



ESA-FedGNN: Efficient secure aggregation for federated graph neural networks

Yanjun Liu¹ · Hongwei Li¹ · Xinyuan Qian¹ · Meng Hao¹

Received: 10 November 2022 / Accepted: 23 February 2023 / Published online: 23 March 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Graph Neural Network (GNN) architecture is a state-of-the-art model, which can obtain complete node embedding features and rich data information by aggregating the information of nodes and neighbors. Therefore, GNNs are widely used in electronic shopping, drug discovery (especially for the treatment of COVID-19) and other fields, promoting the explosive development of machine learning. However, user interaction, data sharing and circulation are highly sensitive to privacy, and centralized storage can lead to data isolation. Therefore, Federated Learning with high efficiency and strong security and privacy enhancement technology based on secure aggregation can improve the security dilemma faced by GNN. In this paper, we propose an Efficient Secure Aggregation for Federated Graph Neural Network(ESA-FedGNN), which can efficiently reduce the cost of communication and avoid computational redundancy while ensuring data privacy. Firstly, a novel secret sharing scheme based on numerical analysis is proposed, which employs Fast Fourier Transform to improve the computational power of the neural network in sharing phase, and leverages Newton Interpolation method to deal with the disconnection and loss of the client in reconstruction phase. Secondly, a regular graph embedding based on geometric distribution is proposed, which optimizes the aggregation speed by using data parallelism. Finally, a double mask is adopted to ensure privacy and prevent malicious adversaries from stealing model parameters. We achieve $O(\log N \log(\log N))$ improvements compared to $O(N^2)$ in state-of-the-art works. This research helps to provide security solutions related to the practical development and application of privacy-preserving graph neural network technology.

Keywords Federated learning · Graph neural network · Secure aggregation · Fast fourier transform · Privacy preserving · Newton interpolation

1 Introduction

In recent years, machine learning technology driven by data and computing resources is ushering in an explosive development, which is widely used in medicine, image processing, face recognition and other fields [1]. In the field of machine learning, Graph Neural Network (GNN) occupies a pivotal position. It can perform graph analysis and graph data modeling from complex graph structures in various fields, and mine valuable information. GNN architecture can obtain complete node embedding features and rich data information by aggregating the information of nodes and

neighbors, so the graph neural network has been rapidly developed [2]. In the real world, GNN has a wide range of application scenarios, such as electronic shopping [3], chemical molecules [4], traffic flow modeling [5], etc. More over, GNN also benefits us humans, not only can detect fake news [6] but also drug discovery for the treatment of COVID-19 [7].

However, in practice, the exchange and circulation of data are highly sensitive to privacy. Due to regulatory restrictions and commercial competition, most enterprises and organizations exist in the form of isolated islands, which is the so-called data isolation phenomenon. Therefore, how to use enterprise and organizational graph data to collaboratively train a high-quality model structure reasonably and legally without exposing data privacy is a research hotspot in recent years. Federated Learning (FL) effectively solves this data isolation problem in a privacy-preserving way [8, 9].

Federated learning is a distributed machine learning method that supports large-scale decentralized client

✉ Hongwei Li
hongweili@uestc.edu.cn

¹ School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

collaborative model training, which can realize the technical ability of "available and invisible" of secure and reliable data, this not only removes the correlation between the size of the machine learning capacity and the data storage capacity, but also protects the privacy of client data [10, 11]. Intuitively, integrating federated learning into GNNs can effectively address data privacy and secure communication issues. However, most of the analysis and research of federated learning now mainly focus on the feature extraction of Euclidean spatial data, structured data such as speech and text [10], seldom deal with non-Euclidean spatial data such as social networks and knowledge graphs. And an attacker can exploit the node embedding to reverse infer the training samples in GNN [12]. Therefore, how to balance practicality, security and efficiency in the study of privacy-preserving federated graph neural networks is a major issue for our future development.

1.1 Related work

In this section, we briefly review the literature on Federated Graph Neural Network and Privacy-preserving model aggregation.

Privacy-preserving model aggregation Although federated learning can fully release the computing and communication value of data through its own distributed data architecture, it has not fulfilled its responsibilities in defending against attacks, and still needs to rely on cryptography and other related technologies to solve privacy issues. Recently, a secure aggregation scheme has been applied to FL, which can effectively protect the model parameters from being attacked during the uploading process [13]. Therefore, the federated learning scheme based on secure aggregation is the focus of our research, which mainly includes the following aspects: differential privacy, homomorphic encryption, and secure multi-party computation.

From the perspective of life cycle, the secure aggregation based on Differential Privacy (DP) is to add noise to the calculation result of the data, which achieves the effect of confusion [14, 15] combined centralized differential privacy (CDP) and Local Differential Privacy (LDP) in federated learning to ensure that attackers cannot infer sensitive information based on output differences, which is more efficient and practical than LDP alone. Although differential privacy can protect data privacy, it causes some losses to the accuracy and usability of model results.

Homomorphic encryption (HE) is a great discovery, Gentry released the first fully homomorphic encryption method that can do both addition and multiplication on ciphertext in 2009 [16]. HE focus on the security of the

data processing process, that is to say, performing various operations on the encrypted data, then after decryption, it is exactly the result of operating on the original plaintext data [17]. At present, the biggest bottleneck of homomorphic encryption is that the computational overhead is too large, the efficiency is low, and it lacks practicality.

Secure Multi-Party Computing (MPC) is a solution to the problem of collaborative computing to protect privacy between a group of untrusting clients [18]. Through secure algorithms and protocols, each client can guarantee the independence of input, the correctness of calculation, and in addition to their own input, do not disclose any other input and output information that can be used for deduction [19]. The Shamir-based secret sharing proposed by [12] better protects the private information of clients and servers on the basis of secure aggregation, but its communication and computing overhead is too high. So et al. [20] proposed the use of additional secret sharing and new coding techniques to achieve the robustness of the protocol, but it is not suitable for the joining of new clients in real-time, because a new system configuration is required to generate the mask.

Federated graph neural network The rise and application of the federated graph neural network has successfully promoted the research in the field of artificial intelligence. It couples the graph neural network and the federated learning algorithm, and learns from each other to achieve a better way of dealing with problems. It is widely used in various high-risk scenarios, such as financial analysis, drug discovery and other important fields [21]. Wu et al. [3] proposed a GNN-based personalized federated recommendation system FedGNN, which applies local differential privacy and pseudo-interaction history information to anonymize local gradients, extending and improving the local user item graph in a privacy-preserving manner. Zhang et al. [22] fused FedAvg and GraphSage models to jointly train local node features to obtain missing topologies. Chen et al. [23] proposed a general federated graph learning framework FedGL, which handles the complementarity and heterogeneity of graph data by uploading the client's model parameters, prediction results, node embeddings to the server, and protects privacy through global self-supervision information. Zheng et al. [24] proposed a separate federated GNN model, which decouples the training process into two parts: GraphSAGE calculates the model parameters locally, and calculates the loss function on the server. At the same time, the model parameters are uploaded through homomorphic encryption or protected by secret sharing. In this paper, combining the advantages of federated learning and graph neural networks, we propose a novel learning framework for federated graph neural networks with privacy-preserving functions.

