VOSA: Verifiable and Oblivious Secure Aggregation for Privacy-Preserving Federated Learning

Yong Wang, *Graduate Student Member, IEEE*, Aiqing Zhang, *Member, IEEE*, Shu Wu, and Shui Yu, *Fellow, IEEE*

Abstract—Federated learning has emerged as a promising paradigm by collaboratively training a global model through sharing local gradients without exposing raw data. However, the shared gradients pose a threat to privacy leakage of local data. The central server may forge the aggregated results. Besides, it is common that resource-constrained devices drop out in federated learning. To solve these problems, the existing solutions consider either only efficiency, or privacy preservation. It is still a challenge to design a verifiable and lightweight secure aggregation with drop-out resilience for large-scale federated learning. In this article, we propose VOSA, an efficient verifiable and oblivious secure aggregation protocol for privacy-preserving federated learning. We exploit aggregator oblivious encryption to efficiently mask users' local gradients. The central server performs aggregation on the obscured gradients without revealing the privacy of local data. Meanwhile, each user can efficiently verify the correctness of the aggregated results. Moreover, VOSA adopts a dynamic group management mechanism to tolerate users' dropping out with no impact on their participation in future learning process. Security analysis shows that the VOSA can guarantee the security requirements of privacy-preserving federated learning. The extensive experimental evaluations conducted on real-world datasets demonstrate the practical performance of the proposed VOSA with high efficiency.

Index Terms—Secure aggregation, federated learning, privacy preservation, data security

1 Introduction

Federated learning (FL) has become a popular distributed machine learning paradigm for collaboratively training models among a number of users, which may be mobile phone or other mobile devices. It enables decentralized resource-constrained devices to participate in model training without sharing raw data [1], [2]. In FL, each user only shares the trained local model gradients periodically, which are aggregated by a central server to get a more accurate global model [3]. In this way, FL facilitates the usage of sensitive datasets and alleviates the data privacy problem caused by collecting different datasets.

However, the transmitted model gradients also leak private information in FL [4], [5], [6], [7]. It leads to FL vulnerable to some attacks, for example, model inversion attacks [8] and membership inference attacks [9]. It is necessary to protect the individual model updates during the training process of FL [10]. The problem of gradient privacy protection can be addressed using secure aggregation, where each user encrypts their shared gradients, and the server aggregates the received gradient ciphertexts. Thus, the server only learns the aggregated gradients without any private information about the local gradients.

In addition to the privacy of model gradients, the verifiability of the aggregated result from central server is also an important factor in privacy-preserving FL [15]. In particular, the server is likely to be semi-trusted, which may dishonestly aggregate partial of gradients for saving computing resources. Even worse, the malicious server may tamper or forge the aggregated result to mislead involved users' learning results [16]. Therefore, verifying the correctness of the aggregated result is an important security requirement in FL.

To protect privacy of gradients, some recent works adopted homomorphic encryption to build privacy-preserving FL systems [11], [12], [13]. Unfortunately, the homomorphic encryption is extraordinarily expensive, which causes that the system is not efficient to process large-scale data. Besides, Google proposed a secure aggregation (SA) protocol [14], which integrates lightweight cryptographic primitives into FL framework to guarantee the privacy of gradients. Whereas the computational and communication overhead of SA grow quadratically against the amount of users, which limits the scalability of SA.

With the rapid development of the mobile Internet of Things, a large number of resource-constrained devices participate in FL for collaborative training models [17], [18]. In

- *Yong Wang and Aiqing Zhang are with the School of Physics and Electronic Information, Anhui Provincial Engineering Laboratory on Information Fusion and Control of Intelligent Robot, Anhui Normal University, Wuhu, Anhui 241002, China. E-mail: 493265093@qq.com, aqzhang2006@163.com.*
- *Shu Wu is with the School of Electronic and Information Engineering, West Anhui University, Lu'an, Anhui 237000, China. E-mail: wuyshu@126.com.*
- *Shui Yu is with the School of Computer Science, University of Technology Sydney, Sydney, NSW 2007, Australia. E-mail: shui.yu@uts.edu.au.*

Manuscript received 17 June 2022; revised 3 November 2022; accepted 26 November 2022. Date of publication 5 December 2022; date of current version 1 September 2023.
This work was supported in part by the National Natural Science Foundation of China under Grant 62072005, in part by the Natural Science Foundation of Anhui Province under Grant 2108085Y22.
(Corresponding author: Aiqing Zhang.)
Digital Object Identifier no. 10.1109/TDSC.2022.3226508

this large-scale FL scenario, it is a common phenomenon for users dropping out caused by unreliable connectivity [19], [20]. In addition, the computational and communication resources are limited for resource-constrained devices [21], [22]. It brings new challenges to the design of verifiable secure aggregation in privacy-preserving FL. Especially, the following two problems arise: (1) How to realize a verifiable secure aggregation in a computation-efficient and communication-efficient way on resource-constrained devices? (2) How to manage users' dropping out, and support drop-out users participating in any future rounds with security and efficiency? To handle the above issues, we propose a verifiable and oblivious secure aggregation for privacy-preserving FL. It is efficient for the resource-constrained devices. In this work, the shared local gradients and aggregated auxiliary information of multiple users are masked by themselves independently. Then, the gradient ciphertexts and aggregated auxiliary information are sent to an aggregation server and a collector respectively. In such a way, neither the aggregation server nor the collector can violate the privacy of individual gradients of users. Besides, if the information for aggregation of a certain user are not sent to both the collector and the aggregation server, they will not be summed in the aggregation. It helps to tolerate users dropping out. Moreover, the aggregation server can efficiently compute a proof for the aggregated result. Each user can verify the correctness of the aggregated result with designed bilinear pairing. It prevents the dishonest behavior of the aggregation server and ensures the integrity of the aggregated result.

In summary, our contributions are summarized as follows.

- We propose an efficient verifiable and oblivious secure aggregation protocol for privacy-preserving federated learning, called VOSA. It guarantees the privacy of users' local gradients during the training process. The untrusted aggregation server only learns the sum over users' individual gradients. Furthermore, it allows users to verify the correctness of the aggregated result from the aggregation server with low overhead.
- We design a dynamic group management mechanism to deal with users' dropping out, and support drop-out users' participation in any future rounds. In our VOSA, there is no trust key center to distribute key and no key agreement among users when encrypting their private data. It selects a semi-trusted user as collector to gather partial key information which can handle the dropout of users efficiently while keeping their secret keys confidential during the training process. Our scheme allows users to dynamically join or leave without key redistribution.
- We provide a comprehensive security analysis and proof for our VOSA. It demonstrates the privacy preservation of the local gradients as well as the verifiability of the aggregated result. Besides, we conduct extensive experiments over real-world data and evaluate the model accuracy and performance of the VOSA. The results demonstrate that our scheme achieves comparable accuracy to the plaintext baseline. Furthermore, our scheme has lower computational and communication overhead than existing schemes.

The rest of this paper is organized as follows. Section 2 discusses the related works about our scheme. Section 3 introduces the preliminary technologies in this paper. Section 4 gives the system model, security model and design goals. In Section 5, we present the detailed VOSA protocol. Then, we analyze and prove the security of the proposed scheme in Section 6, and evaluate the performance in Section 7. Finally, Section 8 concludes this paper.

2 Related Work

2.1 Secure Aggregation for Privacy-Preserving Federated Learning

Our research focuses on the line of works using secure aggregation to protect the privacy of gradients in FL. Bonawitz et al. [14] presented the first lightweight secure aggregation protocol (SA) for FL. In their protocol, the pairs of users use Diffie-Hellman key exchange to generate pairwise masks and use secret sharing to share secret key. They adopted the double-masking mechanism to encrypt the client's local gradients, which is resilient to client dropouts during the aggregation process. However, it requires high additional computational and communication cost to recover the masks of dropping out clients. Due to the fragments of clients' private keys are shared to others, their private keys will be revealed when some clients accidentally drop out in an aggregation round, which prevents the clients safely participating in any future rounds unless a new key setup is conducted.

There are some follow-up works [20], [23], [24], [25], [26] attempting to improve the performance of [14]. Kadhe et al. [23] proposed a multi-secret sharing scheme based on Fast Fourier Transform (FFT) and leverages it to design a secure aggregation protocol, FastSecAgg. This work reduced the computational cost of SA, but maintains the same communication cost. Besides, FastSecAgg only tolerated users dropouts of a certain constant fraction of the users. The work [24] applied additive homomorphic secret sharing to improve SA, which can aggregate shared data without reconstruction. Liu et al. [20] used the homomorphic pseudorandom generator to replace the Diffie-Hellman key exchange protocol of SA. These works reduce the computational and communication overhead of SA to some extent, but they cannot deal with users' dropouts efficiently.

Choi et al. [25] proposed an improved version of SA, to reduce the computational and communication cost. The key idea is to replace the complete graph topology of secret-sharing nodes as a sparse random graph. Similarly, to reduce the overhead of SA, So et al. [26] designed a multi-group circular strategy for efficient model aggregation. It employs additive secret sharing to enable privacy of users and combines Lagrange coding to tolerate dropped users, but the system performance decreases with the increasing number of dropped clients. The works [25] and [26] both require additional assumptions regarding client topology information. The above solutions only support semi-honest server. They did not consider the verifiability of secure aggregation.

In the aspect of verifiable secure aggregation, some works are based on double-masking framework [15], [16], [28]. VerifyNet [15] combined double-masking with homomorphic

hash function to realize verifiable secure aggregation in FL. As each gradient is encrypted by using homomorphic hash and pseudorandom functions, this scheme needs a lot of extra overhead for verification. VERIFL [28] integrated linear homomorphic hash with equivocal commitment scheme to improve VerifyNet, which significantly reduced the communication overhead of VerifyNet. Nevertheless, the performance of VERIFL also decreases with the increasing number of dropped clients. VERSA [16] achieved the verifiability of secure aggregation by employing double aggregation protocol. It minimizes the verification cost by using pseudorandom generator, but each user had to keep the same secret vectors hidden from server, which posed a serious security threat.

Fu et al. proposed VFL [29], a verifiable privacy-preserving FL scheme with efficient communication. It used Lagrange interpolation to verify the correctness of the aggregated gradients, and applied the Chinese Remainder Theorem to reduce communication overhead. However, this scheme did not consider users dropouts, and was not robust to the collusion attacks between server and multi users. Zheng et al. [30] integrated the quantization technique into secure aggregation realizing lightweight and communication-efficient aggregation. In this solution, the quantizer required a pre-set threshold based on a public calibration dataset.

The existing secure aggregation schemes consider either improving the efficiency of secure aggregation, or achieving verifiability of secure aggregation, but fail to simultaneously tackle secure and lightweight aggregation, verifiability, and scalable drop-out resilience in federated learning.

2.2 Aggregator Oblivious Encryption

Aggregator oblivious encryption is a useful privacy-preserving aggregation method that was first put forward by Shi et al. [32]. It allows an untrusted aggregator periodically compute the sum of all participants' encrypted value, without compromising each participant's privacy. However, the proposed encryption scheme only supports small plaintext spaces for computing sums, which limits its usage in numerous applications. To get rid of the limitation of [32], Joye et al. [33] proposed a scalable scheme that supports large plaintext spaces and number of participants. This work is built over Paillier's encryption scheme. The required parameters are somewhat large. Next, Benhamouda et al. [34] presented a general framework with shorter ciphertexts and better encryption times. It constructed a privacy-preserving aggregator oblivious encryption scheme by a variant of Cramer-Shoup's paradigm of smooth projective hashing. However, these works fail to tolerate participants dropouts as well as dynamic joining and leaving.

Chan et al. [35] proposed a privacy-preserving stream aggregation scheme with fault tolerance. It allows aggregator to aggregate the sum over the remaining participants when an arbitrary subset of participants dropouts. This scheme builds a binary interval tree to handle users dropouts. Besides, Jawurek et al. [36] used a custom zero-knowledge proof to tolerate users failure, but it needs heavy cryptographic operations. Leontiadis et al. [40] further proposed a privacy-preserving data aggregation scheme, which supports dynamic group management and arbitrary user failures without relying on trusted key center. These schemes solve the problem of users failure, but do not consider the attacks of malicious aggregator.

