implemented on the same computing environment (Linux Ubuntu 16.04, Intel i7-6950X CPU, 64 GB RAM and 5TB SSD) with Tensorflow, Keras and PyCryptodome.

7.1 Experimental Design

Dataset. The datasets used in our experiments consist of Trec06p and Trec07, which are two English e-mail datasets from the real world. The Trec06p dataset

1 We give the download link as: https://trec.nist.gov/data/spam.html.
contains 12910 hams and 24912 spams in the main corpus with messages, and the Trec07 dataset contains 25220 hams and 50199 spams. **Network connectivity topology.** As shown in Fig. 5 in our experiments, there are 200 potential clients distributed in a large-scale P2P network which is divided into five clusters. Among the potential clients, 100 target clients are illustrated as the gray nodes, and the black nodes are leader clients. Further, we illustrate the connections among potential clients based on the communication keys as the gray lines and the aggregation route as the red lines.

![Fig. 5: Network connectivity topology.](image)

**Parameter tuning.** In our experiments, all target clients use the local datasets to collaboratively train a global model according to the proposed CFL protocol. It is worth mentioning that the Trec06p dataset is divided into the Trec06p training set and the Trec06p testing set according to the ratio of 3:1, and the Trec06p training set is adopted to train the original global model. In addition, the Trec07 dataset is divided into two parts. One part is further divided into 100 local datasets for target clients, and the set size follows a normal distribution of mean 600 and variance 100. The other part serves as the Trec07 testing set to evaluate the global model performance. The original global model and local models have the same structure, which consists of two convolution layers, two pooling layers, and three fully connected layers. We also set the loss function as the cross-entropy error and the active function as the ReLU. The gradient descent algorithm is set as SGD with the learning rate of 0.1. Besides, we use the AES-GCM-128bit algorithm [23] for authenticated encryption.

### 7.2 Experimental Results and Evaluation

In order to validate the model performance of the proposed CFL protocol, we evaluate the accuracy and loss value of the global model in Fig. 6. After 14 rounds of global model training, the global model converges, and the accuracy and loss value of the final global model are 99.32% and 0.0356 on the Trec07 testing set. The results show that the proposed CFL protocol can guarantee the convergence of the global model with high classification accuracy.
Moreover, we compare the performances of the original global model and the final global model by illustrating the receiver operating characteristic (ROC) curve and the area under the curve (AUC) on both Trec06p and Trec07 testing sets in Fig. 7. The AUC of the final global model is larger than that of the original global model on the Trec07 testing sets and is close to the AUC on the Trec06p testing set, which shows that the final global model outperforms the original global model in terms of model generalization.

Besides, in order to validate the dropout-robustness of the proposed CFL protocol, we randomly set 15% dropout clients in each training round. As shown in Fig. 6, in this case, the global model converges at 16 rounds, and the accuracy and loss value of the global model are 99.26% and 0.0343, which are close to the values without dropout clients. Such a result shows that the proposed CFL protocol ensures the classification accuracy and the convergence rate of the global model facing dropout issues. Similar results can also be observed in Fig. 7, where the final global model obtains AUC values of 0.97 and 0.98 on the Trec06p and Trec07 testing sets, which also reflect the dropout-robustness of the proposed CFL protocol.

More importantly, to demonstrate the communication efficiency of the proposed CFL protocol, we record the averaging communication times achieving a round of aggregation in TABLE I. Compared to the PPT protocol [9], CFL
improves the communication efficiency by about 43.25%. Such an improvement benefits from the cluster-based division. It is worth mentioning that the communication times are not necessarily negatively correlated with the number of dropout clients but are related to the distribution of network topology.