1.2 Our contributions

In this paper, we clarify the differences and connections between federated learning, secure aggregation, and graph neural networks, and conduct coupling research for commonalities, it has played a major role in promoting the solution of the secure problems faced by the graph neural network. It employs federated learning to improve computing power in graph neural networks, allows clients to jointly train machine learning models without disclosing private datasets, and achieves high efficiency and high security while ensuring model accuracy. It uses secure aggregation to hide model parameters during model training, prevent malicious adversaries from stealing model parameters, and ensure model data and user privacy. Based on the above two research results, to coordinate federated learning and secure aggregation, with the goal of improving the security of graph neural networks, this paper proposes the core framework of this paper—Efficient Secure Aggregation for Federated Graph Neural Network(ESA-FedGNN). We summarize the main contributions as follows:

- Firstly, a secret sharing scheme based on numerical analysis is proposed, which uses Fast Fourier Transform (FFT) in the data sharing stage to improve the computational power of graph neural network, and achieves high efficiency and high security while ensuring the model accuracy. In the recovery phase, a flexible and easy-to-use Newton interpolation method is used to deal with the disconnection and loss of clients, which is beneficial to large-scale clients participating in collaborative training.
- Secondly, a regular graph embedding based on geometric distribution is proposed. Using data parallelism, it can not only ensure that there are enough neighbor nodes to satisfy its security, but also maximize the use of computing resources and optimize the aggregation speed.
- Finally, a double mask is used to ensure privacy, prevent malicious adversaries from stealing model parameters, and ensure the privacy and security of model parameters and clients.

This paper comprehensively and deeply analyzes the ESA-FedGNN protocol in terms of computing power, communication cost, privacy and robustness to client disconnection, and obtains a secure aggregation framework with high efficiency and strong privacy. We theoretically focus on analyzing the correctness and security of this protocol, reduce the time overhead to $O(\log N \log(\log N))$, and provide a strong basis for the actual development and application of technology.

2 Preliminaries

2.1 Federated learning

Federated Learning is a machine learning framework that effectively helps multiple organizations conduct data training and machine learning modeling while meeting the common requirements of user privacy protection, data security and government regulations. It works as follows: client terminal (such as mobile devices or groups) from a central server (such as service providers) to download the existing prediction model, the model is trained by using the local data, and the model updating content uploaded to the cloud, training model through the model updates to the fusion of different terminal, repeat the above process, until the model convergence. It embodies the principles of centralized collection and data minimization and can mitigate many of the systemic privacy risks and costs generated by traditional centralized machine learning.

2.2 Graph neural network

Graph Neural Network (GNN) is a deep learning method based on the graph structure. GNN learns the features of nodes and edges through a certain strategy, converts the graph structure into a neat and standardized representation, and undergoes various operations such as multi-layer graph convolution and activation function, and finally get the embedding of each node to facilitate tasks such as node classification, link prediction, graph and subgraph generation, etc.

GNN in this paper is modeled with a Message Passing Neural Network (MPNN) framework, it unifies various graph neural network and graph convolutional network methods [25]. The GNN training process consists of two stages : Message-passing (same for all tasks)and Readout (different across tasks). The first stage consists of two steps: (1) Aggregating local neighborhood embeddings of nodes (2) Updating the state of nodes.

In the framework of this paper, we assume that there are N clients, and each client has its own graph data $G = (V, E)$, where V is a set of fixed points, E is a set of edges, and the edge connecting fixed points i and j is e_{ij} . Now, an MPNN containing T layers is formalized as follows:

$$m_i^{(k,t+1)} = \text{AGGREGATE}\left(\left\{M_i\left(h_i^{(k,t)}, h_j^{(k,t)}, e_{ij}\right)\right\}j \in N_i\right) \quad (1)$$

$$h_i^{(k,t+1)} = U_i\left(h_i^{(k,t)}, m_i^{(k,t+1)}\right) \quad (2)$$

where $h_i^{(k,0)} = x_i^{(k)}$ is the node feature of the k th client, AGGREGATE is the aggregation function, and the

aggregation function is symmetric, and the input arrangement is unchanged. (e.g in the GraphSAGE model, the aggregation function can be a MEAN operation, a SUM operation or an LSTM operation.), N_i is the local neighbor set of node i , M_i is the message function, U_i is the update function.

$$\hat{y} = R\left(\left\{h_i^{(k,T)} \mid i \in G_S\right\}\right) \quad (3)$$

Note R is readout function, which can learn differentiable functions, S stands for handling different downstream tasks, For example Graph Classification, Link Prediction and Node Classification. We propose an Efficient Secure Aggregation for Federated Graph Neural Network(ESA-FedGNN), as shown in Fig. 1.

2.3 Shamir secret sharing

Shamir secret sharing is a technique for sharing secrets between a group of participants [26]. It is mainly used to protect important information from being lost, damaged, or tampered with. By dividing the secret S and sharing the secret among N participants, the secret value can be calculated or restored only if more than any t participants cooperate, while the secret value can not be restored if less than t participants cooperate. In a nutshell, Shamir secret sharing involves two steps:

- **Sharing Phase:** given a finite field F_q (q is a prime). The secret message S takes the parameter $\{a_1, a_2, \dots, a_i, \dots, a_{t-1}\}$, $i \in N$ and generates the polynomial $\varphi(x)$, as shown below:

$$\varphi(x) = a_0 + a_1x + a_2x^2 + \dots + a_{t-1}x^{t-1} \quad (4)$$

where $a_0 = \varphi(0) = S$, that is, the original secret, defines any coefficient a_1, a_2, \dots, a_{t-1} , given a set of input values x_1, x_2, \dots, x_N on F_q , $(x_1, S_1), \dots, (x_N, S_N)$ are calculated respectively. Distribute the secret S to N participants and get N group shares $(x_1, S_1), \dots, (x_N, S_N)$. The secret S can be recovered by calculating Lagrange interpolation as long as t sub-secret shares of N participants are obtained.

- **Reconstruction Phase:** known secret shares $(x_1, S_1), \dots, (x_N, S_N)$, construction polynomial $\varphi(x)$, as shown below:

$$\varphi(x) = \sum_i^t \prod_{1 \leq j \leq t, j \neq i} \frac{x_0 - x_j}{x_i - x_j} \varphi(i) \quad (5)$$

where the Lagrange coefficient is $L_i(x) = \prod_{i \neq j, 1 \leq j \leq t} \frac{x_0 - x_j}{x_i - x_j}$, the value of $\varphi(0)$ is the secret S .