Considering the aggregator may provide a bogus aggregated result to users, Leontiadis et al. [39] proposed a verifiable secure data aggregation scheme. In this scheme, the aggregator has to compute a proof for the correctness of the aggregated result using homomorphic tag sent by users. However, each user encrypts its data according to Shi et al. [32] scheme, which inherits the drawbacks of small plaintext spaces. Emura [38] improved Leontiadis et al. [39] scheme using a variant of the computational Diffie-Hellman assumption. This scheme can be constructed on elliptic curves of a smaller order and lead to efficient implementation.

Motivated by the above works and consider the demands of privacy-preserving federated learning, we design a novel verifiable aggregator oblivious encryption protocol with fault tolerance. Moreover, we exploit the designed protocol to realize verifiable secure aggregation for privacy-preserving federated learning.

3 Preliminaries

This section briefly introduces the knowledge of federated learning, aggregator oblivious encryption and cryptographic assumptions in our scheme.

3.1 Federated Learning

Federated learning (FL) is a distributed learning technique that allows users to train a shared global model cooperatively by their local private data [1]. In FL, the participating users train a same machine learning model locally to get local model update. An aggregation server collects and aggregates all the local model update to get a joint global model. Assume that there are N users in FL, each user has its private dataset $\mathcal{D}_i$, where $|\mathcal{D}_i| = d_i$. The training process is summarized as follows.

- *User selection:* At the beginning of each epoch $t \in \{1, 2, \ldots\}$, the aggregation server randomly selects a subset of users $\mathbb{U}_t$ and sends them the current model parameters w_{t-1} which is trained in the previous epoch. These users may drop out at any process in the current epoch.
- *Local training:* Each selected user $\mathcal{U}_i$ trains the model on the local training dataset and updates the model parameters $w_{i,t}$ by an optimization algorithm, e.g., stochastic gradient descent (SGD). Then, it uploads the gradients $\Delta_{i,t} \leftarrow w_{i,t} - w_{i,t-1}$ to the aggregation server.
- *Model aggregation:* The aggregation server collects the gradients from online users and computes an aggregated gradients, e.g., using the federated averaging algorithm [27] to get a weighted average of these gradients $w_t \leftarrow w_{t-1} + \sum_{i \in \mathbb{U}_t} \lambda_i \Delta_{i,t}$, where $\lambda_i = d_i / \sum_{i \in \mathbb{U}_t} d_i$.
- *Global model update:* The aggregation server updates the global model and publishes it to participating users for the subsequent epoch.

This training process will be iteratively executed several epochs until the model convergence.

3.2 Verifiable Aggregator Oblivious Encryption With Fault Tolerance

Aggregator oblivious encryption allows an untrusted data aggregator to periodically compute an aggregate value over ciphertexts from multiple users, while protecting each individual's privacy. In our scheme, we require a secure aggregation protocol to aggregate local model parameters from users in federated learning. The protocol updates global model parameters via aggregation server while hiding individual local parameters [42], [45]. Besides, each user should be able to verify the correctness of the aggregated result for preventing malicious server to modify or forge the result [46], [47]. It is common for users to drop out accidently in training process. Therefore, the protocol should handle dropped-out users and aggregate model parameters of surviving users. We propose a verifiable aggregator oblivious encryption with fault tolerance to solve the above challenges. The proposed protocol consists of the following algorithms:

- $Setup(1^k) \rightarrow (pm, sk_A, \{sk_i, tk_i\})$: Input the security parameter k, it outputs the public parameter pm, the secret key sk_A of aggregation server, and the secret keys $\{sk_i, tk_i\}$ of user $\mathcal{U}_i$.
- $MaskTag(t, w_{i,t}, sk_i) \rightarrow (C_{i,t}, T_{i,t})$: Taking current epoch t, local model parameter $w_{i,t}$ and secret key sk_i of user $\mathcal{U}_i$ as input, it outputs gradients ciphertext $C_{i,t}$ and authentication tag $T_{i,t}$.
- $Collect(\{Au_{i,t}, Vk_{i,t}\}) \rightarrow (Au_t, Vk_t)$: Given the auxiliary information $\{Au_{i,t}, Vk_{i,t}\}$ of all online users $\mathcal{U}_i$, it outputs the unmasking key Au_t and verification key Vk_t.
- $UnmaskAgg(\{C_{i,t}, T_{i,t}\}, sk_A, Au_t) \rightarrow (W_t, T_t)$: It takes the gradients ciphertext $C_{i,t}$ and authentication tag $T_{i,t}$ from online users, sk_A of aggregation server, and unmasking key Au_t from collector as input, and outputs the aggregated gradient $W_t = \sum w_{i,t}$ and corresponding proof T_t.
- $Verify(W_t, T_t, Vk_t, pk_{A,t}) \rightarrow \{0, 1\}$: Input the aggregated gradient W_t and corresponding proof T_t, verification key Vk_t from collector, and public key $pk_{A,t}$ of aggregation server, it outputs 1 if users $\mathcal{U}_i$ convince that the aggregated gradient W_t is correct; and 0 otherwise.

3.3 Cryptographic Assumptions

Definition 1. Decisional Composite Residuosity Assumption (DCR). *Let $N = pq$ be a product of two big primes. The DCR problem is to distinguish $R_0 = \{y = x^N \bmod N^2, x \in Z^*_{N^2}\}$ and $R_1 = \{y \in Z^*_{N^2}\}$. The advantage of a probabilistic polynomial-time adversary $\mathcal{A}$ to deciding the DCR problem [37] is*

$$Adv_{DCR} = \left| \begin{matrix} \Pr[\mathcal{A}(y = x^N \bmod N^2,\ x \in Z^*_{N^2}) = 1] \\ -\Pr[\mathcal{A}(y \in Z^*_{N^2}) = 1] \end{matrix} \right|.$$

If the advantage $Adv_{DCR} \leq \varepsilon$, where ε is a negligible function, it is assumed that the DCR problem is a hard problem in polynomial time.

Definition 2. Decisional Diffie-Hellman Assumption (DDH). *Let G be a cyclic group of prime order p. The DDH problem is to distinguish $D_0 = \{(g, g^x, g^y, g^{xy}) | g \in G, x, y \in Z_p\}$ and $D_1 = \{(g, g^x, g^y, g^z) | g \in G, x, y, z \in Z_p\}$. The advantage of a probabilistic polynomial-time adversary $\mathcal{A}$ to deciding the DDH problem [38] is*

$$Adv_{DDH} = \left| \begin{matrix} \Pr[\mathcal{A}(g, g^x, g^y, g^{xy}) = 1] \\ -\Pr[\mathcal{A}(g, g^x, g^y, g^z) = 1] \end{matrix} \right|.$$

If the advantage $Adv_{DDH} \leq \varepsilon$, where ε is a negligible function, it is assumed that the DDH problem is a hard problem in polynomial time.

Definition 3. Bilinear Computational Diffie-Hellman Assumption (BCDH). *Let g_1 and g_2 be two generators for cyclic group G_1 and G_2 respectively, and $a, b, c \in Z^*_q$ be random numbers. The BCDH problem is to compute $e(g_1, g_2)^{abc} \in G_T$. The advantage of a probabilistic polynomial-time adversary $\mathcal{A}$ to computing the BCDH problem [38] is*

$$Adv_{BCDH} = \Pr[\mathcal{A}(g_1, g_2, g_1^a, g_1^b, g_1^c, g_2^a, g_2^b) = e(g_1, g_2)^{abc}].$$

If the advantage $Adv_{BCDH} \leq \varepsilon$, where ε is a negligible function, it is assumed that the BCDH problem is a hard problem in polynomial time.

Definition 4. LEOM Assumption. *Let $D = (e, p, g_1, g_2, G_1, G_2)$ be the parameters of bilinear groups. Define $A = g_2^a$, $T = g_2^{\sum_{i=1}^n \gamma_i}$, where $a, \gamma_i \in Z_p$. The LEOM oracle $\mathcal{O}_{LEOM}$ takes as input $(t, \{w_{i,t}\}_{i=1}^n)$, chooses random elements $\alpha, \beta_t \in G_1$, and returns $(\alpha, \beta_t, \{\beta_t^{\gamma_i} \alpha^{w_{i,t}a}\})$. The LEOM problem is to query $\mathcal{O}_{LEOM}$ with $(t, \{w_{i,t}\}_{i=1}^n)$. The advantage of a probabilistic polynomial-time adversary $\mathcal{A}$ to querying the LEOM problem [39] is*

$$Adv_{LEOM} = \Pr[\mathcal{A}_{\mathcal{O}_{LEOM}}(D, A, T) \rightarrow (t, z \neq \textstyle\sum_{i=1}^n w_{i,t}, \alpha, \beta_t, \{\beta_t^{\sum_{i=1}^n \gamma_i} \alpha^{za}\})].$$

If the advantage $Adv_{LEOM} \leq \varepsilon$, where ε is a negligible function, it is assumed that the LEOM problem is a hard problem in polynomial time.

4 System Model

4.1 System Architecture

As shown in Fig. 1, the proposed system consists of four types of entities, Key Generation Center (KGC), Users, Collector, and Aggregation Server.

- *Key Generation Center (KGC):* The KGC mainly initializes the system and generates public parameters. It distributes the parameters to users and aggregation server.
- *Users:* At each epoch, each user $\mathcal{U}_i$ trains the model using local private data to get local updated gradient. Then, he/she encrypts his/her local gradients, generates gradient tag, and sends them to aggregation server. Furthermore, the online users obfuscate their secret key as auxiliary information and send it to the collector. It will be used later to compute and verify the aggregated sum of individual gradient.

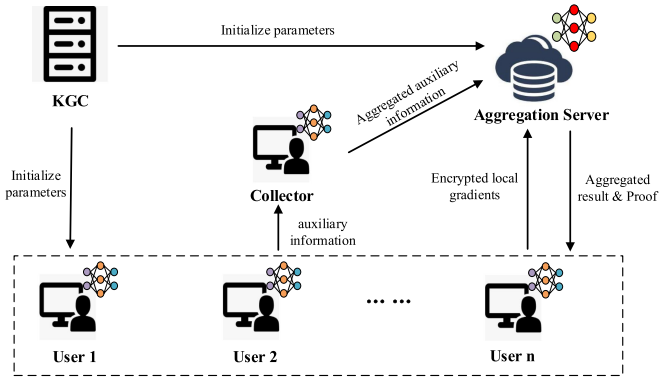


Fig. 1. System architecture.

The users can verify the correctness of the aggregated gradients.

- *Collector:* A collector is randomly selected from online users at each epoch. It's responsible for collecting obfuscated secret keys from online users who participant in current iteration. Afterwards, he/she sends the collective auxiliary information to aggregation server.
- *Aggregation Server:* The aggregation server aggregates the uploaded encrypted gradients from all online users at each epoch. Besides, it generates proof for its aggregated results using auxiliary information and then distributes the results to online users.

4.2 Security Model

In this paper, we focus on the potential security and privacy vulnerability of FL posed by malicious aggregation server. We assume that the KGC is a trusted third party who will not collude with any entity. The KGC can be performed by an authority party in practice. It only generates and publishes system parameters and goes offline. It is significant to note that the KGC does not generate the individual secret keys of users and aggregation server. All users are considered to be honest-but-curious. They will provide effective gradients and execute protocol honestly, but they also aim to infer the data privacy of other users. Since the collector is selected from the online users during each iteration, he/she has the same security properties as the users. Besides, we suppose that the aggregation server may be captured by malicious entity. It may deviate the protocol or execute different computation to capture private gradients of honest users. Meanwhile, it may modify or forge the aggregated result and proof for deceiving the users to accept a wrong global parameter. We allow that the aggregation server colludes with multiple users to forge the aggregated result and proof, but the collusion of aggregation server and collector is not allowed in this scheme.

Remark. The non-colluding setting between aggregation server and collector is actually weaker than the single-server setting, but it has been widely formalized and presented in previous works. For instance, SecureML (IEEE S&P'17) [41], EPRFL(ACM ACSAC'21) [43], PEFL(IEEE TIFS'21) [44] all adopted double non-colluding server architecture. There are two advantages for this setting: 1) Users significantly reduce communication complexity and overheads by only transmitting partial key information to collector without exchanging key information with all others users which is necessary in existing popular secure aggregation scheme [14], [15], [28]. 2) It can effectively tolerate users dropping out without additional overhead and support drop-out users' participation flexibly in any future rounds.

