

A Survey of Security Aggregation

Sufang Zhou*, Mingzhe Liao*, Baojun Qiao*, Xiaobo Yang**

*School of Computer and Information Engineering, Henan University, Kaifeng 475004, China

**China United Network Communication Corporation Henan Branch, Zhengzhou 450003, China

zsf@henu.edu.cn, 326995871@qq.com, qiaobaojun2009@163.com, yangxb72@chinaunicom.cn

Corresponding author: qiaobaojun2009@163.com

Abstract— Machine learning (ML) requires the collection of large amounts of data to train robust predictive models. Federated learning (FL) allows sensitive data to be kept on the client side to train a shared model, but shared models still pose privacy concerns. Secure aggregation is an important algorithm for securing federation learning by being able to compute the sum of client-side model updates without revealing information about individual client updates. In this study, we investigate the results of secure aggregation protocols in recent years and review the research results in the field of secure aggregation according to secret sharing, differential privacy, and homomorphic encryption mechanisms. On this basis, we compare and analyze the advantages and disadvantages of different mechanisms and then evaluate the security aggregation protocols in terms of security, dynamic user robustness, computational cost, and communication cost. Finally, we provide an outlook on the future development of secure aggregation protocols and give possible future research directions.

Keywords— Federated Learning, Security Aggregation, Secret Sharing, Differential Privacy, Homomorphic Encryption

I. INTRODUCTION

With the explosive growth of data, it is of great strategic importance to analyze and mine the potential value of data. Machine learning (ML)[1] is an important tool for enabling data analysis and mining. However, data often contains sensitive information. Therefore the collection of sensitive data poses significant risks and is often prohibited. Thus preventing the value of large amounts of data from being realized.

Federated learning (FL)[2] is a field of training ML model that benefits from multiple datasets while keeping the training data on the client side. FL algorithms typically iterate training rounds, during which model updates are typically aggregated at the server after local training. This approach holds great promise for leveraging the value of data while protecting data privacy, but there are still two main challenges to the large-scale application of federation learning techniques in the real world: privacy concerns and expensive communication. While keeping data in the client is a significant improvement for data privacy, sharing sensitive information that is indirectly leaked by intermediate models. Secure aggregation can be used to address the privacy concerns of FL, which enables FL servers to compute the sum of client-side model updates without the

need to know information about any individual client-side updates.

The problem of secure aggregation has been an area of research with different approaches including works based on secret sharing (SS)[3], differential privacy (DP)[4], homomorphic encryption (HE)[5], etc. In recent years, a large number of secure aggregation protocols have been designed to provide strong privacy guarantees in FL. Secure aggregation for secure computational sums has received a great deal of research in the last few years. However, FL poses some unique challenges for secure aggregation protocols.

(1) Huge user size: up to 10,000 users can be selected to participate in the iteration.

(2) Higher communication costs: model updates may be high-dimensional vectors, e.g. ResNet models consist of hundreds of millions of parameters.

(3) User dropout: The client is a mobile device that can exit at any time. The exit rate can be up to 10%.

(4) Privacy: since malicious servers can launch powerful attacks through the use of sybils[3], it is crucial to provide the strongest possible privacy protection, in which the adversary simulates a large number of fake client devices.

In addition, secure aggregation itself has issues that need to be addressed: high computational volumes, high communication volumes, and security against semi-honest and malicious attackers, among others.

To address the above challenges, experts have proposed many security aggregation protocols for one of these challenges or for multiple challenges at the same time, as well as many optimizations for various application scenarios. In 2017 Keith Bonawitz et al.[7] worked out a practical privacy-preserving machine learning security aggregation method, which proposes a secret sharing and its innovative double-mask structure based on a secure aggregation protocol for FL that not only protects against semi-honest and malicious attackers but also is robust to users' mid-way dropout. After this study was published, many scholars joined the study of secure aggregation for FL and proposed many innovative secure aggregation protocols, although their performance makes them unsuitable for most practical applications. However, they have created a very active research area in the design and implementation of secure aggregation protocols. As a result of these efforts, the performance and usability of secure aggregation protocols have steadily improved.

In this paper, we broadly evaluate some secure aggregation protocols and discuss them categorically by technique in Section II. This is followed by a description of secure aggregation protocols based on secret sharing, differential privacy, and homomorphic encryption in sections III, IV, and V, respectively. In Section VI, the future development of secure aggregation protocols is given an outlook and possible future research directions are proposed.