2.4 Cryptographic primitives

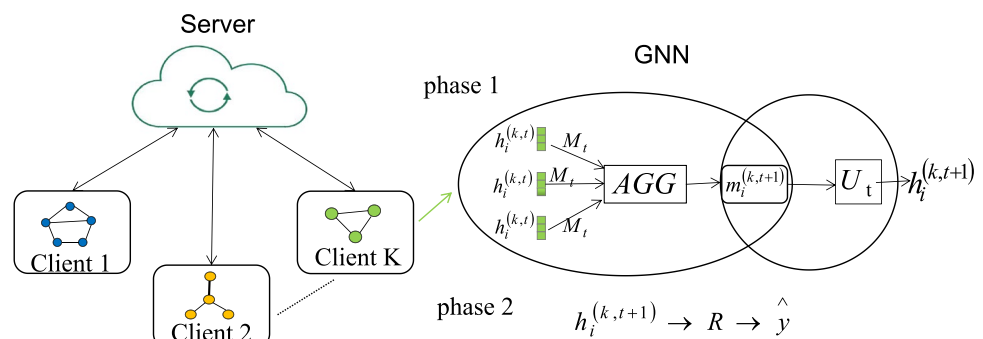
2.4.1 Key agreement

The Key Agreement protocol mainly consists of three algorithms (KA.param, KA.gen, KA.agree). Parameter generation algorithm $pp \leftarrow \text{KA.param}(\lambda)$, some public parameters are generated after the safety parameter is given, over which the protocol will be parameterized. Key generation algorithm $(pk_i, sk_i) \leftarrow \text{KA.gen}(pp)$ allows any party i to generate a private-public key pair. Key agreement program $k_{i,j} \leftarrow \text{KA.agree}(sk_i, pk_j)$ allows the private key of client i and the public key of client j are negotiated to obtain a private shared key $K_{i,j}$. Here according to using KA.gen with the same parameter pp , its correctness suggests that: $\text{KA.agree}(sk_i, pk_j) = \text{KA.agree}(sk_j, pk_i)$. The specific Key Agreement scheme we will use is Diffie-Hellman key agreement [27].

2.4.2 Pseudorandom generator

The pseudorandom generator is used to calculate the pseudorandom numbers from a series of seed values when the system needs random numbers. His input is a uniform random seed of some fixed length whose output space is $[0, R]^m$ (the input space of the protocol). The security of the pseudorandom generator ensures that the PRG output generated by the honest user is indistinguishable from the real output [28].

Fig. 1 An illustration of the Federated Graph Neural Network (ESA-FedGNN)



2.4.3 Authenticated encryption

Authentication encryption is to provide identity authentication function while encrypting, and ensuring the confidentiality and integrity of data. The authentication encryption here includes an encryption algorithm $AE. enc$ that encrypts a piece of plaintext with a key and a decryption algorithm $AE. dec$ that simultaneously decrypts a piece of plaintext with the same key. For correctness, we require that for all keys $c \in \{0, 1\}^\lambda$ and all messages x , $AE \cdot dec(c, AE. enc(c, x)) = x$ [29].

3 System model

Federated learning is a distributed learning framework that uses multiple clients to train machine learning models locally, which can not only protect privacy, improve performance, but also extend to larger scale model training. This paper uses the GraphSAGE framework in the message passing neural network [30]. Compared with GCN using full graph information for training, GraphSAGE aggregates neighbors by sampling, which is more suitable for large-scale graph training.

3.1 System architecture

ESA-FedGNN allows the use of client-server communication architectures [31]. That is to say, N clients with the same data structure train a machine learning model together with the help of the server. It is worth noting that there is no direct communication between each client, which is transmitted through the server.

3.2 Threat model

In ESA-FedGNN protocol, we focus on honest but curious adversary settings. We generally assume that clients are honest in a horizontal federated learning system. The object to be careful and guarded against is an honest but curious server. Specifically, the adversary honestly adheres to the protocol, but tries to learn from the received information and infer information other than the out-put [31].

4 ESA-FedGNN protocol

4.1 High-level overview

We first outline the proposed ESA-FedGNN protocol framework, this paper mainly studies the scenario of horizontal federation. It focus on the honest but curious

adversary settings, See the ESA-FedGNN Protocol for the specific protocol process. the system is initialized as follows:

1. The participants of the protocol are the server and N clients.
2. Each client initializes a security parameter λ to generate DH related parameters $pp \leftarrow KA \cdot param(\lambda)$.
3. Specifies the threshold value $t \geq \frac{\log N}{2}$.
4. The size of the transmitted data vector is L , $x_i (x_i \in Z_R)$ defined as model parameter to input, the total aggregate result in the server output is $z (z \in Z_R)$.
5. Each client and server has an authenticated channel.
6. Client i obtains the private key from the third party and gives the public key to all other clients j .

In ESA-FedGNN protocol, we introduce a double-mask structure to protect the aggregation results. First, the server sends a random number to each client, which is used to prevent the server from colluding with other clients; second, in the sharing phase, each client generates shares by Fast Fourier Transform(FFT) and sends it to other clients, adding a secondary safeguard for the clients.

In the reconstruction phase, the server receives responses from at least t clients in the k th group, otherwise the protocol is terminated. After receiving the final aggregated value the server first subtracts the random mask sum sent out by the server, secondly, collects at least t shares for all dropped clients, and after recovering the original value by Newton interpolation, the server subtracts all masks to obtain the final aggregated value which is also the model parameter.

Handling dropped clients The method generally used in dealing with the client dropout problem is to notify online clients in the reconstruction phase and let them upload the seeds generated jointly with the dropped client. Two problems will arise at this time, one is that other online clients suddenly drop out when uploading seeds in the reconstruction phase, then the next round will need to continue to process this new dropout, with endless results. Second, the previously dropped client is not really dropped, but the network delay is large and arrives after the server asks other clients, then the server can easily calculate the privacy data of the dropped client.

Therefore, we use the Newton interpolation method to solve the client dropout problem by finding the difference quotient of each order, an expression obtained recursively, when adding interpolation nodes, as long as part of the computational workload is added to the original one, while the original computational results are still available, even if other participants drop out in the reconstruction phase, as long as some minimum number (equal to the threshold) of participants can keep the response and be recovered.

4.2 A novel secret sharing scheme based on numerical analysis

Firstly, we start with Shamir's idea of secret sharing, which is a very classic cryptographic technique. User i holds secret S , which chooses a degree of $t - 1$ polynomial φ , so that $\varphi(0) = S$. Then send the secret share $\varphi(i)$ to N participants respectively, in the reconstruction process, participants out of N participants recover $S = \varphi(0)$ with Lagrange interpolation. In fact, you don't have to worry about what the specific subset is in the reconstruction process, but you have any t subset to recover the secret S . If the threshold t is set to $N/2$, the processing time and the number of participants show a quadratic function. In fact, when N is relatively small, we can get relatively good performance. As N increases, the performance of the scheme decreases. If N is very large, the practicability is beyond the scope.