### Table 1: Communication Times

| Percentage of dropout clients (%) | 0    | 1   | 2   | 5   | 10  | 15  |
|----------------------------------|------|-----|-----|-----|-----|-----|
| CFL's communication times       | 105  | 106 | 104 | 105 | 102 | 92  |
| PPT's communication times       | 191  | 186 | 185 | 184 | 172 | 164 |

In addition, we record the time of each operation in TABLE II, where the authenticated encryption (decryption) in our protocol consumes less time than the encryption (decryption) and signature (verification) in the PPT protocol [9], which achieves a 0.87% improvement in terms of computational efficiency. TABLE II also shows the time for a single leader client in our protocol to perform noise generation, addition, and subtraction operations, which are almost the same as the PPT protocol.

### Table 2: Computational Performance

| Index | Operations† | Time(ms/1000byte) |
|-------|-------------|-------------------|
| 1     | Encryption(AES-128bit) | 170.8248* |
| 2     | Decryption(AES-128bit)  | 0.0282* |
| 3     | Signature(Elgamal-2048bit) | 0.0003* |
| 4     | Verification(Elgamal-2048bit) | 0.0071* |
| 5     | Encryption (AES-GCM-128bit) | 169.3596 |
| 6     | Decryption(AES-GCM-128bit)  | 0.0163 |
| 7     | Noise generation | 1.2368 |
| 8     | Noise addition     | 0.0877 |
| 9     | Noise subtraction  | 0.1316 |

† Local model update parameters size: 114MB in plaintext and 428.8MB in ciphertext.
* Based on the same experimental settings, we directly use the results of relevant operation time in the PPT protocol [9] for comparison.

### 8 Conclusion

Aiming to solve the problems of a single point of failure and high communication costs in FL, we propose a CFL protocol for FL in large-scale P2P networks, focusing on efficient and privacy-preserving model training. Some interesting researches could further improve CFL in the future. On the one hand, an effective design for FL in dynamic P2P networks could flourish the development of decentralized FL in practice. On the other hand, along with the hardware development of clients, a computation-friendly homomorphic encryption algorithm could improve the communication efficiency of the system more.
References