II. SECURE AGGREGATION PROTOCOL

We investigate the most advanced security aggregation protocols in recent years. We mainly investigate security aggregation protocols that have been proposed one after another after Keith Bonawitz et al.[7] proposed SecAgg in 2017. These protocols all follow the following general flow.

1) Preparation Phase: Establishment of communication keys between server and user and between user and user, and establishment of keys for the cryptographic primitives that the protocol will subsequently need to use.

2) User Processed Data: The user processes his data, the processed data will not express any of the original information and the processed data will support the server for aggregation. The user sends the processed data to the server.

3) Server Aggregates the Data: The server aggregates all the received data that has been processed to get the sum of the original data.

A. Basic Information on Security Aggregation Protocols

The basic information of these protocols is shown in TABLE 3 which includes semi-honest security, malicious security[8], user dropout robustness, computational cost, communication cost, number of communication rounds, and some other features.

Semi-honest security is arguably one of the most essential conditions for judging a secure aggregation protocol to be practical. The secure aggregation protocols we investigated all guarantee semi-honest security well (all provide proofs of semi-honest security in their papers). Malicious security is a stronger security guarantee than semi-honest security. In practice (and in the security aggregation protocols we investigated), security against malicious adversaries is implemented as a malicious abort scheme, where if malicious activity is detected, the protocol will abort and the malicious adversaries will not cause a wrong answer nor gain additional information but may cause the protocol to abort resulting in no answer at all. Only SecAgg[7], SAFER[13] and SecAgg+[9] of the security aggregation protocols we investigated can implement a full malicious abort scheme (where both the server and the user can be malicious attackers), while Hybridalpha[14] can only implement a malicious abort scheme where the malicious attacker is only the user. In existing security aggregation schemes, the cost of guaranteeing malicious security is very high and is generally accompanied by high computational communication costs and communication rounds. SecAgg[7], for example, requires not only an additional signature of the data using a digital

signature and an additional round of operations added to the protocol but also a trusted third-party dispatch key to guarantee malicious security. Computational cost, communication cost, and the number of communication rounds is the most intuitive elements to measure the practicality of a protocol. Improving security, functionality, and robustness while keeping costs within acceptable limits is the most immediate goal in the development of secure aggregation protocols. Also finding the best balance between security and cost is a very important research element. The unacceptable growth method of cost with the number of users, data volume, etc. is one of the biggest obstacles to move towards practical applications of secure aggregation protocols today. Storage costs are not as important as they used to be (compared to computational and communication costs) due to the rapid advances in storage technology today.

B. Security Aggregation Protocols Classified by Technology

We broadly classify these secure aggregation protocols into three categories based on the techniques they use: Secret Sharing-based protocols, Differential Privacy-based protocols, and Homomorphic Encryption-based protocols. The classification is shown in TABLE 1. The advantages and disadvantages of the three techniques are statistically presented in TABLE 2.

TABLE 1. CLASSIFIED BY TECHNOLOGY

Use of Technology	Protocol
Secret Sharing	SecAgg[7], SecAgg+[9], SAFElearning[11], Fastsecagg[12], SAFER[13]
Differential Privacy	Hybridalpha[14], DPFL[15], Dopamine[16]
Homomorphic Encryption	PrivFL[10], SAFElearning[11], Batchcrypt[17], POSEIDON[18]

TABLE 2. THE ADVANTAGES AND DISADVANTAGES OF THE THREE TECHNIQUES

Technology	Cost	No Loss of Accuracy	no Redesign Required	Dropout Robustness
Secret Sharing	medium	✓	×	✓
Differential Privacy	low	×	✓	×
Homomorphic Encryption	high	✓	✓	×

III. SECURE AGGREGATION PROTOCOLS BASED ON SS

The two properties of secret sharing-based secure aggregation protocols that are robust to user dropout and acceptable overhead make it suitable for cross-device federation learning (where the client is a large number of mobile or IoT devices with limited computational power and unreliable communication). Additive secret sharing is also used in SAFER[13] in combination with neural network update compression techniques. Some secure aggregation protocols based on secret sharing are described below.