4.2.1 Sharing phase—fast fourier transform

In this subsection, we focus on the secret sharing stage in Shamir, where we employ the Fast Fourier Transform (FFT) to reduce the time overhead to $O(N \log N)$, this not only increases efficiency, but also maximizes the value of data. Our key technique is to convert polynomials to value representations using the Fast Fourier Transform. Assume an n th degree polynomial $\varphi(x) = \sum_{i=0}^{n-1} a_i x^i$, without loss of generality, let $n = 2^s$ (treat n as an integer power greater than or equal to 2), find its value at N points, please note that the N points here are not arbitrarily selected, but the n -th roots of F_q . Specific steps are as follows:

Step 1 : Divide the terms in $\varphi(x)$ into two parts according to the parity of the a_i subscripts, namely

$$\begin{aligned} \varphi(x) = & (a_0 + a_2 x^2 + a_4 x^4 + \cdots + a_{n-2} x^{n-2}) \\ & + (a_1 x^1 + a_3 x^3 + a_5 x^5 + \cdots + a_{n-1} x^{n-1}) \end{aligned} \quad (6)$$

Let us define

$$\varphi_1(x) = a_0 + a_2 x^1 + a_4 x^2 + \cdots + a_{n-2} x^{\frac{n}{2}-1} \quad (7)$$

$$\varphi_2(x) = x(a_1 + a_3 x^1 + a_5 x^2 + \cdots + a_{n-1} x^{\frac{n}{2}-1}) \quad (8)$$

Then

$$\varphi(x) = \varphi_1(x^2) + x\varphi_2(x^2) \quad (9)$$

Step 2 : Use the special property of the roots of the unity complex number, where ω_n^k is a primitive N th root of unity.

When $x = \omega_n^k \left(k < \frac{n}{2}\right)$, and calculate

$$\begin{aligned} \varphi(\omega_n^k) &= \varphi_1(\omega_n^{2k}) + \omega_n^k \varphi_2(\omega_n^{2k}) \\ &= \varphi_1\left(\omega_n^{\frac{k}{2}}\right) + \omega_n^k \varphi_2\left(\omega_n^{\frac{k}{2}}\right) \end{aligned} \quad (10)$$

When $x = \omega_n^{k+\frac{n}{2}} \left(k < \frac{n}{2}\right)$ and calculate

$$\begin{aligned} \varphi\left(\omega_n^{k+\frac{n}{2}}\right) &= \varphi_1(\omega_n^{2k+n}) + \omega_n^{k+\frac{n}{2}} \varphi_2(\omega_n^{2k+n}) \\ &= \varphi_1(\omega_n^{2k} \cdot \omega_n^n) + \omega_n^k \varphi_2(\omega_n^{2k} \cdot \omega_n^n) \\ &= \varphi_1\left(\omega_n^{\frac{k}{2}}\right) - \omega_n^k \varphi_2\left(\omega_n^{\frac{k}{2}}\right) \end{aligned} \quad (11)$$

Step 3 : Observe the structure of the two formulas 1 and 2, we only need to ask for $\varphi_1\left(\omega_n^{\frac{k}{2}}\right)$, $\varphi_2\left(\omega_n^{\frac{k}{2}}\right)$, and then after similar steps, we can continue to transform the problem into finding $\varphi_1\left(\omega_n^{\frac{k}{4}}\right)$, $\varphi_2\left(\omega_n^{\frac{k}{4}}\right)$ with a time complexity of $O(1)$, the final problem is transformed into the

$$\varphi_1(\omega_1^k) = \varphi_2(\omega_1^k) = 1 \quad (12)$$

Therefore, $\varphi(\omega_n^k)$, $\varphi(\omega_n^{k+\frac{n}{2}})$ can be obtained with a time complexity of $O(\log N)$, and all $\varphi(\omega_n^k)$ can be obtained with a time complexity of $O(N \log N)$.

The value of FFT at the unit complex root can greatly reduce the amount of calculation. Not only that, this paper uses bit-reversal permutation [32] to simulate the divide and conquer steps of recursive FFT, so as to optimize FFT, which can avoid consuming additional memory space. Now suppose $N = 8$ and ω_8 is the 8th root of unity, φ is evaluated at evaluation point $\omega_8^0, \omega_8^1, \dots, \omega_8^7$. The example is as follows:

The recursive method is used for optimization. Due to the properties of $\omega_n^{k+\frac{n}{2}} = -\omega_n^k$, each layer can be easily simplified. The first layer is converted to $(x - \omega_8^0)(x - \omega_8^4) = x^2 + \omega_8^4$, similarly, the second layer is $(x - \omega_8^0)(x - \omega_8^4)(x - \omega_8^2)(x - \omega_8^6) = x^4 + \omega_8^4$, and finally get $\prod_{i=0}^7 (x - \omega_8^i) = (x^4 + \omega_8^0)(x^4 - \omega_8^4) = x^8 - 1$. Due to the advantage of unit complex roots, the accumulator can be optimized as $(x^8 - 1) = \prod_{i=0}^7 (x - \omega_8^i)$.

4.2.2 Reconstruction phase—newton interpolation method

In this subsection, we focus on the secret recovery phase in Shamir. In the beginning, the recovery phase uses Gaussian Elimination, and its algorithmic complexity is $O(N^3)$, that is, if the coefficient matrix is $N \times N$, then the amount of computation required by the Gaussian Elimination method is approximately proportional to n^3 , which exponentially increases the computational complexity. With the deepening of the theoretical understanding of data elements by researchers, the Lagrange Interpolation method is used for improvement, reducing the time cost to $O(N^2)$ [33]. The

formula of Lagrangian interpolation method has a complete and compact structure, and it is very simple in theoretical analysis. However, when the number of interpolation points increases or decreases, the corresponding basis function needs to be recalculated, and the entire formula will change, and the process is very cumbersome. And when there are many interpolation points, the degree of the polynomial will be very high, and the value will be unstable. What's more, the algorithm in this paper is aimed at the environment where the network is unstable, easy to delay, and there are many participants. Therefore, the Lagrange Interpolation method is not suitable for application in the algorithm of this paper. Based on this, we adopt the Newton Interpolation method to calculate the secret recovery phase, which not only reduces the computational complexity to $O(N \log N)$, but also has good numerical stability when the number of interpolation points increases [32].

The advantage of the Newton interpolation method is inheritance, that is, only one more item needs to be added after the newly added node. This inheritance makes Newton interpolation more flexible and easy to use than Lagrange Interpolation in some situations. The Newton interpolation method is applied in the fourth round of the algorithm in this paper, the specific process is as follow:

Step 1: Define the Newton interpolation polynomial as

$$\varphi(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \dots + b_t(x - x_0)(x - x_1) \dots (x - x_{t-1}) \quad (13)$$

where $b_k (k = 0, 1, 2, \dots, t)$ is the undetermined coefficient.

Step 2: The online client needs to upload its own ID and the corresponding function value to the server, and the server calculates the corresponding difference quotient table. First, define first-order difference quotient, that is, the ratio of the difference between the independent variable and the difference between the dependent variable is called the difference quotient. The expression is: the average rate of change over interval $[x_i, x_{i+1}]$ is $\varphi[x_i, x_{i+1}] = \frac{\varphi(x_{i+1}) - \varphi(x_i)}{x_{i+1} - x_i}$. Secondly, Similarly, the K-order difference quotient can be obtained $\varphi[x_0, x_1, \dots, x_k] = \frac{\varphi[x_1, x_2, \dots, x_k] - \varphi[x_0, x_1, \dots, x_{k-1}]}{x_k - x_0}$. Finally, according to the properties of the difference quotient, the difference quotient table can be obtained.