4.3 Design Goals

According to the proposed security model, our scheme aims to achieve the following design goals:

- *Aggregation server obliviousness.* It ensures that the aggregation server learns nothing except the aggregated gradients from users' encrypted gradients, authentication tag and auxiliary information. If the malicious aggregation server colludes with an arbitrary set of users, it will only learn the aggregated value of other honest users without individual private data.
- *Aggregate unforgeability and verifiability.* It guarantees the aggregation server honestly aggregates the individual gradients of online users. The aggregate value supports strong verifiability to each user so that the malicious adversary cannot convince users to accept a bogus aggregate sum.
- *Collector obliviousness.* It ensures that the collector cannot infer any private information from the received auxiliary information of users. In addition, even if the collector colludes with an arbitrary set of users, the collector does not obtain additional private information about other honest users.
- *Users dropping out tolerance.* It allows the aggregation server compute the correct aggregated value even if a set of users fail to participate at an arbitrary epoch. If the users send encrypted gradients to the aggregation server and then drop out due to communication delay or heardware errors, the aggregation server cannot get privacy data of the offline users from the received information.

5 Proposed VOSA Scheme

In this section, we design a verifiable and oblivious secure aggregation protocol for privacy-preserving federated learning. We first present the high level view of the proposed VOSA scheme. Then, we explain the technical details of our protocol.

5.1 Overview of the VOSA

The proposed VOSA is consisted of five phases: Set up phase, masking and tag phase, collection phase, unmasking and aggregating phase, and verification phase. We give a high level view of our VOSA in Fig. 2.

In set up phase, KGC generates public system parameters pm and publishes it to users and aggregation server. Then, it goes offline and does not participate in other phases anymore. Each user $\mathcal{U}_i$ who participates in federated learning chooses its secret key (sk_i, tk_i) independently. Besides, the aggregation server generates its secret key sk_A.

In masking and tag phase, each user $\mathcal{U}_i$ encrypts its local gradients $w_{i,t}$ to generate gradient ciphertext $C_{i,t}$, and

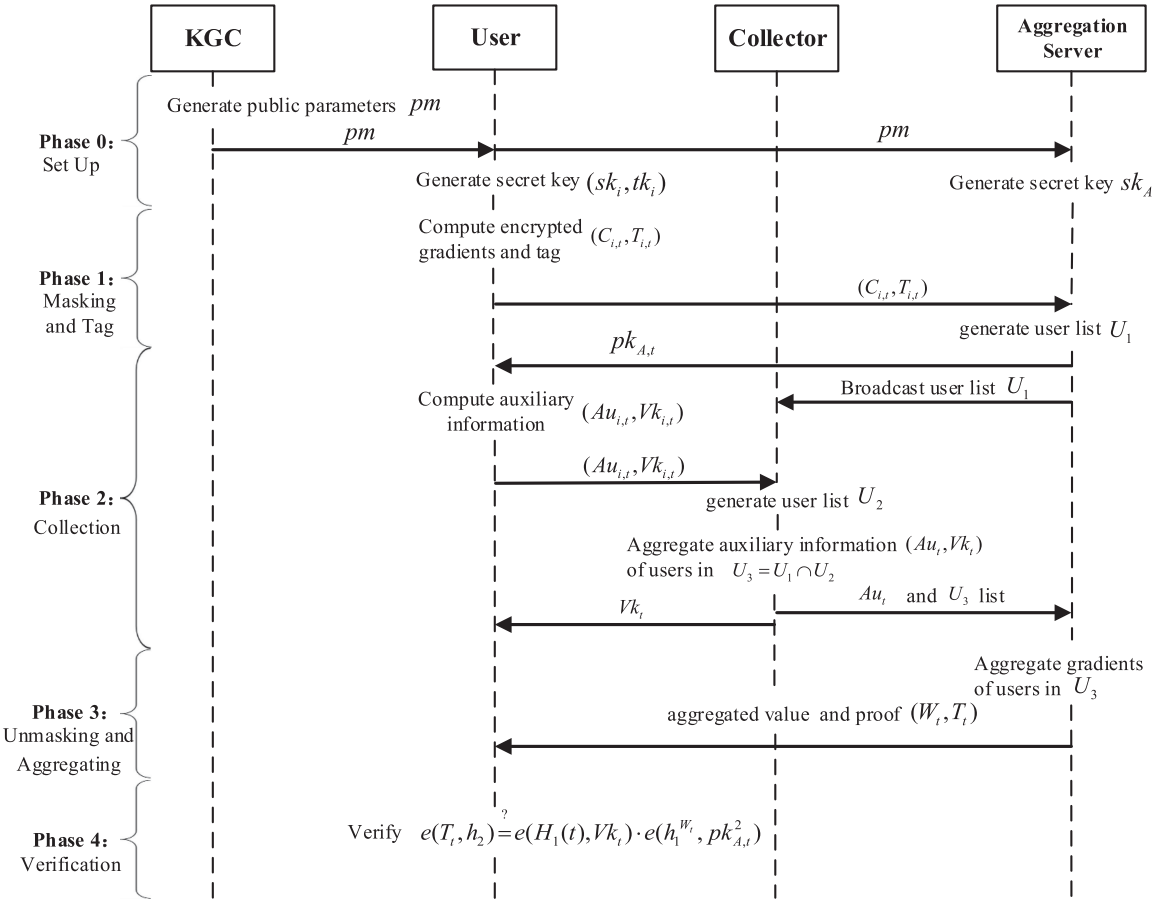


Fig. 2. High-level view of VOSA.

construct an authentication tag $T_{i,t}$. Then, it transmits $(C_{i,t}, T_{i,t})$ to the aggregation server for getting global optimized gradients. Aggregation server collects all the messages from individual users and denotes this set of users as U_1.

In collection phase, the aggregation server first broadcasts its public key $PK_{A,t}$ to users in U_1 and informs the collector the list U_1. Once the public key is received, each user $\mathcal{U}_i \in U_1$ calculates the auxiliary information $(Au_{i,t}, Vk_{i,t})$ and sends them to the collector. However, some users may drop out accidentally and do not send auxiliary information to collector in time. The collector gathers all auxiliary information from the surviving users and denotes this set of users as U_2. The users in $U_3 = U_1 \cap U_2$ are regarded as valid online users who send both encrypted gradients and auxiliary information. Then, the collector aggregates the auxiliary information of users in U_3 to generate unmasking key and verification key (Au_t, Vk_t). The verification key Vk_t is sent to online users in U_3, while the unmasking key and users list U_3 are sent to the aggregation server.

In unmasking and aggregating phase, the aggregation server utilizes unmasking key to aggregate the global gradients $W_t = \sum_{\mathcal{U}_i \in U_3} w_{i,t}$. Moreover, it aggregates authentication tag from individual users to generate proof T_t for supporting users verify the correctness of aggregated value. Then, it broadcasts the aggregated value and proof to users in U_3.

In verification phase, each user verifies the availability of proof by using verification key. If the validation passes, the user accepts the aggregated value and uses the optimized parameter to train local data. Otherwise, it rejects the calculated result.

5.2 The Proposed VOSA at Each Epoch

Phase 0: Set up phase

We assume that there are n users participating in federated learning training, where the users are represented by $U = \{\mathcal{U}_i, i \in [1, n]\}$. All the users agree on a model architecture and train the model locally on their private data sets D_i.

First, the KGC initializes the model parameter w_0, chooses two secure big primes p, q and sets $N = pq$. Let G_1 and G_2 be two multiplicative cyclic groups with the same prime order p. g_1 and g_2 are two random generators of G_1 and G_2 respectively. The KGC selects a secret value $a \in Z_p^*$ and compute $h_1 = g_1^a$, $h_2 = g_2^a$. Moreover, it chooses a computable bilinear pairings $e : G_1 \times G_2 \rightarrow G_T$ and two secure hash functions $H_0 : \{0,1\}^* \rightarrow Z_{N^2}^*$, $H_1 : \{0,1\}^* \rightarrow G_1$. Then, the KGC publishes the public system parameters $pm = (N, w_0, g_1, g_2, h_1, h_2, G_1, G_2, G_T, H_0, H_1)$ and goes offline. Besides, the aggregation server randomly generates its secret key $sk_A \in Z_{N^2}^*$. Each user $\mathcal{U}_i$ picks two random numbers $sk_i \in [0, N^2]$ and $tk_i \in Z_{N^2}^*$ as his/her encryption key and tag key, respectively.

Phase 1: Masking and tag phase

Generally, we denote a neural network as a function $f(x, w) = \hat{y}$, where x is the input of users, and $\hat{y}$ is the output through the function f with the model parameter w. The user $\mathcal{U}_i$ holds the local training set $D_i = \{\langle x_j, y_j \rangle | j = 1, 2, \ldots, T\}$, where x_j is data feature, y_j is data label, T is the

size of D_i. The loss function can be defined as

$$L_f(D_i, w) = \frac{1}{T} \sum_{< x_j, y_j > \in D_i} l(y_j, f(x_j, w)),$$

where $l(y_j, f(x_j, w))$ can be a selective specific loss function. In this scheme, we set the loss function as the cross entropy loss.

The goal of training neural network is to find an optimal gradient parameter w to minimize the loss function. In this phase, each user executes stochastic gradient descent algorithm to update local gradient $w_{i,t}$ as follows.

$$w_{i,t+1} \leftarrow w_{i,t} - \lambda \nabla L_f(D_i^*, w_{i,t}),$$

where $w_{i,t}$ denotes the gradient parameter of user $\mathcal{U}_i$ after tth iteration. λ is learning rate, and ∇L_f is derivative of the loss function, where D_i^* is a random subset of D_i.

Without loss of generality, we exemplify the secure aggregation operation in the tth epoch of training. For epoch t, each user $\mathcal{U}_i$ needs to encrypt its local gradients using the encryption key sk_i to generate gradient cipheretext

$$C_{i,t} = (1 + w_{i,t}N) H_0(t)^{sk_i} \bmod N^2.$$

The user $\mathcal{U}_i$ also constructs an authentication tag with its tag key tk_i, and outputs tag

$$T_{i,t} = H_1(t)^{tk_i} h_1^{w_{i,t}}.$$

After that, $\mathcal{U}_i$ sends $(C_{i,t}, T_{i,t})$ to the aggregation server.

Phase 2: Collection phase

When the aggregation server received $(C_{i,t}, T_{i,t})$ from the user $\mathcal{U}_i$, it will include $\mathcal{U}_i$ into user list U_1. It waits for enough users and informs the collector of the users list U_1. At the same time, aggregation server generates its public key $pk_{A,t} = (pk_{A,t}^1, pk_{A,t}^2)$ and distributes it to users in U_1, where $pk_{A,t}^1 = H_0(t)^{sk_A}$ and $pk_{A,t}^2 = h_2^{sk_A}$. Each user $\mathcal{U}_i \in U_1$ obfuscates its secret key to generate the auxiliary information

$$Au_{i,t} = (pk_{A,t}^1)^{sk_i} = H_0(t)^{sk_A sk_i}$$

$$Vk_{i,t} = (pk_{A,t}^2)^{tk_i} = h_2^{sk_A tk_i},$$

and sends $(Au_{i,t}, Vk_{i,t})$ to collector.

Consider a scenario where some users drop out accidentally and do not send auxiliary information to the collector. The collector assumes that these users have dropped, and builds a dynamic user group U_2. Then, the users in $U_3 = U_1 \cap U_2$ who send effective $(C_{i,t}, T_{i,t})$ to the aggregation server as well as sending $(Au_{i,t}, Vk_{i,t})$ to the collector are regarded as valid online users. For simplicity, we assume there are m valid online users in U_3. The collector computes unmasking key

$$Au_t = \prod_{i=1}^m Au_{i,t} = \prod_{i=1}^m H_0(t)^{sk_A sk_i} = H_0(t)^{sk_A \sum_{i=1}^m sk_i},$$

and verification key

$$Vk_t = \prod_{i=1}^m Vk_{i,t} = \prod_{i=1}^m h_2^{sk_A tk_i} = g_2^{sk_A a \sum_{i=1}^m tk_i}.$$

Afterwards, the collector sends Au_t and user list U_3 to the aggregation server, and distributes Vk_t to all users in U_3.