A. SecAgg

SecAgg[7] is a secure aggregation protocol based on a dual mask structure and a threshold sharing scheme and is mainly considered for implementation on mobile devices. The authors

TABLE 3. BASIC INFORMATION ON SECURITY AGGREGATION PROTOCOLS. IN THE COLUMN OF USER DROPOUT ROBUSTNESS, THE WORST-CASE DROPOUT IS CONSIDERED UNLESS OTHERWISE SPECIFIED. IN THE COST COLUMN, C: CLIENT, S: SERVER, L: LINEAR REGRESSION MODEL, G: LOGISTIC REGRESSION MODEL, M: DATA SIZE, N: NUMBER OF CLIENTS, S: NUMBER OF SERVERS, D: NUMBER OF DATA. A NON FIXED AGGREGATION PROTOCOL IS NESTED IN PRIVFL, SO THE NUMBER OF COMMUNICATION ROUNDS OF THE NESTED AGGREGATION PROTOCOL NEEDS TO BE ADDED TO THE NUMBER OF COMMUNICATION ROUNDS OF PRIVFL.

Secure Aggregation Protocol	Semi-Honest Security	Malicious Security	User Dropout Robustness	Computation Cost		Communications Cost		Rounds	Other
SecAgg[7]	√	√(extra round)	√	$CO(n^2+mn)$	$SO(mn^2)$	$CO(n+m)$	$SO(n^2+mn)$	4	\
SecAgg+[9]	√	√	√(random dropout)	$CO(m\log n+\log^2 n)$	$SO(mn\log n+\log^2 n)$	$CO(m+\log n)$	$SO(mn+\log n)$	3	\
PrivFL[10]	√	×	√	$CO(R(m^2+mn+nd))$	$SO(R(m^2+mn+nd))$	$CLO(R(m+n))$ $CGO(R(m+n+d))$	$SLO(R(m^2mn))$ $SGO(m^2+Rm(n+d))$	2 + aggregation protocol rounds	protecting model privacy
SAFELearn[11]	√	×	√	$CO(mn)$	$SO(mn)$	$CO(m)$	$SO(m)$	2	at least 2 non-colluding servers
Fastsecagg[12]	√	×	√(random dropout)	$CO(m-\log n)$	$SO(m-\log n)$	$CO(m+n)$	$SO(mn+n^2)$	3	\
SAFER[13]	√	√	×	only additions		$2snd, \lceil \log(2n+1) \rceil + 2sn-32(\text{minimum})$		3	\
Hybridalpha[14]	√	√(only clients)	√(dynamic user)	$CO(m)$	$SO(nm)$	$CO(m+n)$	$SO(nm+n)$	1 (excluding key distribution)	need for trusted TPA

propose two variants of the protocol to facilitate balancing cost and security: one is more efficient and can be shown to be secure against honest but curious adversaries in the common model. The other guarantees privacy against malicious attackers (including malicious servers), but requires an extra round of consistency checks and sign and verify some of these communications using a third-party signing key and a verifying public key distributed by a third-party organization.

1) User Processing Data: Users process data using a double mask structure. Each user calculates y_u , and sends it to the server:

$$y_u = x_u + \text{PRG}(b_u) + \sum_{v \in U: u < v} \text{PRG}(s_{u,v}) - \sum_{v \in U: u > v} \text{PRG}(s_{v,u}) \pmod{R}$$

where x_u is the data of user u ; U is the set of users. $s_{u,v}$ is the shared secret value (unknown to the server) for user u and user v . Each user generates a random sample of b_u and sends their share of b_u secrets to all other users.

2) Server Aggregated Data: During recovery, the server requests the secret share of the b_u of all online users and the secret share of the keys of all offline users from online users. As long as a user less than the threshold provides the server with both the secret share of a user's b_u and the secret share of the key, the server is unable to obtain x_u . After the server recovers b_u for all online users and subtracts b_u from y_u , it can next compute the aggregated information z :

$$z = \sum_{u \in U} y_u = \sum_{u \in U} \left(x_u + \sum_{v \in U: u < v} s_{u,v} - \sum_{v \in U: u > v} s_{v,u} \right) = \sum_{u \in U} x_u \pmod{R}$$

B. SecAgg+

SecAgg+[9] reduces the computation and communication overhead by replacing the original SecAgg[7] protocol of "one client communicates with the remaining $n-1$ clients" with "one client communicates with the surrounding $\log n$ clients". SecAgg+ also proposes two protocols for semi-honest attackers and malicious attackers.