Step3: From the Newton interpolation polynomial and the difference quotient, it can be deduced that the coefficients in the Newton difference formula are

$$\begin{aligned} b_0 &= \varphi(x_0) \\ b_1 &= \varphi[x_0, x_1] \\ b_2 &= \varphi[x_0, x_1, x_2] \end{aligned} \quad (14)$$

Where the general formula is $b_k = \varphi[x_0, x_1, \dots, x_k]$. The coefficients are put into the polynomial to get the t -th

Newton interpolation formula. Bring in $x = 0$ to get the secret value of $S = \varphi(0)$. The Newton interpolation method not only overcomes the shortcoming of adding a node and restarting the entire calculation amount, but also saves the times of multiplication and division, and ensures that the client is suitable for environments such as network delay and unstable equipment, which is conducive to large-scale client participation and collaboration train.

4.3 A regular graph embedding based on geometric distribution

In the secure aggregation article proposed by Bonawitz et al., a complete graph is chosen to construct the network architecture. If a graph has N vertices, then when there is an edge between any two vertices, the graph is called a complete graph, that is, any two of the N clients need to transmit the public key and secret share through the server. In fact, it is completely beyond its safety to add any $N - 1$ sub-information to each client, which seriously wastes computing resources and storage space, so we choose to use multiple regular graph parallel operations to replace a large complete graph.

A good regular graph needs to satisfy safety and correctness. Security needs to satisfy the following two points: each client's neighbors do not have too many corrupt neighbors (less than t), and all clients remain connected after excluding corrupt parties and dropped clients. Satisfying the correctness is that after deleting the dropped client, each client has at least t neighbors, otherwise the final sum cannot be recovered. According to the law of hypergeometric distribution [34], we choose $H = \log(N)$, that is, when each client has $\log(N)$ neighbor nodes, the effect is the perfect, which can not only ensure that there are enough neighbor nodes to satisfy its security, but also maximize the use of computing resources.

This paper adopts a network framework of multiple groups of regularized graph structures, assuming that there are N clients, each with an identifier $1, 2, 3 \dots n$. Firstly, divide all clients into $K = \frac{N}{\log(N)}$ groups, make sure the number of each group is $H = \log(N)$. Secondly, count the number of online client in each round of each group and record as D_i^k ($D_i^k \subseteq R$). Then our ultimate goal is to find the model parameter of the aggregation, namely $Z = \sum_{k \in [K]} \sum_{i \in D_i^k} x_i^k$, where x_i^k is the model parameter that needs to be uploaded to the server, as shown in Fig. 2.

In this protocol, K groups are executed simultaneously in a data-parallel manner. Client i in group K encodes its own input and sends it to the server. In case of disconnection of the client, a novel coding technology can be used to recover the disconnection information, while ensuring the privacy of the client and the robustness of client exit.

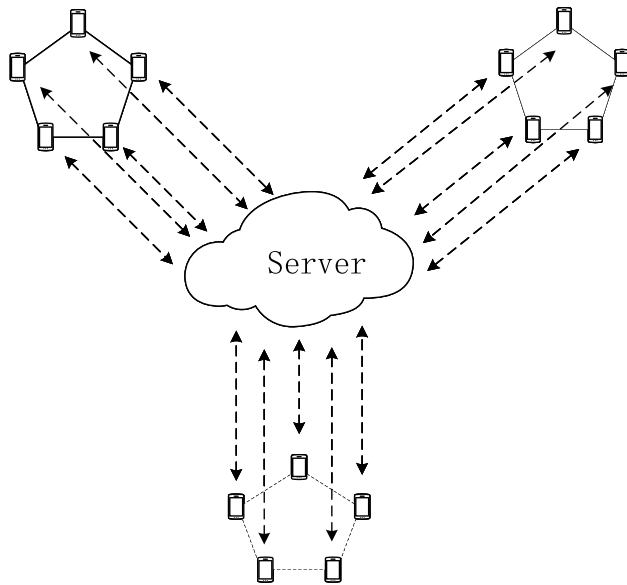


Fig. 2 Network topology with N clients partitioned into K groups, with H clients in group

The number of members of each group needs to be discussed. Since $H = \log(N)$ is based on 2, it is often not an integer, how to determine the number of each group? Let's take $N = 125$ as an example, so $\log_2 125 \approx 6.96$, here, if there are 6 client in each group, they will be divided into 20 groups, and the 21st group will be 11 client, however, at this time, the threshold value of the recovery phase is determined to be $t = 4$, and if the client in the 20th group is disconnected, the secret value cannot be obtained through the recovery phase. Therefore, after thinking and research, this paper found that it can be divided into two cases:

1. When $H = \lfloor \log(N) \rfloor$, then $k = \lceil K \rceil$ to determine that the secret value can be recovered.
2. When $H = \lceil \log N \rceil$, then the number of groups is reduced, but k still takes $\left\lceil \frac{N}{\log N} \right\rceil$ to ensure the security of the secret value.

4.4 Privacy guarantee by double mask

As we all know, data protection and sharing are a set of contradictions, and we need to mine the value of data on the premise of protecting data. Secure aggregation in federated learning can help organizations and industries separate ownership and usage rights, and use technical means to ensure that data will not be leaked and ownership will not change. This paper adheres to this concept and uses double masks to protect the right to use data to improve the privacy of graph neural networks. Specifically as follows:

$$y_i^k = x_i^k + r_i^k + \text{PRG}(p_i^k) \quad (15)$$

In this paper, double masks are used to guarantee privacy. ESA-FedGNN mainly includes two steps: (1) r_i^k is a random number sent by the server to the client to prevent the server from collusion with other clients to expose the client's input. (2) $\text{PRG}(p_i^k)$ is to deal with the client disconnection caused by network delay, equipment instability and other problems, which is conducive to large-scale client participation in collaborative training.

5 Correctness and security

In this section, we focus on the security of this protocol. Specifically, the correctness of the scheme is briefly analyzed at first, and then the security of the privacy of the client input data is further analyzed.

5.1 Correctness analysis

In this subsection, we mainly describe the correctness analysis of the proposed scheme. According to the analysis of the adversary in 3.1.1 subsection, we analyze whether the cryptographic techniques used in this paper meet the correctness requirements one by one.

First, for the construction of the regular graph, it requires that when the client is disconnected, there are still at least t neighbor nodes around each client to ensure that the original value can be restored. Second, in the protocol, the Shamir-based secret sharing requirement is divided into N shares that cannot be derived from each other, the original value can be obtained if and only if greater than or equal to $t + 1$ shares cooperate together, so when an attacker attacks any t shares, it is quite difficult in polynomial time. In the protocol, when the two communicating clients complete the Decision Diffie-Hellman (DDH) session exchange, they will obtain a shared secret, namely $\text{KA.agree}(sk_i, pk_j) = \text{KA.agree}(sk_j, pk_i)$. For correctness of authentication encryption, we require that for all keys $c \in \{0, 1\}^\lambda$ and all messages x $\text{AE.dec}(c, \text{AE} \cdot \text{enc}(c, x)) = x$. For correctness of reconstruction phase, as long as the subsecret share of t participants is obtained, the original value can be recovered by calculating Newton interpolation. All of the above methods demonstrate the correctness of this protocol.