Phase 3: Unmasking and aggregating phase

After the aggregation server receives the unmasking key Au_t from the collector, it performs aggregation and generates corresponding proof. The aggregation server first calculates

$$C_t = (\prod_{i=1}^m C_{i,t})^{sk_A} \bmod N^2,$$

and obtains the aggregated gradients W_t by evaluating

$$W_t = sk_A^{-1} \frac{\frac{C_t}{Au_t} - 1}{N} \bmod N = \sum_{i=1}^m w_{i,t} \bmod N.$$

The sum of gradients is meaningful as long as $\sum_{i=1}^m w_{i,t} < N$. In practice, N is a 2048-bit value, so that there is no restriction on the size of $w_{i,t}$ or the total number n of users.

The correctness of aggregation value is as follows.

$$\begin{aligned} C_t &= (\prod_{i=1}^m C_{i,t})^{sk_A} \bmod N^2 \\ &= (\prod_{i=1}^m (1 + w_{i,t}N) H_0(t)^{sk_i})^{sk_A} \bmod N^2 \\ &= (1 + \sum_{i=1}^m w_{i,t}N)^{sk_A} H_0(t)^{sk_A \sum_{i=1}^m sk_i} \bmod N^2 \\ &= (1 + sk_A \sum_{i=1}^m w_{i,t}N) H_0(t)^{sk_A \sum_{i=1}^m sk_i} \bmod N^2 \end{aligned}$$

$$\begin{aligned} W_t &= sk_A^{-1} \frac{\frac{C_t}{Au_t} - 1}{N} \bmod N \\ &= sk_A^{-1} \frac{\frac{(1 + sk_A \sum_{i=1}^m w_{i,t}N) H_0(t)^{sk_A \sum_{i=1}^m sk_i}}{H_0(t)^{sk_A \sum_{i=1}^m sk_i}} - 1}{N} \bmod N \\ &= sk_A^{-1} \frac{(1 + sk_A \sum_{i=1}^m w_{i,t}N) - 1}{N} \bmod N \\ &= \sum_{i=1}^m w_{i,t} \bmod N. \end{aligned}$$

Furthermore, the aggregation server aggregates all authentication tags and outputs proof

$$\begin{aligned} T_t &= (\prod_{i=1}^m T_{i,t})^{sk_A} = (\prod_{i=1}^m H_1(t)^{tk_i} h_1^{w_{i,t}})^{sk_A} \\ &= H_1(t)^{sk_A \sum_{i=1}^m tk_i} g_1^{sk_A a \sum_{i=1}^m w_{i,t}}. \end{aligned}$$

Finally, the aggregation server broadcasts the aggregated gradients and corresponding proof (W_t, T_t) to users in U_3.

Phase 4: Verification phase

After receiving the aggregated gradients and proof, each user verifies the correctness of the aggregation value with the verification key Vk_t by checking the following equality:

$$e(T_t, h_2) \stackrel{?}{=} e(H_1(t), Vk_t) \cdot e(h_1^{W_t}, pk_{A,t}^2).$$

If the equation holds, the aggregated gradients is correct. The user accepts the aggregated gradients and updates the local model. Otherwise, it drops the aggregation value and enters into next iteration.

VOSA Protocol

- **Phase 0 *(Setup)*:**
  KGC:
  - Initialize system parameters $pm = (N, w_0, g_1, g_2, h_1, h_2, G_1, G_2, G_T, H_0, H_1)$, where $N = pq$, w_0 is model parameter, G_1, G_2, G_T are multiplicative cyclic groups, g_1, g_2 are generators of G_1 and G_2, $h_1 = g_1^{\ a}, h_2 = g_2^{\ a}$, H_0, H_1 are hash functions.

  User :
  - Choose $sk_i \in [0, N^2]$ as his/her encryption key.
    Choose $tk_i \in Z^*_{N^2}$ as his/her tag key.

  Aggregation server :
  - Choose $sk_A \in Z^*_{N^2}$ as its secret key.
- **Phase 1 *(Masking and Tag):***
  User :
  - Train local data with model parameter w and generate local gradients $w_{i,t}$.
  - Encrypt the local gradients as $C_{i,t} = (1 + w_{i,t}N)H_0(t)^{sk_i} \mod N^2$.
  - Constructs the authentication tag as $T_{i,t} = H_1(t)^{tk_i} h_1^{w_{i,t}}$.
  - Send $(C_{i,t}, T_{i,t})$ to aggregation server.
- **Phase 2 (Collection):**
  Aggregation server :
  - Generate user list U_1, which consists of users who send $(C_{i,t}, T_{i,t})$ to the aggregation server.
  - Inform the user list U_1 to the collector.
  - Generate and distribute its public key $pk_{A,t} = (pk^1_{A,t}, pk^2_{A,t}) = (H_0(t)^{sk_A}, h_2^{sk_A})$ to users in U_1.

  Users in U_1:
  - Calculate the auxiliary information $Au_{i,t} = (pk^1_{A,t})^{sk_i} = H_0(t)^{sk_A sk_i}$ and $Vk_{i,t} = (pk^2_{A,t})^{tk_i} = h_2^{sk_A tk_i}$.
  - Send $(Au_{i,t}, Vk_{i,t})$ to the collector.

  Collector :
  - Build users list U_2, which consists of users who send $(Au_{i,t}, Vk_{i,t})$ to the collector.
  - Compute unmasking key of users in U_3 ($U_3 = U_1 \cap U_2$, the users in U_3 are regard as valid Online users.) as $Au_t = \prod_{i=1}^m Au_{i,t} = \prod_{i=1}^m H_0(t)^{sk_A sk_i} = H_0(t)^{sk_A \sum_{i=1}^m sk_i}$.
  - Compute verification key of users in U_3 as $Vk_t = \prod_{i=1}^m Vk_{i,t} = \prod_{i=1}^m h_2^{sk_A tk_i} = g_2^{ask_A \sum_{i=1}^m tk_i}$.
  - Send Au_t and users list U_3 to aggregation server.
  - Distribute Vk_t to all users in U_3.
- **Phase 3 (Unmasking and aggregate):**
  Aggregation server :
  - Calculate the aggregation value of encrypted gradients $C_t = (\prod_{i=1}^m C_{i,t})^{sk_A} \mod N^2$.
  - Unmask the aggregation value as $W_t = sk_A^{-1} \frac{\frac{C_t}{Au_t} - 1}{N} \mod N = \sum_{i=1}^m w_{i,t} \mod N$.
  - Generate corresponding proof $T_t = (\prod_{i=1}^m T_{i,t})^{sk_A} = H_1(t)^{sk_A \sum_{i=1}^m tk_i} g_1^{ask_A \sum_{i=1}^m w_{i,t}}$.
  - Broadcast the aggregated gradients and proof (W_t, T_t) to users in U_3.
- **Phase 4 (Verification):**
  Users in U_3 :
  - Verify the correctness of aggregation value by checking $e(T_t, h_2) \stackrel{?}{=} e(H_1(t), Vk_t) \cdot e(h_1^{W_t}, pk^2_{A,t})$.
    If the above equation holds, accept the aggregated result. Otherwise, reject the result.

Fig. 3. Detailed description of the VOSA.

The correctness of verification follows from bilinear pairing properties:

$$\begin{aligned} e(T_t, h_2) &= e(H_1(t)^{sk_A \sum_{i=1}^m tk_i} g_1^{sk_A a \sum_{i=1}^m w_{i,t}}, g_2^a) \\ &= e(H_1(t)^{sk_A \sum_{i=1}^m tk_i}, g_2^a) \cdot e(g_1^{sk_A a \sum_{i=1}^m w_{i,t}}, g_2^a) \\ &= e(H_1(t), g_2^{sk_A a \sum_{i=1}^m tk_i}) \cdot e(g_1^{\sum_{i=1}^m w_{i,t}}, (g_2^a)^{sk_A}) \\ &= e(H_1(t), Vk_t) \cdot e(h_1^{W_t}, pk^2_{A,t}). \end{aligned}$$

At the end of this phase, users update their local model parameter: $w_{i,t+1} = w_{i,t} - \lambda \frac{W_t}{|U_3|}$. The users, aggregation server and collector iteratively run the entire protocol until it meets the termination condition required by users. We summarize our protocol in Fig. 3.

6 Security Analysis and Proof

6.1 Aggregation Server Obliviousness

Theorem 1. *The proposed protocol can ensure aggregation server obliviousness in the random oracle model under the decisional composite residuosity (DCR) [37] and Decisional Diffie-Hellman (DDH) [38] assumption.*

Proof. Assume that aggregation server $\mathcal{A}$ is a polynomial-time adversary having a non-negligible advantage ε in breaking the aggregation server obliviousness of our protocol. We build a challenger $\mathcal{C}$ playing the security game [40] with $\mathcal{A}$ as follows. □

Setup. The challenger $\mathcal{C}$ performs the setup of the protocol and returns public system parameters $pm = (N, g_1, g_2, h_1, h_2, G_1, G_2, G_T, H_0, H_1)$ to $\mathcal{A}$. The aggregation server $\mathcal{A}$ selects

a secret key $sk_A \in Z^*_{N^2}$ and publishes its public key $pk_{A,t} = (pk^1_{A,t}, pk^2_{A,t}) = (H_0(t)^{sk_A}, h_2^{sk_A})$ for each epoch t.

Learning. $\mathcal{A}$ issues the following queries.

Corruption Queries. Whenever $\mathcal{A}$ decides to corrupt a user $\mathcal{U}_i$, it queries with the identifier of the user $\mathcal{U}_i$. The challenger $\mathcal{C}$ responds the encryption key $sk_i \in [0, N^2]$ and tag key $tk_i \in Z^*_{N^2}$ of user $\mathcal{U}_i$.

Masking and Tag Queries. Whenever $\mathcal{A}$ makes masking queries with $w_{i,t}$, the challenger $\mathcal{C}$ returns matching gradients ciphertext $C_{i,t} = (1 + w_{i,t}N)H_0(t)^{sk_i}$ and authentication tag $T_{i,t} = H_1(t)^{tk_i} h_1^{w_{i,t}}$.

Auxiliary Information Queries. Whenever $\mathcal{A}$ queries the auxiliary information at epoch t, the challenger $\mathcal{C}$ computes $Au_t = \prod_{i=1}^m Au_{i,t} = H_0(t)^{sk_A \sum_{i=1}^m sk_i}$ and $Vk_t = \prod_{i=1}^m Vk_{i,t} = g_2^{ask_A \sum_{i=1}^m tk_i}$ and forwards them to $\mathcal{A}$.

Challenge. In the challenge phase, $\mathcal{A}$ chooses a challenge epoch t^* for which it does not make a masking query during the learning phase, an uncorrupted subset $U^* \subseteq U_3$ and two distinct gradients w^0_{i,t^*} and w^1_{i,t^*} which satisfy $\sum w^0_{i,t^*} = \sum w^1_{i,t^*}$. Then, the challenger $\mathcal{C}$ flips a fair coin $b \in \{0, 1\}$ and computes the gradients ciphertext $\{C^b_{i,t^*} = (1 + w^b_{i,t^*}N) H_0(t^*)^{sk_i}\}_{\mathcal{U}_i \in U^*}$. It also computes the authentication tag $T^b_{i,t^*} = H_1(t^*)^{tk_i} h_1^{w^b_{i,t^*}}$. Finally, the challenger $\mathcal{C}$ returns $\{C^b_{i,t^*}, T^b_{i,t^*}\}_{\mathcal{U}_i \in U^*}$.

It is important to note that the interaction of the aggregation server $\mathcal{A}$ with simulated challenger $\mathcal{C}$ or actual oracles is computationally indistinguishable. Actually, the messages are correctly computed and given to the aggregation server $\mathcal{A}$. The verification of the aggregated value is correct in the simulation, more precisely

$$
\begin{aligned}
&e((\prod_{\mathcal{U}_i \in U^*} T^b_{i,t^*})^{sk_A}, h_2) = e((\prod_{\mathcal{U}_i \in U^*} H_1(t^*)^{tk_i} g_1^{aw^b_{i,t^*}})^{sk_A}, g_2^a) \\
&= e(H_1(t^*)^{sk_A \sum_{\mathcal{U}_i \in U^*} tk_i}, g_2^a) \cdot e(g_1^{ask_A \sum_{\mathcal{U}_i \in U^*} w^b_{i,t^*}}, g_2^a) \\
&= e(H_1(t^*), g_2^{ask_A \sum_{\mathcal{U}_i \in U^*} tk_i}) \cdot e(g_1^{a \sum_{\mathcal{U}_i \in U^*} w^b_{i,t^*}}, (g_2^a)^{sk_A}) \\
&= e(H_1(t^*), Vk_{t^*}) \cdot e(h_1^{W^b_{t^*}}, pk^2_{A,t^*}).
\end{aligned}
$$

Guess. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. If $b' = b$, the challenger $\mathcal{C}$ outputs 1, which means the ciphertext $\{C^b_{i,t^*}, T^b_{i,t^*}\}_{\mathcal{U}_i \in U^*}$ is an encryption of gradient w^1_{i,t^*}. Otherwise, it is an encryption of gradient w^0_{i,t^*}.