In semi-honest protocol, SecAgg+ builds an undirected graph suitable for the protocol by undirected graph building techniques using the user as the vertex, connecting each user with $k = \log n$ bits of other users, and when sharing the user only has to share with the user connected to him. users. k and the secret sharing threshold t are determined by ensuring security and user dropout robustness. In Semi-malicious

protocol, the overall idea is to first help the individual clients exchange keys, then generate a directed graph (both users are trusted unilaterally with each other), and finally execute the summation algorithm. When exchanging keys each client generates 2 pairs of keys, sends public key 1 and public key 2 to the server, and the server puts the received public key on top of the Merkle tree.

C. Fastsecagg

FastShare is a novel secret sharing scheme based on a finite domain version of the Fast Fourier Transform (FFT) that aims to exploit the federation learning setting. Fastsecagg[12], on the other hand, is precisely a FastShare-based and scalable FL-secure aggregation protocol with low computational and communication costs and strong privacy guarantees.

1) FastShare: ① Construct the signal: Find two numbers, n_0 and n_1 such that signal length $N = n_0 n_1$, and n_0 and n_1 are mutually prime. Applying the Chinese remainder theorem to establish a two-dimensional mesh of $n \times n$. Set four domains Z_0 , Z_1 , S , and T . A secret is placed at index S and T is used as privacy proof. Random masks are placed in the other regions; 0 is placed at Z_0 and at the index of Z_1 such that the sum of each row and column is satisfied to equal 0 (to subsequently recover lost shares). ② Generate the secret share: Use the (fast) Fourier transform to generate secret shares. ③ Recover lost shares: Iteratively recover all lost shares using a linear relationship between shares. Recover the missing share by iteratively decoding the missing share in rows and columns. ④ Inverse Fourier transform: the secret share is inverse Fourier transformed to extract the secret information at the appropriate location to recover the secret. ⑤ Linear characterization: if $\{(i, [s_1]_i)\}_{i \in \mathbb{C}}$ and $\{(i, [s_2]_i)\}_{i \in \mathbb{C}}$ are shared by s_1 and s_2 , then $\{(i, a[s_1]_i + b[s_2]_i)\}_{i \in \mathbb{C}}$ is shared by $as_1 + bs_2$ for any $a, b \in \mathbb{F}_q$.

2) Fastsecagg: Using the linear nature of FastShare, in the Fastsecagg protocol, users first share the secret shares of their messages with each other, and then add the secret shares of their messages to the secret shares shared over by others to arrive at the summed secret share S_{all} of all users. Users send their S_{all} to the server, and the server restores the secret. This ensures the privacy of each user's information. Robustness to user dropouts can also be achieved by recovering lost shares in FastShare.

D. SAFER

SAFER[13] combines neural network update compression techniques with additive secret sharing making neural update compression techniques compatible with secure multi-party computation, obtaining similar accuracy to non-secure uncompressed federation learning at lower communication cost and it allows a good trade-off between privacy and communication cost depending on the level of security required for each practical use case.

1) Additive Secret Sharing[19]: In this protocol, each client has a vector $X_i \in \mathbb{Z}_m^n$ and wants to know the sum of all client vectors, but no client wants to reveal its vectors to the other parties. The protocol is secure against semi-honest adversaries and private against malicious adversaries that control up to $S-1$ servers and an arbitrary number of clients.

2) SAFER: $\odot$ TopBinary Coding: the authors' approach is based on compressing the updates sent from the client to the aggregation server, with error compensation to ensure convergence. For the compression method, a combination of Top-k sparsification, as well as one-bit quantization, is used. The whole process is referred to here as TopBinary Coding. $\odot$ Separate Aggregation: use separate aggregation to avoid multiplicative safety calculations:

$$U = \frac{1}{C^2} \left(\sum_{i=1}^C \alpha_i \right) \left(\sum_{i=1}^C D_i \right) \quad (\text{SepAgg})$$

where α_i and D_i are calculated from TopBinary Coding and C is the number of users. The aggregation part of the SAFER protocol can be described in three steps: Secure Union Protocols, Secure Sum of Signs, and Secure Sum of Factors. Where the Secure Union Protocols authors give four methods for different cases, which is conducive to the trade-off between security and efficiency.

E. SAFElearn

SAFElearn[11] is a generic design for FL-safe aggregation that can accommodate a variety of security and efficiency requirements as well as multiple aggregation mechanisms. It is instantiated in the paper using Secure Two-Party Computation (STPC).