5.2 Security analysis

This subsection mainly analyzes the protocol from a security perspective, analyzing its security in an honest and curious server environment. Specifically, the server is able to perform the correct operation for the specified protocol,

such as the aggregation protocol and update gradient. We also allow the server to collude with up to $N - 1$ clients for maximum attack power. Security analysis proves that this protocol can guarantee that the server and its colluding parties only know the final aggregation results, and cannot obtain the plaintext information of other honest clients.

For ease of description, we introduce some notation, assuming there are N clients, divide into K groups according to geometric distribution, and let H denote the number of each group. Since the protocol has a total of four rounds, each round will cause the client to drop due to network delay and other problems. Therefore, use $D_4^k \subseteq D_3^k \subseteq D_2^k \subseteq D_1^k \subseteq H$ to represent the number of online players in the k th group in each round, t^k is the threshold, that is, the minimum number of people online in each group, the set of all clients $E^k \subseteq H \cup \{S\}$ is represented as $REAL_A^{N,p,a,b}(X^k, D_1^k, D_2^k, D_3^k, D_4^k)$ under the real protocol.

Theorem 5.1 (Security against multi-client collusion). $\forall a$, security parameter b , adversary subset $A^k \subseteq H$ with $A^k < t$, uploaded model parameter set, so $t^k \subset D_4^k \subseteq D_3^k \subseteq D_2^k \subseteq D_1^k \subseteq H$. Here is a probabilistic polynomial time (PPT) simulator SIM_A , the output of SIM_A is completely indistinguishable from the real protocol $REAL_A^{H,a,b}$.

$$\begin{aligned} & REAL_A^{H,a,b}(x_i^k, D_1^k, D_2^k, D_3^k, D_4^k) \\ & \equiv \\ & SIM_A^{H,a,b}(x_A, D_1^k, D_2^k, D_3^k, D_4^k) \end{aligned} \quad (16)$$

Proof: The simulator can use the real protocol to interact with the adversary in four rounds, while for the honest client can use the virtual input, as follows: Specifically, in the second round, the honest participant sends the masked value y_i to the server. At this moment, the server first determines whether the number of messages received is greater than t . If it is greater than t , denote this set as D_3^K and sends it to the client. Where D_3^K contains a list of the user's identities, not the specific value of y_i , then the simulator can input the virtual value to the honest person in C_2 .

Theorem 5.2 (Honest But Curious Security, with curious server) $\forall a$, security parameter b , adversary subset $A^k \subseteq H$ with $A^k < t$, uploaded model parameter set, so $t^k \subset D_4^k \subseteq D_3^k \subseteq D_2^k \subseteq D_1^k \subseteq H$, $A \subseteq H \cup \{S\}$ with $A \setminus \{S\} < t^k$. Here is a probabilistic polynomial time (PPT)

simulator SIM_A , the output of SIM_A is completely indistinguishable from the real protocol $REAL_A^{H,a,b}$.

$$\begin{aligned} & REAL_A^{H,a,b}(x_i^k, D_1^k, D_2^k, D_3^k, D_4^k) \\ & \equiv \\ & SIM_A^{H,a,b}(x_A, \xi, D_1^k, D_2^k, D_3^k, D_4^k) \end{aligned} \quad (17)$$

Proof. The simulator can use the real protocol to interact with the adversary in four rounds, while for the honest client can use the virtual input. In detail, in the 1 round, SIM modifies honest users and leverage random numbers to replace the key pairs $KA.agree(sk_i, pk_i)$ generated between honest users, in the 2 round, SIM modifies the behavior of honest users by using encrypted random numbers instead of e_i shares sent to other users. Symmetric authentication encryption ensures that the modification is indistinguishable from the original protocol. Also in the first round, the SIM replaces the honest user-generated PRG with random numbers. In the 3 round, the SIM modified the way to encrypting the local gradient. Due to the security of PRG, this modification was guaranteed to be indistinguishable from the original protocol. In round 4, when $D_3^k < t^k$ the honest user is offline.

$$D^{*k} = \begin{cases} D_3^k \setminus A & \text{if } \xi = \perp \\ D_3^k \setminus D_4^k \setminus A & \text{otherwise} \end{cases} \quad (18)$$

As long as $A < t$, the true content of p_i^k can not be obtained obviously, then the security of secret sharing guarantees that the modification is indistinguishable from the original protocol. In general, the view generated by SIM is indistinguishable from the view under the real protocol $REAL_E^{H,t,k}$.

6 Evaluation

In this section, we mainly summarize the performance of the protocol, as shown in Table 1. We evaluate the version of the protocol as a honest-but-curious version, in which all calculations below assume a server and N clients, and each client holds a data vector size is L . We focus on reporting the computational cost and communication overhead of a single client and server when $H = O(\log N)$, we assume that the basic operations in Fq such as addition or multiplication are $O(1)$.

Table 1 Comparison of the proposed Fast-Aggregate with Secure Aggregation. Here N is the total number of clients and L is the size of model updates

Protocol	[13]	[34]	ESA-FedGNN
Computation (Server)	$O(LN^2)$	$O(N \log^2 N + LN \log N)$	$O(\log N \log \log N + LN \log N)$
Communication (Server)	$O(N^2 + LN)$	$O(N \log N + LN)$	$O(N \log N + LN)$
Computation (Client)	$O(N^2 + LN)$	$O(\log^2 N + L \log N)$	$O(\log N \log \log N + L \log N)$
Communication (Client)	$O(N + L)$	$O(\log N + L)$	$O(\log N + L)$

6.1 Performance analysis of client

Computation Cost at a client The computational overhead of each client consists of the following three parts:

1. Perform $2H$ key agreements and H encryptions ($O(\log N)$ complexity).
2. Create secret shares via Fast Fourier Transform(FFT) ($O(\log N \log(\log N))$ complexity).
3. To ensure privacy, generate a single mask ($O(L \log N)$ complexity).

Overall, each client computation cost is $O(L \log N + \log N \log(\log N))$.

Communication cost at a client The communication overhead of each client includes the following four parts in total:

1. Perform key agreement, including sending a public key, receiving $H - 1$ public keys ($O(\log N)$ complexity).
2. Send a share of encrypted model parameters ($O(\log N)$ complexity).
3. Send a masked input message of size L ($O(L)$ complexity).
4. Regain $2H$ shares ($O(\log N)$ complexity).

Overall, the communication complexity of each client is $O(\log N + L)$.