If $\mathcal{A}$ has a non-negligible advantage ε in breaking the aggregation server obliviousness of our protocol, it will output a correct guess b' with a non-negligible advantage ε. According to the security proof of [32], [33], it means there is a challenger $\mathcal{C}$ which solves the DCR and DDH problem with the same non-negligible advantage ε. It is a contradiction under the DCR and DDH assumption. Therefore, the proposed protocol achieves aggregation server obliviousness in the random oracle model under the DCR and DDH assumption.

6.2 Aggregate Unforgeability and Verifiability

The proposed protocol can ensure aggregate unforgeability and verifiability in two cases. The first one is that aggregation server $\mathcal{A}$ never makes a masking and tag query for challenging epoch t^*. We denote this type of forgery as Type I. The second is that the aggregation server $\mathcal{A}$ has made a masking and tag query at challenging epoch t^*, but the forged sum $W_{t^*} \neq \sum_{\mathcal{U}_i \in U^*} w_{i,t^*}$. We call this type of forgery as Type II.

Theorem 2. *Our scheme realizes aggregate unforgeability and verifiability against the Type I attack in the random oracle model under the BCDH [38] assumption.*

Proof. Assume the aggregation server $\mathcal{A}$ is a polynomial-time adversary which breaks the aggregate unforgeability and verifiability with a non-negligible advantage ε. We build a challenger $\mathcal{C}$ to play the security game [39] with $\mathcal{A}$ as follows. □

Setup. The challenger $\mathcal{C}$ performs the setup and returns public system parameters $pm = (N, g_1, g_2, h_1, h_2, G_1, G_2, G_T, H_0, H_1)$ to $\mathcal{A}$. It randomly selects encryption key $sk_i \in [0, N^2]$ and tag key $tk_i \in Z^*_{N^2}$ for users $\mathcal{U}_i \in U$, and the secret key $sk_A \in Z^*_{N^2}$ for the aggregation server $\mathcal{A}$. Then, it computes the public key $pk_{A,t} = (pk^1_{A,t}, pk^2_{A,t}) = (H_0(t)^{sk_A}, h_2^{sk_A})$ of $\mathcal{A}$ and verification key $Vk_t = h_2^{sk_A \sum_{u_i \in U_3} tk_i}$.

Learning. $\mathcal{A}$ issues the following queries.

Hash Queries. $\mathcal{A}$ chooses an epoch t and queries the random oracle H_1. $\mathcal{C}$ constructs a H_1 list and responds $\mathcal{A}$ as follows:

1) If t has not been queried before, it chooses a random number $r_t \in Z_{N^2}$ and flips a random coin $b_t \in \{0, 1\}$. If $b_t = 0$ with probability p, $\mathcal{C}$ returns $H_1(t) = g_1^{r_t}$. Otherwise if $b_t = 1$, $\mathcal{C}$ responds $H_1(t) = g_1^{cr_t}$ and adds the new tuple $\langle r_t, b_t, H_1(t)\rangle$ to H_1 list.
2) If t already appears in H_1 list, then $\mathcal{C}$ responds $\mathcal{A}$ the $H_1(t)$ from H_1 list.

Masking and Tag Queries. Whenever $\mathcal{A}$ makes masking and tag queries with $(t, \mathcal{U}_i, w_{i,t})$, $\mathcal{C}$ constructs a list T and responds $\mathcal{A}$ as follows:

1) If $\mathcal{A}$ has never made the queries at epoch t, then:
   - $\mathcal{C}$ initializes a variable $W_t = 0$.
   - $\mathcal{C}$ executes hash queries and responds the new tuple $\langle r_t, b_t, H_1(t)\rangle$ to H_1 list. If $b_t = 1$, the simulation abort. Otherwise, it chooses secret tag key $tk_i \in Z^*_{N^2}$ where i represent the users $\mathcal{U}_i$.
   - $\mathcal{C}$ computes the tag $T_{i,t} = g_1^{br_t tk_i} g_1^{aw_{i,t}} = H_1(t)^{btk_i} g_1^{aw_{i,t}}$, which is a valid tag for gradient $w_{i,t}$.
   - $\mathcal{C}$ also chooses a secret encryption key $sk_i \in [0, N^2]$ and computes the gradients ciphertext $C_{i,t} = (1 + w_{i,t}N)H_0(t)^{sk_i}$, which is a correct ciphertext for gradient $w_{i,t}$.
   - $\mathcal{C}$ sets $W_t = W_t + w_{i,t}$ and adds the tuple $(t, \mathcal{U}_i, w_{i,t}, T_{i,t})$ to the T list.
2) Else if the T list contains $i' = \mathcal{U}_i$ and $w'_{i,t} = w_{i,t}$, $\mathcal{C}$ returns the corresponding $T_{i,t}$ from the list to $\mathcal{A}$.
3) Else if the T list contains $i' = \mathcal{U}_i$ and $w'_{i,t} \neq w_{i,t}$, $\mathcal{C}$ aborts.
4) Otherwise, $\mathcal{C}$ looks for the H_1 list with the index t to get tuple $\langle r_t, b_t, H_1(t)\rangle$. If the tuple does not exist, $\mathcal{C}$ flips a random coin $b_t \in \{0, 1\}$. If $b_t = 0$, then aborts. If $b_t = 1$, $\mathcal{C}$ execute the same operations in step 1).

Challenge. $\mathcal{A}$ forges the value $(W^*_{t^*}, T^*_{t^*})$ at epoch t^*. If $W^*_{t^*} \neq W_{t^*}$, $\mathcal{C}$ queries the H_1 list to get the appropriate tuple $\langle r_{t^*}, b_{t^*}, H_1(t^*)\rangle$.

- If $b_{t^*}=0$, $\mathcal{C}$ aborts.
- If $b_{t^*}=1$, $\mathcal{A}$ outputs a valid forged $T^*_{t^*}$ at epoch t^*. It is true that the following equation should hold:

$$e(T^*_t,h_2)=e(H_1(t^*),Vk_{t^*})e(h_1^{W^*_t},pk^2_{A,t^*}).$$

It holds when $\mathcal{A}$ makes n masking and tag queries at epoch t^* to different users during the learning phase. As such $T^*_{t^*}=g_1^{cr_{t^*}bsk_A\sum tk_i}g_1^{sk_AaW^*_t}$, $\mathcal{C}$ outputs

$$\begin{aligned}&e\left(\left(\frac{T^*_{t^*}}{g_1^{ask_AW^*_{t^*}}}\right)^{\frac{1}{r_tsk_A\sum tk_i}},g_2^a\right)\\&=e\left(\frac{g_1^{cr_tbsk_A\sum tk_i}g_1^{ask_AW^*_{t^*}}}{g_1^{ask_AW^*_{t^*}}}\right)^{\frac{1}{r_tsk_A\sum tk_i}},g_2^a\Bigg)\\&=e(g_1^{bc},g_2^a)\\&=e(g_1,g_2)^{abc}.\end{aligned}$$

If the adversary $\mathcal{A}$ successfully forges a Type I forgery T_t in our protocol with a non-negligible advantage ε, then the challenger $\mathcal{C}$ can break the BCDH problem with the same non-negligible advantage ε. It is a contradiction under the BCDH assumption. Therefore, the proposed protocol can resist Type I forgery attack in the random oracle model under the BCDH assumption.

Theorem 3. *Our scheme realizes aggregate unforgeability and verifiability against the Type II attack in the random oracle model under the LEOM [39] assumption.*

Proof. The process of the simulation between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ is essentially the same as that of the Type I forgery attack, except that challenger $\mathcal{C}$ uses the LEOM random oracle $\mathcal{O}_{LEOM}$ to generate the corresponding values in the hash queries and masking and tag queries. □

Hash Queries. $\mathcal{A}$ queries with input $(t,w_{i,t})$. The random oracle $\mathcal{O}_{LEOM}$ replies with $(\alpha,\beta_t,\beta_t^{tk_i}\alpha^{aw_{i,t}})$, where $\alpha=g_1,\beta_t\in G_1,a\in Z_{N^2}$. Then, $\mathcal{C}$ returns $H_1(t)=\beta_t$ to $\mathcal{A}$.

Masking and Tag Queries. $\mathcal{A}$ queries with $(t,\mathcal{U}_i,w_{i,t})$. $\mathcal{C}$ responds with the tag $T_{i,t}=\beta_t^{tk_i}\alpha^{aw_{i,t}}$ and gradients ciphertext $C_{i,t}=(1+w_{i,t}N)\beta_t^{sk_i}$. $\mathcal{A}$ can correctly verify the aggregated result as follows:

$$\begin{aligned}e((\prod T_{i,t})^{sk_A},h_2)&=e((\prod\beta_t^{tk_i}\alpha^{aw_{i,t}})^{sk_A},g_2^a)\\&=e(\beta_t^{sk_A\sum tk_i},g_2^a)\cdot e(\alpha^{ask_A\sum w_{i,t}},g_2^a)\\&=e(\beta_t,g_2^{ask_A\sum tk_i})\cdot e(\alpha^{a\sum w_{i,t}},g_2^{ask_A})\\&=e(H_1(t),Vk_t)\cdot e(h_1^{\sum w_{i,t}},pk^2_{A,t}).\end{aligned}$$

Therefore, the tags $T_{i,t}=\beta_t^{tk_i}\alpha^{aw_{i,t}}$ are valid verifiable forged tags. If the adversary $\mathcal{A}$ successfully makes a Type II forgery attack in our protocol with a non-negligible advantage ε, which means that the challenger $\mathcal{C}$ can break the LEOM problem with the same non-negligible advantage ε. It is a contradiction under the LEOM assumption. We conclude that our scheme can achieve aggregate unforgeability and verifiability for a Type II forgery in the random oracle model under the LEOM assumption.

6.3 Collector Obliviousness

Theorem 4. *The proposed protocol can ensure collector obliviousness in the random oracle model under the DCR and DDH assumption.*

Proof. Assume the collector $\mathcal{C}'$ is a polynomial-time adversary which breaks the collector obliviousness with a non-negligible advantage ε. We build a challenger $\mathcal{C}$ to play the security game [40] with $\mathcal{C}'$ as follows. □

Setup. The challenger $\mathcal{C}$ sends public system parameters $pm=(N,g_1,g_2,h_1,h_2,G_1,G_2,G_T,H_0,H_1)$ to $\mathcal{C}'$. $\mathcal{C}$ simulates the aggregation server $\mathcal{A}$ to selects a secret key $sk_A\in Z^*_{N^2}$ and publishes its public key $pk_{A,t}=(pk^1_{A,t},pk^2_{A,t})=(H_0(t)^{sk_A},h_2^{sk_A})$ for each epoch t.

Learning. $\mathcal{C}'$ issues the following queries.

Corruption Queries. It executes the same simulation as in the game of aggregation server obliviousness.

Masking and Tag Queries. It executes the same simulation as in the game of aggregation server obliviousness.

Auxiliary Information Queries. Whenever $\mathcal{C}'$ queries the auxiliary information with epoch t, user's identity $\mathcal{U}_i$, and gradients ciphertext $C_{i,t}$, $\mathcal{C}$ executes masking queries with tuple $(t,\mathcal{U}_i,0)$ and returns $(1+0\cdot N)H_0(t)^{sk_i}=H_0(t)^{sk_i}$. Then it computes $Au_{i,t}=H_0(t)^{sk_Ask_i}$ and responds to $\mathcal{C}'$.