1) SAFElearn: SAFElearn can be implemented using Full HE, Multi-Party Computation (MPC), or STPC, depending on the efficiency and security requirements of the specific application. The scheme using full HE is described in Section IV. The complete flow of the SAFElearn algorithm is shown in Algorithm 1.

Algorithm 1 SAFElearn

```

1: Input:  $K, G_0, T \triangleright K$ : set of clients,  $G_0$ : initial global model,
    $T$ : # iterations
2: Output:  $G_T \triangleright G_T$  is the global model after  $T$  iterations
3: for each training iteration  $t \in [1, T]$  do
4:   for each client  $i \in K_t \subseteq K$  do  $\triangleright K_t \subseteq K$  is randomly
     chosen in every iteration.
5:      $[W_i] \leftarrow \text{CLIENTUPDATE}([G_{t-1}])$ 
6:   end for
7:    $[G_t] \leftarrow \sum_{i=1}^{|K_t|} [W_i] / |K_t|$ 
8: end for

```

2) Use of MPC or STPC: $[G_t]$ and $[W_i]$ are secret shares created using secret sharing techniques and held by $N \geq 2$ non-

complicit servers. They jointly run secure aggregation on these shares (i.e., N servers together form an aggregator). After the secure aggregation, the servers send the secret shares of the new global model to the client, which combines them to receive the explicit global model for the next local training iteration. In this secure aggregation, the segmentation is implemented using a Boolean circuit-based protocol called Yao's garbled circuit for STPC or BMR for MPC, said protocol can efficiently compute nonlinear operations. Afterward, they again share updates secretly and send a share to each server.

IV. SECURE AGGREGATION PROTOCOLS BASED ON DP

Although DP makes learning accuracy degradation and model disclosure, this privacy violation and potential accuracy degradation may be tolerable for mobile users of cross-device federation learning. Therefore, like secret-sharing-based secure aggregation protocols, DP-based secure aggregation protocols are more suitable for cross-device federation learning, and it has the advantage of compatibility and low overhead over secret-sharing-based secure aggregation protocols. Moreover, DP can easily be mixed with other techniques to form hybrid protocols (e.g., Hybridalpha[14], which will be described below). Some DP-based secure aggregation protocols are described below.

A. DPFL

DPFL[15] is a federated optimization algorithm for client-side DP preservation. It aims to hide the contribution of clients during training and balance the trade-off between privacy loss and model performance. Empirical studies show that when the number of participating clients is sufficiently large, the protocol proposed in the paper also requires only a small cost in terms of model performance to maintain DP at the client level.

In the FL Framework, the server averages the client model (i.e., the weight matrix) after each round of communication. In DPFL, on the other hand, a random mechanism will be used to change and approximate this averaging. This is done to hide the contribution of individual clients in the set, and thus in the overall distributed federation learning process.

B. Dopamine

Dopamine[16] is a FL algorithm with DP for hospital patient sensitive data (medical image data), where the local client training part uses the stochastic gradient descent with differential privacy algorithm (SGD-DP) to make the model converge, and the global model is updated on the server side using secure aggregation (averaging the gradients of all clients). Dopamine can be used in the medical field where model accuracy and privacy are sought, by increasing the communication cost to improve accuracy and privacy.

Protocol Process: (1) For the global gradient do random initialization; (2) For the dataset, slice the dataset into copies of the number of clients (e.g., 10 clients, then the dataset is sliced into 10 unduplicated copies); (3) For each hospital model, i.e., each client model, update the model using SGD-DP, adding noise using a Gaussian mechanism with variance

size $(\sigma^2 C^2 I)/K$ (I is the unit matrix, K is the number of clients); (4) wait for all clients' models to be updated to find the overall privacy budget; (5) de-average the gradient after all clients' updates to obtain the global gradient.

C. Hybridalpha

Hybridalpha[14] employs a DP mechanism and defines an SMC protocol from a multi-input function encryption scheme. This multi-function encryption scheme is also adapted to including additional restrictions to reduce the risk of curious aggregators and colluding participants inferring private information. The protocol also addresses the dynamic participant problem.

1) DP in SMC: A common DP mechanism is to add noise, where a normally distributed noise with a mean of 0 and a specified standard deviation is added to each input. The noise is reduced by SMC. Considering n participants, the total noise adds up to n times the noise. However, when SMC is applied to calculate the mean, each participant can add $1/n$ copies of the noise and then share the values used for aggregation using the SMC technique. This ensures the same DP guarantee while reducing the total noise injected by a factor of n .