6.2 Performance analysis of client

Computation cost at the server The computing overhead of the server consists of the following two parts:

1. Use Newton interpolation to reconstruct the initial secret value of each client ($O(\log N \log(\log N))$ complexity).
2. Remove masks from clients and masks generated by the server itself ($O(LN \log N)$ complexity).

Overall, the server computation cost is $O(LN \log N + \log N \log(\log N))$.

Communication cost at the server The communication overhead of the server mainly depends on the pairwise communication between all clients, namely $O(N \log N + NL)$.

The existing security aggregation protocol [13] for federated learning leads to heavy computing and communication costs, which limits its scalability. See Table 1 for specific comparisons. The method proposed in this paper

not only reduces the communication overhead, but also greatly reduces the computational time complexity.

7 Conclusion

This paper mainly discusses the executable scheme of federated learning, secure aggregation and the coupling of neural networks, and proposes Efficient Secure Aggregation for Federated Graph Neural Network(ESA-FedGNN). Three main factors are proposed in protocol: secret sharing based on numerical analysis, regular graph embedding based on geometric distribution, privacy guarantee by double mask, and the three complement each other and jointly guarantee the federated graph neural network. In addition, we also analyze the privacy and security of this framework, and reduce the computational and communication overhead while ensuring the accuracy of the model.

Algorithm 1 ESA-FedGNN Protocol

- **Parties:** Clients $1, 2, \dots, N$ and the Server S .
- **Input:** x_i^k ($x_i^k \in Z_R$) (for each client i).
- **Output:** z ($z \in Z_R$) (for the server).
- **Round 1**
Server:
– Run $(G, t, D) = \text{RegularizedGraph}(N)$, where G is a regular graph that divides N nodes into $\lceil \frac{N}{D} \rceil$ groups, $N_G(i^k)$ is the set of neighbors of k th client i^k .
Client i^k :
– Generate public-private key pair $(pk_i^k, sk_i^k) \leftarrow KA \cdot \text{gen}(pp)$, and generate signature $\sigma_i^k \leftarrow \text{SIG} \cdot \text{sign}(b_{ki}^k, pk_i^k)$.
– Forwards $(pk_i^k \| \sigma_i^k)$ to $N_G(i^k)$ via the server.
Server:
– Collect at least t^k clients in the k th group (denote this set as D_1^k); otherwise, the protocol is terminated.
– Sends a random numbers r_i^k to each client in D_1^k .
– Send the list $\{(i, \sigma_i^k, pk_i^k)\}_{i \in D_1^k}$ to all clients in D_1^k .
- **Round 2**
Client i^k :
– Receive the list $\{(i, \sigma_i^k, pk_i^k)\}_{i \in D_1^k}$ from the server; verify that $D_1^k \geq t^k$, otherwise, the protocol is terminated. To prevent the server from forging clients to extract client data, then perform message verification on all clients in D_1^k , namely, $\forall i \in N_G(i^k), \text{SIG} \cdot \text{ver}(b_{ki}^k, pk_i^k, \sigma_i^k) = 1$.
– Generate a random PRG seed p_i^k .
– Compute shares of set by FFT algorithm: $P_i^k = \{p_{i,1}^k, p_{i,2}^k, \dots, p_{i,D}^k\} = \text{FFT} \cdot \text{share}(t, D, p_i^k)$.
– Compute encrypted shares $q_{i,j}^k \leftarrow \text{AE} \cdot \text{enc}(f_{i,j}^k, i^k \| j^k \| p_{i,j}^k)$ for each $j^k \in N_G(i^k)$, where $f_{i,j}^k = KA \cdot \text{Agree}(sk_i^k, pk_j^k)$. Using authenticated encryption technology for each client, the key used is the result of key negotiation between the two clients.
– Send all the ciphertexts $(j^k, q_{i,j}^k)$ to the server.
Server:
– Collect at least t^k clients in the k th group (denote this set as D_2^k); otherwise, the protocol is terminated.
– Forwards $(j^k, q_{i,j}^k)$ to clients j^k , who deduces $D_2^k \cap N_G(j^k)$ and move to the next round.

• Round 3

Client i^k :

- Collect ciphertext corresponding to other clients from the server side $(j^k, q_{i,j}^k)$, and get the set D_2^k ; if $D_2^k < t^k$, the algorithm is aborted immediately.
- Calculate the personal mask $PRG(p_i^k)$.
- Send to the server their masked input $y_i^k = x_i^k + r_i^k + PRG(p_i^k)$, if any operation is performed in the above process, the operation is terminated directly.

Server:

- Collect at least t^k clients in the k th group (denote this set as D_3^k); otherwise, the protocol is terminated.
- Send to every client in D_3^k the list D_3^k .

• Round 4

Client i^k :

- Decrypt the ciphertexts $(i_1^k \| j_1^k \| p_{j,i}^k) \leftarrow AE \cdot dec(KA \cdot Agree(sk_i^k, pk_i^k) \cdot q_{j,i}^k)$ and assert that $(i^k = i_1^k) \wedge (j^k = j_1^k)$.
- Send a list of shares to server, which contains $p_{i,j}^k$ for clients in D_3^k .

Server:

- Collect at least t^k clients in the k th group (denote this set as D_3^k); otherwise, the protocol is terminated.
- Collects, for each client $i^k \in D_3^k$, reconstruct p_i^k with all shares in P_i^k sent by clients in D_4^k by Newton interpolation.
- Compute and output $z = \sum_{k \in [K]} \sum_{i \in D_3^k} x_i^k$ as

$$\sum_{i \in D_3^k} x_i^k = \sum_{i \in D_3^k} y_i^k - \sum_{i \in D_3^k} PRG(p_i^k) - \sum_{i \in D_3^k} r_i^k$$

Author contributions Author Yanjun Liu (First Author): Conceptualization, Methodology, Software, Investigation, Formal Analysis, Writing; Author Hongwei Li (Corresponding Author): Conceptualization, Funding Acquisition, Resources, Supervision, Writing - Review; Author Xinyuan Qian: Investigation, Supervision, Writing - Review; Author Meng Hao: Resources, Supervision, Writing - Review.

Funding This work is supported by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B0101360001, National Natural Science Foundation of China under Grants 62020106013, 61972454, and 61802051, Sichuan Science and Technology Program under Grants 2020JDTD0007 and 2020YFG0298, the Fundamental Research Funds for Chinese Central Universities under Grant ZYGX2020ZB027.

Data availability The processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

Declarations

Ethics approval This article does not involve ethical issues (Not applicable).

Consent for publication I have read and understood the publishing policy, and submit this manuscript in accordance with this policy.

Conflict of interest I declare that the authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper. The results/data/figures in this manuscript have not been published elsewhere, nor are they under consideration (from you or one of your Contributing Authors) by another publisher.