1. Assran, M., Loizou, N., Dallas, N., Rabbat, M.: Stochastic gradient push for distributed deep learning. In: Proceedings of the 36th International Conference on Machine Learning, 9-15 June 2019, Long Beach, CA, USA. pp. 344–353. PMLR (2019)
2. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Advances in Cryptology, 3-7 December 2000, Kyoto, Japan. vol. 1976, pp. 531–545. Springer (2000)
3. Bhandari, V., Vaidya, N.H.: On reliable broadcast in a radio network. In: Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing, 17-20 July 2005, Las Vegas, NV, USA. pp. 138–147. ACM (2005)
4. Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečný, J., Mazzocchi, S., McMahan, B., et al.: Towards federated learning at scale: System design. In: Proceedings of Machine Learning and Systems 2019, 31 March-2 April 2019, Stanford, CA, USA. vol. 1, pp. 374–388. mlsys.org (2019)
5. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 30 October-3 November 2017, Dallas, TX, USA. pp. 1175–1191. ACM (2017)
6. Bondy, J.A., Murty, U.S.R., et al.: Graph theory with applications, vol. 290. Macmillan Education UK (1976)
7. Boyd, S., Ghosh, A., Prabhakar, B., Shah, D.: Randomized gossip algorithms. IEEE Transactions on Information Theory 52(6), 2508–2530 (2006)
8. Chan, H., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks. In: 2003 IEEE Symposium on Security and Privacy, 11-14 May 2003, Berkeley, CA, USA. pp. 197–213. IEEE Computer Society (2003)
9. Chen, Q., Wang, Z., Zhang, W., Lin, X.: PPT: A privacy-preserving global model training protocol for federated learning in p2p networks. arXiv preprint arXiv:2105.14408 (2021)
10. Deng, L., Li, X.: Machine learning paradigms for speech recognition: An overview. IEEE Transactions on Audio, Speech, and Language Processing 21(5), 1060–1089 (2013)
11. Dubey, A., Pentland, A.: Differentially-private federated linear bandits. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 6-12 December 2020, virtual. pp. 6003–6014. Curran Associates, Inc. (2020)
12. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 12-16 October 2015, Denver, CO, USA. pp. 1322–1333. ACM (2015)
13. Gibiansky, A.: Bringing hpc techniques to deep learning. Baidu Research, Tech. Rep. (2017)
14. He, C., Tan, C., Tang, H., Qiu, S., Liu, J.: Central server free federated learning over single-sided trust social networks. arXiv preprint arXiv:1910.04956 (2019)
15. Hu, C., Jiang, J., Wang, Z.: Decentralized federated learning: A segmented gossip approach. arXiv preprint arXiv:1908.07782 (2019)
16. Hu, Q., Du, B., Markantonakis, K., Hancke, G.P.: A session hijacking attack against a device-assisted physical-layer key agreement. IEEE Transactions on Industrial Informatics 16(1), 691–702 (2019)
17. Jordan, M.I., Mitchell, T.M.: Machine learning: Trends, perspectives, and prospects. Science 349(6245), 255–260 (2015)
18. Konečný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
19. Li, Q., Wen, Z., He, B.: Practical federated gradient boosting decision trees. In: The 34th AAAI Conference on Artificial Intelligence, 7-12 February 2020, New York, NY, USA. vol. 34, pp. 4642–4649. AAAI Press (2020)
20. Li, S., Xu, L.D., Zhao, S.: The internet of things: a survey. Information Systems Frontiers 17(2), 243–259 (2015)
21. Lian, X., Zhang, C., Zhang, H., Hsieh, C., Zhang, W., Liu, J.: Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, 4-9 December 2017, Long Beach, CA, USA. pp. 5330–5340. Curran Associates, Inc. (2017)
22. Marfoq, O., Xu, C., Neglia, G., Vidal, R.: Throughput-optimal topology design for cross-silo federated learning. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 6-12 December 2020, virtual. vol. 33, pp. 19478–19487. Curran Associates, Inc. (2020)
23. McGrew, D., Viega, J.: The galois/counter mode of operation (gcm). submission to NIST Modes of Operation Process 20. 0278–0070 (2004)
24. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, 20-22 April 2017, Fort Lauderdale, FL, USA. vol. 54, pp. 1273–1282. PMLR (2017)
25. Ramanan, P., Nakayama, K.: Baffle: Blockchain based aggregator free federated learning. In: IEEE International Conference on Blockchain, 2-6 November 2020, Rhodes, Greece. pp. 72–81. IEEE (2020)
26. Ramanathan, R., Redi, J.: A brief overview of ad hoc networks: challenges and directions. IEEE Communications Magazine 40(5), 20–22 (2002)
27. Stallings, W.: Cryptography and network security, 4/E. Pearson Education India (2006)
28. Stejkośka, B.L.R., Trivodaliev, K.V.: A review of internet of things for smart home: Challenges and solutions. Journal of Cleaner Production 140, 1454–1464 (2017)
29. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM Journal on Computing 1(2), 146–160 (1972)
30. Voigt, P., Von dem Bussche, A.: The eu general data protection regulation (gdpr). A Practical Guide, 1st Ed., Cham: Springer International Publishing 10(3152676), 10–5555 (2017)
31. Yang, K., Jiang, T., Shi, Y., Ding, Z.: Federated learning based on over-the-air computation. In: 2019 IEEE International Conference on Communications, 20-24 May 2019, Shanghai, China. pp. 1–6. IEEE (2019)
32. Yang, S., Ren, B., Zhou, X., Liu, L.: Parallel distributed logistic regression for vertical federated learning without third-party coordinator. arXiv preprint arXiv:1911.09824 (2019)
33. Zhang, C., Ma, Y.: Ensemble machine learning: methods and applications. Springer (2012)
34. Zhou, L., Pan, S., Wang, J., Vasilakos, A.V.: Machine learning on big data: Opportunities and challenges. Neurocomputing 237(6245), 350–361 (2017)