2) Multiple Input Function Encryption: To satisfy the distributed setting, a variant of function encryption is used, namely Multiple Input Function Encryption[20], where the function key sk_f can correspond to an n -element function $f(-)$ that takes multiple ciphertexts as input. The authors have made some minor modifications to MIFE by adding an additional step, PKDistribute, to help pass the public key pk to the participants. As a result, each participant will use their pk to encrypt their data, instead of using the master key msk as originally. This modification is beneficial because different parties do not need to share a single master key.

3) Hybridalpha: TPA generates master key msk and master public key mpk and generates public key pk_i for each participant P_i and sends it to participant P_i . Participant P_i uses the data D_i to train to get the model update parameter M_i then performs DP operation to add noise to get M_i^{DP} and uses pk_i to encrypt M_i^{DP} to get $msg_{r,i}$ and finally sends $msg_{r,i}$ to the server. The server computes the weighted vector w_p based on the received msg and uses w_p to request the private key $sk_{f,w}$ from TPA, and then uses $sk_{f,w}$ to decrypt the user data sum.

4) Dynamic User Problem: When a user joins in the middle of the process, he only needs to request a key from the TPA, and the server only needs to adjust w_p according to the amount of data received at the end when aggregating messages.

V. SECURE AGGREGATION PROTOCOLS BASED ON HE

HE-based secure aggregation protocols are more suitable for Cross-Silo federation learning than the other two secure aggregation protocols for two reasons: (1) Because Cross-Silo federation learning has fewer clients, it is vulnerable to inference attacks on structural models. Whereas, in a secure aggregation protocol based on HE, the model is private. (2)

HE has a high overhead, while the clients in Cross-Silo federation learning (federation learning where the clients are few organizations (e.g., financial and medical) with reliable communication and abundant computational resources) have stable communication and abundant computational resources and can afford the overhead of HE to some extent. Some secure aggregation protocols based on HE are described below.

A. PrivFL

PrivFL[10] combined aggregation methods and additive homomorphic oriented linear regression and logistic regression to propose a secure aggregation protocol that guarantees both user data privacy and ML model privacy and provides a prediction service for users.

1) Regression Training: The server performs additive HE on the model θ to obtain $E(\theta)$ and sends $E(\theta)$ to user P_i , each user derives $E(s_i)$ and r_i by SLG algorithm with $E(\theta)$ and D_i (data of user p_i) as input such that $s_i - r_i = \omega_i$ (ω_i is the local gradient). The server then decrypts the received $E(s_i)$ and computes the s_i sum for all users, followed by using the aggregation protocol to solve for r_i as input and output the r_i sum for all users to the server. Finally, the server can then compute ω and update θ after:

$$\omega = \sum_{u \in U_a} s_u - \sum_{u \in U_a} r_u$$

where U_a is the set of users. The process is represented in Figure 1.

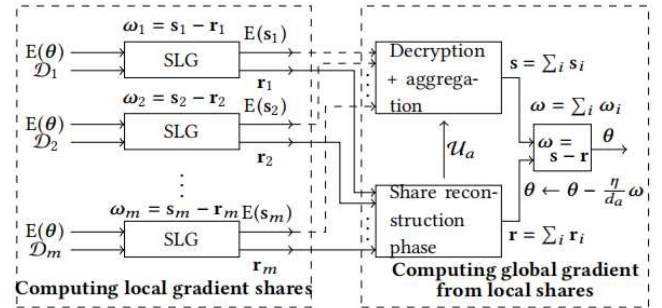


Figure 1. PrivFL's protocol flow of privacy-preserving global gradient computation for regression models

2) Regression Prediction: Each user creates a public and private key pair for an additive HE scheme, denoted (pk_{P_i}, sk_{P_i}) , and whose public key is known to the server. The user sends the encrypted input to the server. For linear regression, the server computes $Epk_{P_i}(\theta - x)$ from $Epk_{P_i}(X)$ and θ and sends it to the user, who decrypts it to obtain the prediction.