Challenge. $\mathcal{C}'$ chooses a challenge epoch t^* for which it does not make a masking query during the learning phase, an uncorrupted subset $U^*\subseteq U$ and two distinct tuples $\mathcal{W}^0_{t^*}=(t^*,\mathcal{U}_i,w^0_{i,t^*})_{\mathcal{U}_i\in U^*}$ and $\mathcal{W}^1_{t^*}=(t^*,\mathcal{U}_i,w^1_{i,t^*})_{\mathcal{U}_i\in U^*}$. Then, $\mathcal{C}$ executes the following simulations:

- It picks $\mathcal{W}^0_{t^*}$ and generates a new tuples $\mathcal{W}'^1_{t^*}=(t^*,\mathcal{U}_i,w'^1_{i,t^*})_{\mathcal{U}_i\in U^*}$ which satisfy $\sum w^0_{i,t^*}=\sum w'^1_{i,t^*}$. Then, it flips a fair coin $b\in\{0,1\}$ and computes the gradient ciphertext $\{C^b_{i,t^*}=(1+w^b_{i,t^*}N)H_0(t^*)^{sk_i}\}_{\mathcal{U}_i}\in U^*$. If $b=0$, $\{C^b_{i,t^*}\}_{\mathcal{U}_i\in U^*}$ is a masking of the $\mathcal{W}^0_{t^*}$. Otherwise, it is a masking of the $\mathcal{W}^1_{t^*}$.
- $\mathcal{C}$ randomly chooses $pk^1_{A,t^*}\in Z^*_{N^2}$ and picks a random number $r^b_{i,t^*}\in Z^*_{N^2}$. Then it computes auxiliary information as $Au^b_{i,t^*}=r^b_{i,t^*}$.
- Finally, the challenger $\mathcal{C}$ returns $\{C^b_{i,t^*},Au^b_{i,t^*}\}_{\mathcal{U}_i\in U^*}$ to $\mathcal{C}'$. It is noteworthy that $\mathcal{C}'$ cannot distinguish that the pk^1_{A,t^*} and Au^b_{i,t^*} are randomly generated or computed as $pk^1_{A,t^*}=H_0(t^*)^{sk_A}$and $Au_{i,t}=H_0(t^*)^{sk_Ask_i}$ under the DDH assumption and the random oracle model.

Guess. $\mathcal{C}'$ outputs a guess $b^*\in\{0,1\}$. If $b=0$, $\mathcal{C}$ will output the correct guess $b^*=0$ with a non-negligible advantage ε, which means the ciphertext $\{C^b_{i,t^*},Au^b_{i,t^*}\}_{\mathcal{U}_i\in U^*}$ is an encryption of gradient $\mathcal{W}^0_{t^*}$. If $b=1$, the ciphertext $\{C^b_{i,t^*},Au^b_{i,t^*}\}_{\mathcal{U}_i\in U^*}$ is independent of the gradients $\mathcal{W}^0_{t^*}$ and $\mathcal{W}^1_{t^*}$ sent by $\mathcal{C}'$. Therefore, $\mathcal{C}$ will output the guess $b^*=0$ or $b^*=1$ with probability $1/2$.

To conclude, if the collector $\mathcal{C}'$ has a non-negligible advantage ε in breaking the collector obliviousness of our protocol, it means there is a challenger which solves the DCR and DDH problem with the same non-negligible advantage ε. It is a contradiction under the DCR and DDH assumption. Therefore, the proposed protocol achieves collector obliviousness in the random oracle model under the DCR and DDH assumption.

6.4 Uers Dropping Out Tolerance

Suppose there is a set of users $\mathcal{U}_i \in D$ fail to execute the proposed protocol at the epoch t. Specifically, each users $\mathcal{U}_i \in D$ finishes the executions in masking and tag phase but does not participate in collection phase. It does not affect the aggregation of other users' gradients. To deal with users dropping out, the collector builds a dynamic group of surviving users at each epoch. Indeed, each user $\mathcal{U}_i \notin D$ computes auxiliary information $Au_{i,t} = (pk^1_{A,t})^{sk_i} = H_0(t)^{sk_A sk_i}$ and $Vk_{i,t} = (pk^2_{A,t})^{tk_i} = h_2^{sk_A tk_i}$. After receiving the auxiliary information from surviving users $\mathcal{U}_i \notin D$, the collector $\mathcal{C}'$ computes unmasking key $Au_t = \prod_{\mathcal{U}_i \notin D} Au_{i,t} = \prod_{\mathcal{U}_i \notin D} H_0(t)^{sk_A sk_i}$ and verification key $Vk_t = \prod_{\mathcal{U}_i \notin D} Vk_{i,t} = \prod_{\mathcal{U}_i \notin D} h_2^{sk_A tk_i}$. When the aggregation server $\mathcal{A}$ receives the gradients ciphertext $C_{i,t}$ from users $\mathcal{U}_i \notin D$ and auxiliary information Au_t from the collector $\mathcal{C}'$, it first computes the aggregated ciphertext $\prod_{\mathcal{U}_i \notin D} C_{i,t}$ and further calculates the aggregated gradients $\sum_{\mathcal{U}_i \notin D} w_{i,t}$ by unmasking key. Our scheme adopts the dynamic user group management to handle users dropping out without revealing the secret key of drop-out users. In this way, these drop-out users can participate in any future epoch safely. Moreover, according to the security analysis of aggregation server obliviousness, the aggregation server $\mathcal{A}$ cannot infer any privacy information of users $\mathcal{U}_i \in D$ from received ciphertext $C_{i,t}$. Thus, our scheme realizes correct aggregation even if an arbitrary number of users drop out in the protocol as long as the collector $\mathcal{C}'$ operates properly.

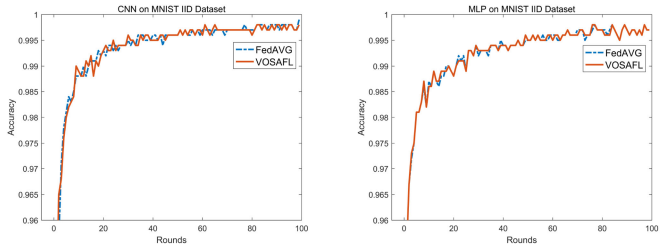
7 Performance Evaluation

This section verifies the effectiveness of the proposed VOSA scheme on model accuracy by conducting experiments on the representative dataset. Further, we evaluate computational overhead and communication overhead of the scheme.

7.1 Experimental Setup

The proposed cryptographic protocol is implemented with pairing-based cryptography (PBC) library[1] and python programming language. We adopt Type A pairings [48] to execute the algorithms, which is constructed on the elliptic curve $y^2 = x^3 + x$ in the finite field $\mathbb{F}_q$ (q is prime and $q \equiv 3 \bmod 4$). Our simulation experiment is conducted on a 64-bits Windows 10 PC with Intel(R) Core(TM) i7-11700 CPU @ 2.50GHz, 16GB RAM, and NVIDIA GeForce RTX 3060 GPU. The GPU is only used to accelerate the model training, while the cryptographic protocol is implemented on the CPU.

The experiments are tested on the MNIST database of handwritten digits, which has 60,000 training data, and 10,000 test data. A multilayer-perceptron (MLP) and a convolution neural network (CNN) are used as the local trained model in federated learning. The MLP network consists of two hidden layers with 200 units each using ReLu activations. The total number of gradient parameters is 199,210. The CNN network consists of two 5×5 convolutional layers (the first with 32 channels, the second with 64 channels, each followed with 2×2 max pooling), a dropout layer with a drop probability of 0.2, a fully connected layer with 512 neurons and ReLu activations, and a softmax output layer. The total number of gradient parameters is 582,026. We implement cross entropy loss as loss function and stochastic gradient descent (SGD) with momentum for optimization. The learning rate we set is 0.01 and 0.03 on CNN and MLP, respectively.



(a) Accuracy of CNN (b) Accuracy of MLP

Fig. 4. The model accuracy comparison between FedAVG and VOSA on IID dataset.

7.2 Model Accuracy

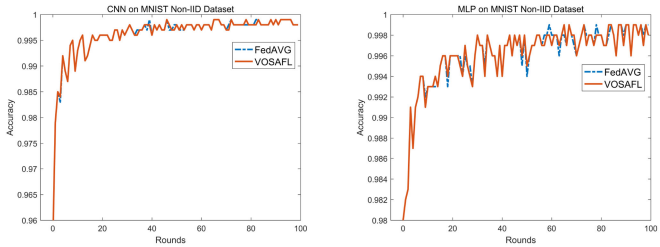
We examine two ways of partitioning the MNIST dataset over users: IID and non-IID. In IID data distribution, the training data is shuffled and distributed to 100 users each receiving 600 examples. In non-IID data distribution, we sort the training data by digit labels, divide them into 200 shards of size 300, and then distribute each user two random shards. We evaluate the plaintext federated learning [27] without privacy preservation of model updates and our VOSA using MLP and CNN model under IID and non-IID database. Following the plaintext federated learning [27], the fraction of users being selected to participate in model update in each round is set to 10%. Each selected user executes SGD updates over 5 epochs with batch size being 10. Then, all users upload the raw value of gradients directly, and aggregation server computes global model gradients by federated averaging (FedAVG) algorithm. However, the users upload gradients ciphertext in our VOSA scheme. The aggregation server unmasks the aggregated value to get the sum of plaintext gradients and then executes FedAVG to update global model.

Fig. 4 shows the model accuracy of our VOSA and plaintext FedAVG in IID database under different rounds. The experiment results confirm that the proposed VOSA realizes similar model accuracy with plaintext federated learning and protect data privacy at the same time. In addition, we show the evaluation results of two model accuracy in non-IID database in Fig. 5. It demonstrates that our VOSA achieves comparable accuracy to the plaintext federated leaning. Therefore, the data heterogeneity is related to the federated learning itself while does not affect the security design of our scheme. Furthermore, the selected users drop-out only affects the number of aggregated gradients in global model and hardly sacrifices the model accuracy. Our scheme ensures correct aggregation even if selected users drop out in federated learning.

7.3 Computational Overhead

We mainly focus on the computational overhead of cryptographic operations in our scheme, and omit the computations performed by the model training on the local dataset. We evaluate the performance of different entities with different

1. https://crypto.stanford.edu/pbc/



(a) Accuracy of CNN (b) Accuracy of MLP

Fig. 5. The model accuracy comparison between FedAVG and VOSA on non-IID dataset.

TABLE 1
Computational Overhead of Each Step for User

gradient numbers	KeyGen	MaskingTag	Collection	Verify	Total
1×10^5	0.49 (ms)	407.62 (ms)	1.59 (ms)	7.52 (ms)	417.22 (ms)
2×10^5	0.62 (ms)	720.49 (ms)	1.45 (ms)	7.53 (ms)	730.09 (ms)
3×10^5	0.48 (ms)	1083.99 (ms)	1.56 (ms)	7.46 (ms)	1093.49 (ms)
4×10^5	0.48 (ms)	1445.58 (ms)	1.46 (ms)	7.48 (ms)	1455 (ms)
5×10^5	0.49 (ms)	1814.33 (ms)	1.46 (ms)	7.44 (ms)	1823.72 (ms)

amount of gradients and users. The amount of gradients and users are denoted by the symbol $|\mathcal{G}|$ and $|\mathcal{U}|$, respectively.

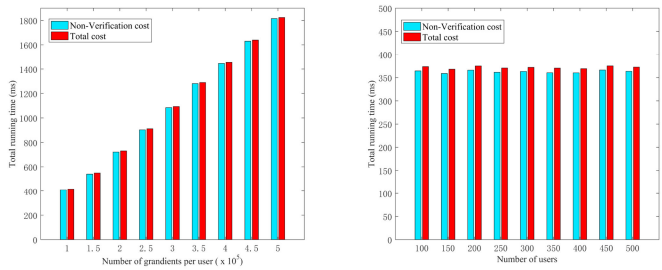
7.3.1 Performance Analysis of User

The user's running time of each step is shown in Table 1. The computational overhead of user mainly comes from four steps: *KeyGen*, *MaskingTag*, *Collection*, and *Verify*. The *Keygen* step generates the user's encryption key and tag key. Then, each user encrypts the trained local gradients and generates the authentication tag at *MaskingTag* step. At *Collection* step, each user calculates the auxiliary information for unmasking and verification. Finally, each user verify the correctness of aggregated gradient values from aggregation server at *Verify* step. From Table 1, we can observe that the user's running time increases linearly with the increasing of the amount of gradient parameters at *MaskingTag* step. Because the encryption operation is related to the amount of gradients, while other steps are independent of the amount of gradients. Even if the main computational cost of each user comes from the *MaskingTag* process, our VOSA still maintains good performance in terms of computational cost. For example, when the total number of gradient parameters are 199210 in MLP and 582026 in CNN, each user only spends around 726 ms and 2512 ms to encrypt the gradients, respectively.