References

1. Zhang H, Wu B, Yuan X, Pan S, Tong H, Pei J (2022) Trustworthy graph neural networks: Aspects, methods and trends
2. Dai E, Zhao T, Zhu H, Xu J, Guo Z, Liu H, Tang J, Wang S (2022) A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. arXiv e-prints
3. Wu C, Wu F, Cao Y, Huang Y, Xie X (2021) Fedgnn: Federated graph neural network for privacy-preserving recommendation. arXiv preprint [arXiv:2102.04925](https://arxiv.org/abs/2102.04925)
4. Sun M, Zhao S, Gilvary C, Elemento O, Zhou J, Wang F (2020) Graph convolutional networks for computational drug development and discovery. *Brief Bioinform* 21(3):919–935
5. Cui Z, Henrickson K, Ke R, Wang Y (2020) Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE* 11
6. Yuan H, Zheng J, Ye Q, Qian Y, Zhang Y (2021) Improving fake news detection with domain-adversarial and graph-attention neural network. *Decis Support Syst* 151
7. Jin W, Stokes JM, Eastman RT, Itkin Z, Zakharov AV, Collins JJ, Jaakkola TS, Barzilay R (2021) Deep learning identifies synergistic drug combinations for treating covid-19. *Proc Natl Acad Sci* 118(39):2105070118
8. Kairouz P, McMahan HB, Avenet B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R et al (2021) Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14(1–2):1–210
9. Xu J, Glicksberg BS, Su C, Walker P, Bian J, Wang F (2021) Federated learning for healthcare informatics. *J Healthc Inform Res* 5(1):1–19
10. Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: Challenges, methods, and future directions. *IEEE Signal Process Mag* 37(3):50–60
11. Lyu L, Yu H, Ma X, Chen C, Sun L, Zhao J, Yang Q, Philip SY (2022) Privacy and robustness in federated learning: Attacks and defenses. *IEEE Trans Neural Netw Learn Syst*
12. He X, Jia J, Backes M, Gong NZ, Zhang Y (2021) Stealing links from graph neural networks. In: 30th USENIX Security Symposium (USENIX Security 21), pp 2669–2686
13. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K (2017) Practical secure aggregation for privacy-preserving machine learning. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp 1175–1191
14. Zhao Y, Zhao J, Yang M, Wang T, Wang N, Lyu L, Niyato D, Lam K-Y (2020) Local differential privacy-based federated learning for internet of things. *IEEE Internet Things J* 8(11):8836–8853
15. Fallah A, Mokhtari A, Ozdaglar A (2020) Personalized federated learning: A meta-learning approach. arXiv preprint [arXiv:2002.07948](https://arxiv.org/abs/2002.07948)
16. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: *ACM*, pp 169–178
17. Liu C, Chakraborty S, Verma D (2019) Secure model fusion for distributed learning using partial homomorphic encryption. In: *Policy-Based Autonomic Data Governance*, pp 154–179. Springer
18. Yao AC (1982) Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), pp 160–164. *IEEE*
19. Mohassel P, Rindal P (2018) ABy3: A mixed protocol framework for machine learning. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp 35–52
20. So J, Güler B, Avestimehr AS (2021) Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE J Sel Areas Inf Theory* 2(1):479–489
21. He C, Balasubramanian K, Ceyani E, Yang C, Xie H, Sun L, He L, Yang L, Philip SY, Rong Y et al (2021) Fedgraphnn: A federated learning benchmark system for graph neural networks

22. Zhang K, Yang C, Li X, Sun L, Yiu SM (2021) Subgraph federated learning with missing neighbor generation. *Adv Neural Inf Proces Syst* 34:6671–6682
23. Chen C, Hu W, Xu Z, Zheng Z (2021) Fedgl: federated graph learning framework with global self-supervision. *arXiv preprint arXiv:2105.03170*
24. Zheng L, Zhou J, Chen C, Wu B, Wang L, Zhang B (2021) Asfgnn: Automated separated-federated graph neural network. *Peer Peer Netw Appl* 14(3):1692–1704
25. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. In: *International Conference on Machine Learning*, pp 1263–1272 PMLR
26. Tompa M, Woll H (1989) How to share a secret with cheaters. *J Cryptol* 1(3):133–138
27. Diffie W, Hellman ME (2022) New directions in cryptography. In: *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pp 365–390
28. Blum M, Micali S (2019) How to generate cryptographically strong sequences of pseudo random bits. In: *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pp 227–240
29. Bellare M, Namprempre C (2008) Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J Cryptol* 21(4):469–491
30. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. *Adv Neural Inf Proces Syst* 30
31. Yang Q, Liu Y, Chen T, Tong Y (2019) Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10(2):1–19
32. Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) *Introduction to Algorithms*, 3rd. The MIT Press
33. Berrut J-P, Trefethen LN (2004) Barycentric lagrange interpolation. *SIAM review* 46(3):501–517
34. Bell JH, Bonawitz KA, Gascón A, Lepoint T, Raykova M (2020) Secure single-server aggregation with (poly) logarithmic overhead. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp 1253–1269

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

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



Yanjun Liu received M.S. degree in computer technology from Chong Qing University of Posts and Telecommunication. She is currently pursuing the Ph.D. degree in School of Computer Science and Engineering, University of Electronic Science and Technology of China(UESTC). Her research interests include applied cryptography and privacy-preserving machine learning.



Hongwei Li is currently the Associate Dean at School of Computer Science and Engineering, University of Electronic Science and Technology of China. He received the Ph.D. degree from University of Electronic Science and Technology of China in June 2008. He worked as a Postdoctoral Fellow at the University of Waterloo from October 2011 to October 2012. His research interests include network security and applied cryptography. Dr. Li has published more than 100 technical papers. He is the sole author of a book, *Enabling Secure and Privacy Preserving Communications in Smart Grids* (Springer,

2014). Dr. Li serves as the Associate Editors of *IEEE Internet of Things Journal*, and *Peer-to-Peer Networking and Applications*, the Guest Editors of *IEEE Network*, *IEEE Internet of Things Journal* and *IEEE Transactions on Vehicular Technology*. He also served the Technical Symposium Co-chairs of *IEEE ICC 2022*, *ACM TUR-C 2019*, *IEEE ICC 2016*, *IEEE GLOBECOM 2015* and *IEEE BigDataService 2015*, and Technical Program Committees for many international conferences, such as *IEEE*

INFOCOM, IEEE ICC, IEEE GLOBECOM, IEEE WCNC, IEEE Smart-GridComm, BODYNETS and IEEE DASC. He won Best Paper Awards from IEEE ICPADS 2020, IEEE MASS 2018 and IEEE HEALTHCOM 2015. Dr. Li currently serves as the Vice Chair(publication) of IEEE Com-Soc CIS-TC. He is the Senior Member of IEEE and the Distinguished Lecturer of IEEE Vehicular Technology Society.



Xinyuan Qian received B. Eng. degree in security engineering in People's Public Security University of China in 2018 and M. Eng. degree in computer technology in University of Chinese Academy of Sciences in 2021. He is currently pursuing the Ph.D. degree in School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include IBE, ABE, FE, FHE, applied cryptography, and privacy-preserving machine learning.



Meng Hao received the B.S. degree in information security in 2018, from the University of Electronic Science and Technology of China (UESTC). He is currently working toward the Ph.D. degree in cyber security at UESTC. His research interests include applied cryptography and privacy-preserving deep learning.