B. BatchCrypt

BatchCrypt[17] solves the encryption and communication bottleneck created by HE using a simple batch encryption technique. That is, the client first quantizes its gradient values into a low-bit integer representation. It then encodes the batch of quantized values into a long integer and encrypts them in a single pass. Compared to encrypting individual gradient values with full precision, batch encryption significantly reduces the encryption overhead and data transfer volume.

There are two key technical challenges in implementing this technique.

1) The First Challenge: The decryption result remains unchanged. In BatchCrypt, a custom quantization scheme is designed to quantize gradient values to signed integers that are uniformly distributed in the symmetric range. In addition, to support simple additive forms of gradient aggregation and to add gradients that do not cause overflow to corrupt the encoding, they developed a new batch encoding scheme that uses two signed bits to represent the complementary representation of two of the quantized values. In addition, padding and advance scaling are used to avoid overflow. All these techniques allow performing gradient aggregation on the ciphertext of an encoded batch without first decrypting it.

2) Second Challenge: Choosing the threshold for clipping. They propose an efficient analytical model, dACIQ, by extending ACIQ[21], a state-of-the-art shearing technique for centralized data, to Cross-SiloFL for decentralized data. dACIQ allows the selection of the optimal shearing threshold with minimum cumulative error.

C. POSEIDON

The paper extends SPINDLE (Privacy-Preserving Training and Evaluation for Generalized Linear Models) by building a new system (named POSEIDON[18]) that enables privacy-preserving training and evaluation of more complex neural network models in a distributed setting through multiparty HE, providing end-to-end protection: training data for all parties, the resulting model, and evaluation data for the querier. The weights of the neural network models are always encrypted during training and evaluation.

D. SAFElearn

Having already described the use of secret sharing to implement SAFElearn in Part V, we now describe the use of HE to implement SAFElearn.

Using a server with a single semi-honest server (acting as an aggregator), $[G_i]$ and $[W_i]$ are the results of HE of the model parameters. The clients use a multi-party encryption scheme to encrypt their updates. The HE keys are securely assigned to the clients to decrypt and access the global updates in plaintext for local training. The clients then re-encrypt the generated local model updates and return them to the server, which uses the additive homomorphic property of the encryption scheme to aggregate the encrypted data into a new global model $[G_i]$. This full HE is computationally intensive and should only be used if a non-complicit server cannot be used due to application and security requirements.

VI. DISCUSSIONS

A. Cost

Cost reduction is one of the most important research directions for secure aggregation protocols, especially in the context that secure aggregation protocols are so far too expensive to be practically applied, every secure aggregation protocol researcher is racking their brains to reduce the

protocol cost. Research on more efficient security aggregation protocols is the most direct research direction.

B. Flexibility in Safety and Efficiency Trade-Offs

Trade-offs between security and efficiency have always been a focus of cryptographic protocols. Being able to strike a good balance with various cost levels based on the security required for each practical use case helps improve the utility and flexibility of security aggregation protocols. Fixed security and cost combinations in protocols often tend to result in a few scenarios that can be used in practice. SecAgg[7] and SAFER split the protocol into two and four versions, improving flexibility and balancing security and cost when the protocol is used in practice, but this is still not enough for practical applications. SecAgg+[9], although it is also similar to SecAgg+ is also divided into two versions like SecAgg, but SecAgg+ can calculate the appropriate k (number of neighboring users per user) to adjust the cost according to the security requirements, which greatly improves the flexibility of SecAgg+.

The flexibility of a secure aggregation protocol is a very important property. Flexibility can also be a research point for a secure aggregation protocol. We hope that more flexible security aggregation protocols will emerge in the future.

C. User Dropout Robustness

We note that all current secure aggregation protocols that we are aware of use threshold sharing schemes to solve the user dropout problem. Since threshold sharing involves extensive sharing between users for secret shares and communication between users is typically left to the server, this approach introduces significant communication overhead and additional communication rounds. This single technical limitation also leads to secure aggregation protocols for scenarios that must have user dropout robustness to use only secret sharing primitives for encryption.

We hope that in the future development of secure aggregation more solutions to the user dropout problem will emerge or targeted sharing solutions with better performance can be proposed, and FastShare[12] gives us a good start.

D. Code Open Source

We note that out of the seven secure aggregation protocols we investigated, only the code of SAFElearn[11] is open source, which is not very conducive to the development of secure aggregation protocols. From the perspective of secure aggregation protocols alone, we suggest and hope that more advanced secure aggregation protocols will be open-sourced.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (61701170), Science and technology development plan of Henan Province (192102210282, 202102210327).