In addition, we evaluate and compare the computational overhead of the verification and non-verification setting in Fig. 6. It shows that the running time of verification is very small in our scheme. Besides, the computational overhead of the verification procedure does not increase with the increasing of the amount of gradient parameters and users. Thus, it can be concluded that the users can verify the aggregated results efficiently in our VOSA scheme.

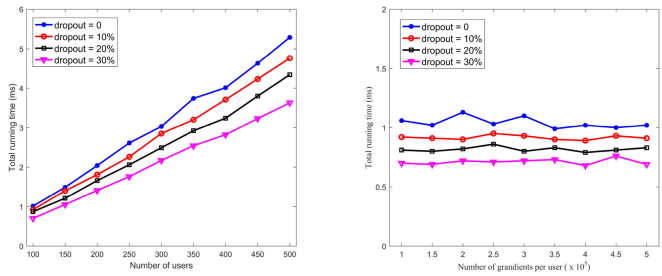
7.3.2 Performance Analysis of Collector

The computational overhead of a collector comes from computing the unmasking key and verification key at collection step. Fig. 7 shows the running time of collector during collection process. It shows that the collector's running time increases linearly with the increasing amount of users, but keeps a constant as the amount of gradients increases. From Fig. 7, we can know that the collector only takes 1 ms to compute the auxiliary information of 100 uesrs. It is affordable for a normal user to act as a collector in our scheme.



(a) (b)

Fig. 6. Comparison between the computational overhead of verification and non-verification for each user. (a) $|\mathcal{U}|$ = 100, with the different amount of gradients per user. (b) $|\mathcal{G}| = 1 \times 10^5$, with the different amount of users.



(a) (b)

Fig. 7. Total running time of the collector in each epoch. (a) $|\mathcal{G}| = 1 \times 10^5$, with the different user amounts and dropout rates. (b) $|\mathcal{U}|$ = 100, with the different gradient amounts and dropout rates.

7.3.3 Performance Analysis of Aggregation Server

The aggregation server's running time of each step is shown in Table 2. The computational overhead of the aggregation server mainly comes from three steps: *KeyGen*, *Unmasking*, and *ProofGen*. The server generates its private key and public key at *KeyGen* step. At *Unmasking* step, the server has to unmask the aggregated gradients ciphertext for getting the sum of the gradients plaintext. Then, it calculates the

TABLE 2
Computational Overhead of Each Step for Aggregation Server ($|\mathcal{U}|$=100)

gradient numbers	KeyGen	Unmasking	ProofGen	Total
1×10^5	2.68 (ms)	14178.56 (ms)	2.35 (ms)	14183.6 (ms)
2×10^5	1.94 (ms)	28459.85 (ms)	2.30 (ms)	28464.1 (ms)
3×10^5	1.78 (ms)	42724.77 (ms)	2.25 (ms)	42728.8 (ms)
4×10^5	1.83 (ms)	56625.13 (ms)	2.27 (ms)	56629.2 (ms)
5×10^5	1.86 (ms)	71018.50 (ms)	2.34 (ms)	71022.7 (ms)

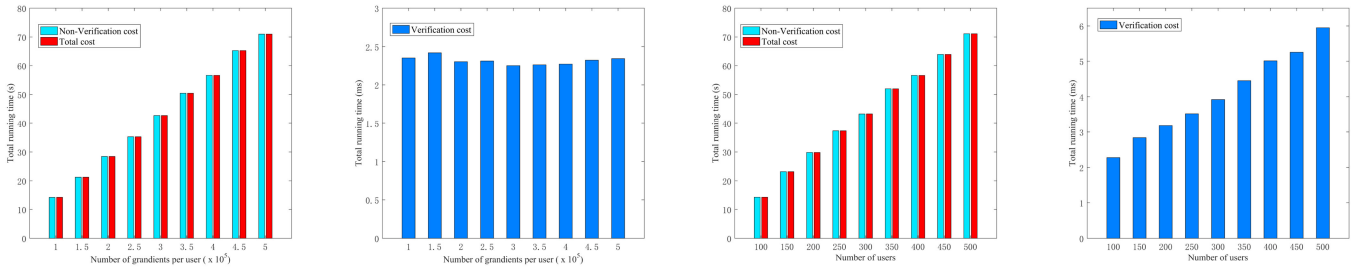


Fig. 8. Computational overhead of the aggregation server. (a) Comparison of time between verification and non-verification setting, $|\mathcal{U}| = 100$, with the different amount of gradients per user. (b) The verification cost, $|\mathcal{U}| = 100$, with the different amount of gradients per user. (c) Comparison of time between verification and non-verification setting, $|\mathcal{G}| = 1 \times 10^5$, with the different amount of users. (d) The verification cost, $|\mathcal{G}| = 1 \times 10^5$, with the different amount of users.

corresponding proof of the aggregated result at *ProofGen* step. We can find that the main computational cost of the aggregation server comes from Unmasking step, and the running time increases linearly with the increasing of the amount of gradients or users. The main reason is that as the amount of gradients or users increases, the server need to calculate and decrypt the aggregated result for new added gradients. In our experiment, the unmasking overhead of aggregation server is around 28.17 s in MLP model and 197.40 s in CNN model.

In addition, we evaluate the computational overhead of the aggregation server with the verification and non-verification setting in Fig. 8. It shows that the proportion of verification time is very small of the total time. The verification overhead of the server comes from generating proof. Fig. 8b shows that the verification overhead keeps a constant with increasing of the amount of gradients. However, as shown in Fig. 8d, it slowly increases linearly with the user amounts. We can conclude that the verification cost has a very limited impact on the computing efficiency of the aggregated server.

7.4 Communication Overhead

7.4.1 Performance Analysis of User

The communication cost of each user with verification and non-verification setting is shown in Fig. 9. As shown in Fig. 9a, the user's total communication overhead increases linearly with the increasing of the amount of gradients. Besides, it indicates that the proportion of verification overhead is not obvious to the total overhead. Fig. 9b shows that the verification cost keeps only 0.3 KB with increasing of the amount of gradients. Since there is no communication between users in our scheme, the communication cost of users will not change with the increasing amount of users. The experiments demonstrate that our scheme have a good performance in communication overhead. For example, when the amount of gradients is 199210 in MLP and 582026 in CNN, each user only needs about 3.80 MB and 11.10 MB to complete model update.

7.4.2 Performance Analysis of Collector

The communication overhead of a collector comes from sending the unmasking key to the aggregation server and distributing the verification key to users. Fig. 10 shows that the collector's communication overhead increases linearly with the increasing of the amount of users, but it keeps independent of the amount of gradients. The experiments demonstrate that the communication cost of the collector is very small. When the number of users is 100, the collector only takes 6.27 KB to transmit data. The communication overhead is affordable for resource-constrained users.

7.4.3 Performance Analysis of Aggregation Server

Fig. 11 compares the verification cost and the total communication cost of the aggregation server. From Figs. 11a and 11c, we can find that as the amount of gradients or users increases, the total communication cost of the aggregation server also increases linearly. However, the verification cost grows linearly with the amount of users (Fig. 11d), and keeps a constant as the increasing amount of gradients (Fig. 11b). The communication cost of the additional verification takes a small proportion of the total cost. When the

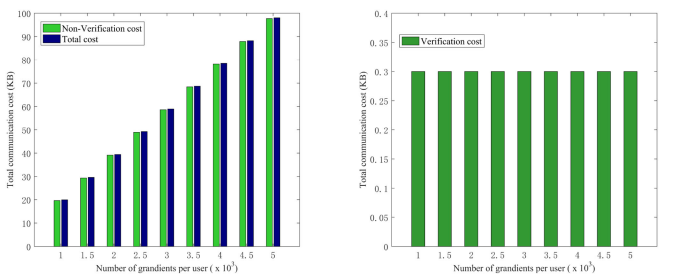


Fig. 9. Communication overhead of each user with the different amount of gradients. (a) Comparison between verification and non-verification setting. (b) The verification cost.

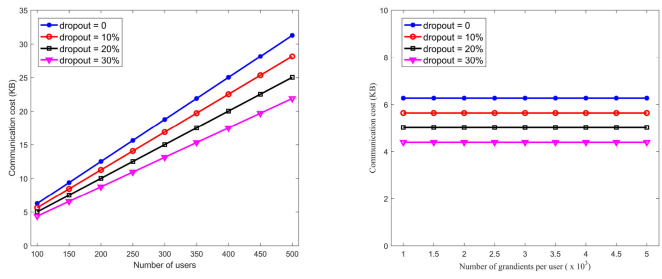


Fig. 10. Communication cost of the collector in each epoch. $|\mathcal{G}| = 1 \times 10^5$ of each user, with the different user amount and dropout rates. (b) $|\mathcal{U}| = 100$, with the different gradient amount and dropout rates.

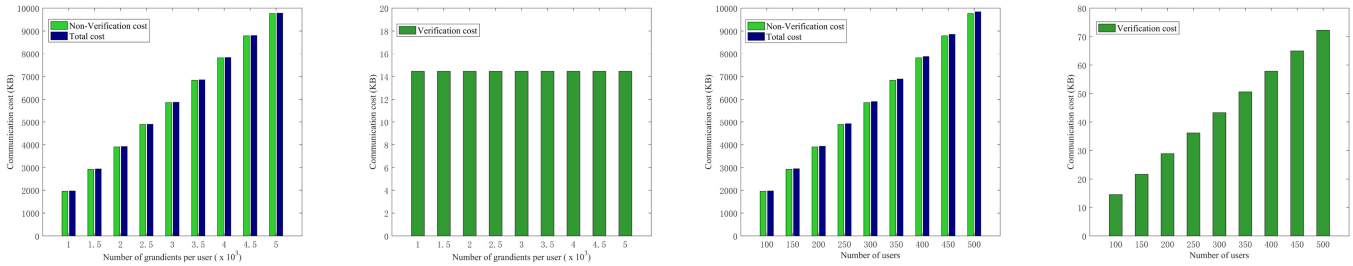


Fig. 11. Communication overhead of the aggregation server. (a) Comparison of cost between verification and non-verification setting, $|\mathcal{U}| = 100$, with the different amount of gradients per user. (b) The verification cost, $|\mathcal{U}| = 100$, with the different amount of gradients per user. (c) Comparison of cost between verification and non-verification setting, $|\mathcal{G}| = 1 \times 10^3$, with the different amount of users. (d) The verification cost, $|\mathcal{G}| = 1 \times 10^3$, with the different amount of users.

amount of users is 100, the aggregation server only takes 14.45 KB to distribute verification information.

7.5 Performance Comparison With Existing Schemes

In this section, we compare the performance of our VOSA with the existing state-of-the-art schemes VerifyNet [15], VERIFL [28], and Non-interactive VFL [31]. All these approaches aim to protect user's privacy in the training process of federated learning and verify the correctness of the aggregated results from the server at the same time. VerifyNet and VERIFL both apply the double-masking protocol [14] to encrypt local gradients. VerifyNet combines homomorphic hash functions and pseudorandom functions to realize the verification functionality, while VERIFL adopts an equivocal commitment to verify the aggregated results. The Non-interactive VFL designs a dual-servers architecture and exploits secure two-party computation to encrypt local gradients. It uses the dual-servers to realize credible matrix exchange for the verification of the aggregated result.

1) *Performance of each user:* We run the cryptographic protocol in python with the same environment. We record the total running time and communication cost of user in VerifyNet, VERIFL, Non-interactive VFL and our VOSA with different number of gradients, as shown in Fig. 12. Since the users may play two different roles in our VOSA, we record their overhead when they work as normal user and collector respectively. We can see that the computational cost of VOSA is significantly smaller than VerifyNet and VERIFL, but it is similar to the Non-interactive VFL. The high computational overhead of VerifyNet and VERIFL comes from the verification process. In details, the VerifyNet verifies all parameter of gradients separately. The VERIFL verifies the linear homomorphic hash of the aggregated gradients. Besides, they adopt Diffie-Hellman key agreement to generate the paired random values for encrypting the local gradients, which causes high computational cost and communication cost. In Non-interactive VFL and our VOSA, the encryption of the gradients and verification functionalities are independent. It improves the computational and communication efficiency of users.