REFERENCES

- [1] M. Mohri, A. Rostamizadeh, A. Talwalkar, *Foundations of machine learning*, MIT press, 2018.

- [2] H. B. McMahan, E. Moore, D. Ramage, et al., "Federated learning of deep networks using model averaging," *arXiv preprint*, arXiv:1602.05629, 2016.
- [3] A. Shamir, "How to share a secret," *Communications of the ACM*, vol.22, no. 11, pp. 612-613, 1979.
- [4] C. Dwork, J. Lei, "Differential privacy and robust statistics," in *Proc. ACM STOC*, 2009, pp. 371-380.
- [5] A. Acar, H. Aksu, A. S. Uluagac, et al., "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1-35, 2018.
- [6] C. Fung, C. J. M. Yoon, I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint*, arXiv:1808.04866, 2018.
- [7] K. Bonawitz, V. Ivanov, B. Kreuter, et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC CCS*, 2017, pp. 1175-1191.
- [8] O. Goldreich, *Foundations of Cryptography*, Volume 2. Cambridge university press, 2003.
- [9] J. H. Bell, K. A. Bonawitz, A. Gascón, et al., "Secure single-server aggregation with (poly) logarithmic overhead," in *Proc. ACM SIGSAC CCS*, 2020, pp. 1253-1269.
- [10] K. Mandal, G. Gong, "PrivFL: Practical privacy-preserving federated regressions on high-dimensional data over mobile networks", in *Proc. ACM SIGSAC CCS Workshop*, 2019, pp. 57-68.
- [11] H. Fereidooni, S. Marchal, M. Miettinen, et al., "SAFElearn: secure aggregation for private federated learning" in *Proc. IEEE SP Workshops*, 2021, pp. 56-62.
- [12] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, et al., "Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning," *arXiv preprint*, arXiv:2009.11248, 2020.
- [13] C. Beguier, M. Andreux, E. W. Tramel. "Efficient Sparse Secure Aggregation for Federated Learning," *arXiv preprint*, arXiv:2007.14861, 2020.
- [14] R. Xu, N. Baracaldo, Y. Zhou, et al., "Hybridalpha: An efficient approach for privacy-preserving federated learning," in *Proc. ACM Workshop on Artificial Intelligence and Security*, 2019, pp. 13-23.
- [15] R. C. Geyer, T. Klein, M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint*, arXiv:1712.07557, 2017.
- [16] M. Malekzadeh, B. Hasircioglu, N. Mital, et al., "Dopamine: Differentially Private Federated Learning on Medical Data," *arXiv preprint*, arXiv:2101.11693, 2021.
- [17] C. Zhang, S. Li, J. Xia, et al., "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. USENIX ATC*, 2020, pp. 493-506.
- [18] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, et al., "POSEIDON: Privacy-preserving federated neural network learning," *arXiv preprint*, arXiv:2009.00349, 2020.
- [19] R. Cramer, I. Damgård, U. Maurer, "General secure multi-party computation from any linear secret-sharing scheme," in *Proc. EUROCRYPT*, 2000, pp. 316-334.
- [20] S. Goldwasser, S. D. Gordon, V. Goyal, et al., "Multi-input functional encryption," in *Proc. EUROCRYPT*, 2014, pp. 578-602.
- [21] R. Banner, Y. Nahshan, D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," *Advances in Neural Information Processing Systems*, vol. 32, pp. 7950-7958, 2019.

Sufang Zhou received the B.S. degree in computer science from Henan Normal University in 2013, and the Ph.D. degree in computer software and theory from Shaanxi Normal University in 2019. She is currently a master supervisor within the School of Computer and Information Engineering, Henan University. Her research interests include secure multiparty computation, privacy-preserving machine learning.

Mingzhe Liao is currently an undergraduate student at Henan University. His research interests include secure multiparty computation and security aggregation.

Qiao Baojun, Professor, doctoral supervisor, Dean of School of computer and information engineering, Henan University. He obtained a doctoral degree in computer software and theory from Beijing University of technology in 2007. At present, his main research directions include spatial data analysis, software engineering and privacy-preserving distributed computing.

Xiaobo Yang received a master degree in computer software and theory from Shaanxi Normal University in 2016. He is currently a senior engineer within

the China United Network Communication Corporation Henan Branch. Her research interests include network security, distributed computing.