2) *Performance of aggregation server:* Although the architecture of our VOSA is slightly different from other solutions, the aggregation server has the same responsibilities as others. As shown in Fig. 13, we compare the computational cost of the aggregation server with different number of gradients or users in above-mentioned schemes. Because the server has to aggregate many parameters for generating proof in VerifyNet, it causes high computational cost at server side. Our SOVA and the Non-interactive VFL have the minimum computation overhead, while the VERIFL is slightly higher. Fig. 14 shows that the VerifyNet and VERIFL consume more communication resources. The main reason is that they both utilize the Shamir's t-out-of-n secret sharing to share the private key and the verification parameters, which has a significant impact on the communication efficiency. Although the Non-interactive VFL has lowest cost, the scheme requires that the dual-servers exchange an aggregation coded matrix and recover the same aggregation gradient used for cross-verification. It leads to a greater security threat of collusion attacks. However, our VOSA adopts the pairing-based cryptography to design a verifiable and oblivious secure aggregation protocol, which has a better balance between privacy preservation and performance. Therefore, the proposed VOSA scheme is

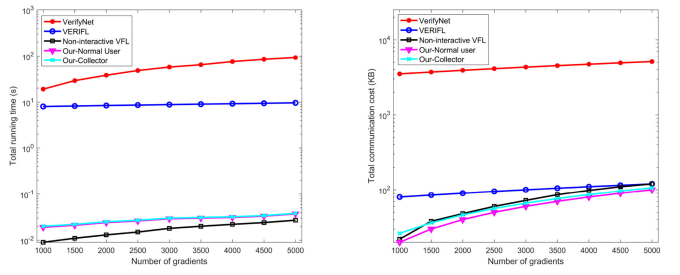


Fig. 12. Performance comparison of each user. (a) Computational overhead of each user, with different amount of gradients. (b) Communication overhead of each user, with different amount of gradients.

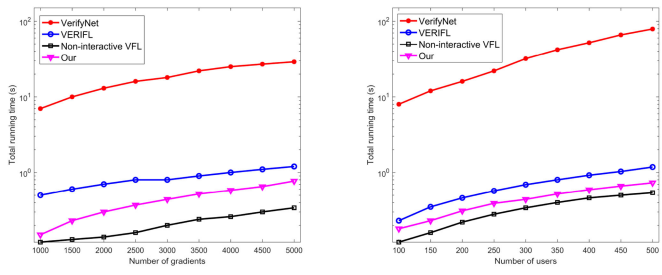


Fig. 13. Computational overhead comparison of the aggregation server. (a) $|\mathcal{U}| = 100$, with the different amount of gradients. (b) $|\mathcal{G}| = 1 \times 10^3$, with the different amount of users.

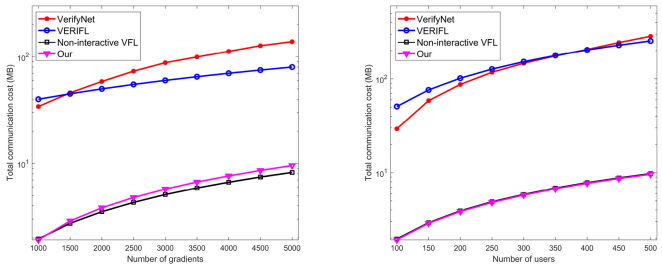


Fig. 14. Communication overhead comparison of the aggregation server. (a) $|\mathcal{U}| = 100$, with the different amount of gradients. (b)$|\mathcal{G}| = 1 \times 10^3$, with the different amount of users.

more suitable for large-scale verifiable secure aggregation scenarios in federated learning.

8 Summary and Future Work

In this paper, we present a verifiable and oblivious secure aggregation protocol for privacy-preserving federated learning with large-scale resource-constrained devices. Our scheme departs from prior works by using a novel designed aggregator oblivious encryption. It realizes lightweight encryption and aggregation as well as users dropping out tolerance without influence of their participation in future rounds. Furthermore, users can dynamic join in our protocol by the dynamic group management mechanism. Our work also provides security analysis and proof to ensure that the proposed protocol guarantees verifiable secure aggregation against malicious adversary. The experimental results show that our scheme is computation and communication efficient. It has better performance than the compared schemes. In this work, we mainly focus on resisting the attacks of malicious aggregation server while ignoring the malicious users scenarios. As part of future research work, we will consider how to detect and prevent malicious attacks on user side under this system architecture, and combine the incentive mechanism to balance the interests of each party.

References

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[2] W. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, Third Quarter 2020.

[3] A. Hard et al., "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*.

[4] T. Li, A. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[5] V. Mothukuri et al., "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, 2021.

[6] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1C2, pp. 1–210, 2021.

[7] E. Bagdasaryan et al., "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[8] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1322–1333.

[9] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.

[10] D. Wu, M. Pan, Z. Xu, Y. Zhang, and Z. Han, "Towards efficient secure aggregation for model update in federated learning," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.

[11] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.

[12] S. Truex et al., "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 1–11.

[13] C. Zhang et al., "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. USENIX Annu. Tech. Conf.*, 2020, pp. 493–506.

[14] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.

[15] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 911–926, 2020.

[16] C. Hahn, H. Kim, M. Kim, and J. Hur, "VERSA: Verifiable secure aggregation for cross-device federated learning," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2021.3126323.

[17] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. Vincent Poor, "Federated learning for Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1622–1658, Third Quarter 2021.

[18] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for Internet of Things: Recent advances, taxonomy, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1759–1799, Third Quarter 2021.

[19] L. Zhao, J. Jiang, B. Feng, Q. Wang, C. Shen, and Q. Li, "SEAR: Secure and efficient aggregation for Byzantine-robust federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 3329–3342, Sep./Oct. 2022.

[20] Z. Liu, J. Guo, K.-Y. Lam, and J. Zhao, "Efficient dropout-resilient aggregation for privacy-preserving machine learning," *IEEE Trans. Inf. Forensics Security*, to be published, doi: 10.1109/TIFS.2022.3163592.

[21] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet of Things J.*, vol. 9, no. 1, pp. 1–24, Jan. 2022.

[22] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet of Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020.

[23] S. Kadhe et al., "FastSecAgg: Scalable secure aggregation for privacy-preserving federated learning," 2020, *arXiv:2009.11248*.

[24] J. Song, W. Wang, T. R. Gadekallu, J. Cao, and Y. Liu, "EPPDA: An efficient privacy-preserving data aggregation federated learning scheme," *IEEE Trans. Netw. Sci. Eng.*, to be published, doi: 10.1109/TNSE.2022.3153519.

[25] B. Choi et al., "Communication-computation efficient secure aggregation for federated learning," 2020, *arXiv:2012.05433*.

[26] J. So, B. Gler, and S. Avestimehr, "Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 479–489, Mar. 2021.

[27] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[28] X. Guo et al., "VeriFL: Communication-efficient and fast verifiable aggregation for federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1736–1751, 2021.

[29] A. Fu, X. Zhang, N. Xiong, Y. Gao, H. Wang, and J. Zhang, "VFL: A verifiable federated learning with privacy-preserving for big data in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3316–3326, May 2022.

[30] Y. Zheng, S. Lai, Y. Liu, X. Yuan, X. Yi, and C. Wang, "Aggregation service for federated learning: An efficient, secure, and more resilient realization," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2022.3146448.

[31] Y. Xu et al., "Non-interactive verifiable privacy-preserving federated learning," *Future Gener. Comput. Syst.*, vol. 128, pp. 365–380, 2022.

[32] E. Shi, T.-H. Hubert Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Proc. Annu. Netw. Distrib. Syst. Secur. Symp.*, 2011.

[33] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2013, pp. 111–125.

[34] F. Benhamouda, M. Joye, and B. Libert, "A new framework for privacy-preserving aggregation of time-series data," *ACM Trans. Inf. Syst. Secur.*, vol. 18, no. 3, pp. 1–21, 2016.
[35] T. Chan, E. Shi, and D. Song, "Privacy-preserving stream aggregation with fault tolerance," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2012, pp. 200–214.
[36] M. Jawurek and F. Kerschbaum, "Fault-tolerant privacy-preserving statistics," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2012, pp. 221–238.
[37] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.
[38] K. Emura, "Privacy-preserving aggregation of time-series data with public verifiability from simple assumptions," in *Proc. Australas. Conf. Inf. Secur. Privacy*, 2017, pp. 193–213.
[39] I. Leontiadis, K. Elkhiyaoui, M. Onen, and R. Molva, "PUDA - Privacy and unforgeability for data aggregation," in *Proc. Int. Conf. Cryptol. Netw. Secur.*, 2015, pp. 3–18.
[40] I. Leontiadis, K. Elkhiyaoui, and R. Molva, "Private and dynamic time-series data aggregation with trust relaxation," in *Proc. Int. Conf. Cryptol. Netw. Secur.*, 2014, pp. 305–320.
[41] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 19–38.
[42] M. Fang et al., "Local model poisoning attacks to byzantine-robust federated learning," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 1605–1622.
[43] M. Hao, H. Li, G. Xu, H. Chen, and T. Zhang, "Efficient, private and robust federated learning," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2021, pp. 45–60.
[44] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4574–4588, 2021.
[45] J. Gao et al., "Secure aggregation is insecure: Category inference attack on federated learning," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2021.3128679.
[46] C. Niu, F. Wu, S. Tang, S. Ma, and G. Chen, "Toward verifiable and privacy preserving machine learning prediction," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 3, pp. 1703–1721, May/Jun. 2022.
[47] Z. Peng et al., "VFChain: Enabling verifiable and auditable federated learning via blockchain systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, pp. 173–186, Jan./Feb. 2022.
[48] B. Lynn, "PBC library manual 0.5.14," 2006. [Online]. Available: https://crypto.stanford.edu/pbc/manual.pdf

Yong Wang (Graduate Student Member, IEEE) received the BE and MS degrees from Anhui Normal University, Wuhu, China, in 2017 and 2021, respectively. He is currently working toward the PhD degree in physical information and intelligent systems with the School of Physical and Electronic Information Engineering, Anhui Normal University, Wuhu, China. His research interests include applied cryptography, blockchain, and federated learning.

Aiqing Zhang (Member, IEEE) received the MS degree in circuits and systems from Xiamen University, China, in 2006, and the PhD degree in signal and information processing from the Nanjing University of Posts and Telecommunications, China, in 2016. She is currently a professor with Anhui Normal University, China. She has authored more than 30 articles, and holds more than ten inventions. Her research interests include blockchain, applied cryptography, and wireless network security.

Shu Wu received the MS degree in pattern recognition and intelligent system from the Nanjing University of Posts and Telecommunications, China, in 2011. He is currently working toward the PhD degree in physical information and intelligence systems with the School of Physical and Electronic Information Engineering, Anhui Normal University, Wuhu, China. He is currently a lecturer with West Anhui University, China. He has more than 7 years of work experience in the relevant industries, he has authored nearly 10 articles. His research interests include healthcare block-chain, applied cryptography, and security of cyberspace.

Shui Yu (Fellow, IEEE) received the PhD degree from Deakin University, Australia, in 2004. He is currently a professor of the School of Computer Science, University of Technology Sydney, Australia. His research interest includes Big Data, security and privacy, networking, and mathematical modelling. He has published three monographs and edited two books, more than 400 technical papers, including top journals and top conferences, such as *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Emerging Topics in Computing*, *IEEE/ACM Transactions on Networking*, and INFOCOM. His h-index is 58. He initiated the research field of networking for big data, in 2013, and his research outputs have been widely adopted by industrial systems, such as Amazon cloud security. He is currently serving a number of prestigious editorial boards, including *IEEE Communications Surveys and Tutorials* (area editor), *IEEE Communications Magazine*, *IEEE Internet of Things Journal*, and so on. He is a member of AAAS and ACM, a distinguished lecturer of IEEE Communications Society, and an elected member of Board of Governor of IEEE Vehicular Technology Society.