VCD-FL: Verifiable, Collusion-resistant, and Dynamic Federated Learning

Sheng Gao, Jingjie Luo, Jianming Zhu, Xuewen Dong, and Weisong Shi, *Fellow, IEEE*

***Abstract*—Federated learning (FL) is essentially a distributed machine learning paradigm that enables the joint training of a global model by aggregating gradients from participating clients without exchanging raw data. However, a malicious aggregation server may deliberately return designed results without any operation to save computation overhead, or even launch privacy inference attacks using crafted gradients. There are only a few schemes focusing on verifiable FL, and yet they cannot achieve collusion-resistant verification. In this paper, we propose the first Verifiable, Collusion-resistant, and Dynamic FL (VCD-FL) to tackle this issue. Specifically, we first optimize Lagrange interpolation by gradient grouping and compression for achieving efficient verifiability of FL. To protect clients' data privacy against collusion attacks, we propose a lightweight commitment scheme using irreversible gradient transformation. By integrating the proposed efficient verification mechanism with the novel commitment scheme, our VCD-FL can detect whether or not the aggregation server is involved in collusion attacks. Moreover, considering that clients might go offline due to some reason such as network anomaly and client crash, we adopt the secret sharing technique to eliminate the effect of federation dynamics on FL. To the best of our knowledge, this is the first work to achieve collusion-resistant verification and collusion attack detection with supporting the correctness, privacy, and dynamics. Finally, we theoretically prove the effectiveness of our VCD-FL, make comprehensive comparisons, and conduct a series of experiments on MNIST dataset with MLP and CNN models. The theoretical proof and experimental analysis demonstrate that our VCD-FL is computationally efficient, robust against collusion attacks, and able to support the dynamics of FL.**

***Index Terms*—Federated learning, privacy preservation, verifiability, collusion-resistant, dynamics**

I. Introduction

A. Motivation

With the promotion of data privacy legislation, such as the General Data Protection Regulation [1], the California Consumer Privacy Act [2], and the Personal Information Protection Law [3], federated Learning (FL) has emerged as a distributed computing paradigm, which achieves collaborative model training with the advantages of data availability but invisibility [4]. Specifically, each client downloads global parameters, iteratively performs local model training with owned private data, and uploads the trained local gradient to the aggregation server (AS) for updating [5]. However, the shared gradients can be used to launch multiform inference attacks for exploiting clients' data privacy [6]–[8], such as *reconstruction attacks* for identifying sensitive attributes in the training dataset [9], [10] and *membership inference attacks* for judging whether or not a specified target is contained in the training dataset [11], [12].

To resist inference attacks in FL, existing work has proposed various techniques such as secure multiparty computation [13], [14] and differential privacy [15]–[17] to ensure gradient privacy. Nevertheless, most of them are built on a common assumption that the AS is honest-but-curious [18]–[20]. That is, it will not deviate from the pre-arranged operations but try to get some private information as possible. In fact, the AS would probably be malicious or corrupted by adversaries, which can arbitrarily deviate from the FL protocol by deliberately manipulating the training process for benefits [21]. Aside from inferring clients' privacy, it would threaten the correctness of the aggregated results and weaken the availability of the training model. For example, to save computation overhead, it might reduce the number of aggregation operations, or worse, return random results without any operation. Moreover, any client might go offline caused by some reason [22], such as network anomaly, crash, and power outage. These will have serious implications for the correctness of the aggregated results. To inveigle clients' privacy, it might collude with some corrupt clients to design crafted gradients for enticing specific privacy [19]. Therefore, the following fundamental issues in FL should be solved: (1) how to verify the correctness of the aggregated results while supporting the federation dynamics, and (2) how to protect clients privacy against collusion attacks.

To address the above issues, only a few schemes focusing on verifiable and private FL have been proposed. Xu *et al.* [18] first proposed VerifyNet, which supports the correctness, privacy, and dynamics in FL based on the homomorphic hash function integrated with pseudorandom technology and a designed double-masking protocol. To meet all these needs while overcoming the shortcoming that the communication overhead is linearly dependent on gradient dimension in VerifyNet [18], Guo *et al.* [20] proposed a communication-efficient protocol VeriFL, which optimizes the secure aggregation protocol in [22] by using linear homomorphic hash integrated with the equivocal commitment scheme. Fu *et al.* [19] proposed VFL, a verifiable, private, and collusion-resistant FL based on Lagrange interpolation and blinding technology. Although the overhead of the verification mechanism is independent of the number of clients, the computation and communication

This work was supported in part by Beijing Natural Science Foundation under Grant M21036, in part by the National Natural Science Foundation of China under Grant 62072487 and Grant 61972310, and in part by Zhejiang Provincial Natural Science Foundation under Grant LD22F020002.

S. Gao, J. Luo, and J. Zhu are with the School of Information, Central University of Finance and Economics, Beijing 100081, China (e-mail: sgao@cufe.edu.cn; JLuo1998@163.com; zjm@cufe.edu.cn).

X. Dong is with the School of Computer Science and Technology, Xidian University, Xi'an 710071, China (e-mail: xwdong@xidian.edu.cn).

W. Shi is with the Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA (e-mail: weisong@udel.edu).

overheads are still very expensive. In addition, all these works cannot achieve collusion-resistant verification or collusion attack detection. Specifically, the AS may collude with some corrupt clients to convince others of the manipulative aggregated results and ultimately pass verification. Furthermore, none of them can identify whether or not the AS is involved in collusion attacks. Maybe it is just a lazy server that provides aggregated results with low availability. By identifying the type of malicious behaviors, it will be more beneficial to take targeted safeguards for securing FL.

B. Our Contributions

In this paper, we propose VCD-FL, the first verifiable, collusion-resistant, and dynamic FL. To achieve collusion-resistant verification, we design a lightweight commitment scheme for gradients and an efficient verification mechanism based on optimized Lagrange interpolation to prevent those corrupt clients that collude with the AS from passing verification. Compared with Fu *et al.* [19], our VCD-FL can reduce the computation and communication overheads for verification by using gradient grouping and compression. Besides, our VCD-FL can detect collusion attacks whether or not the AS has been involved. To support the dynamics of FL, we also integrate the secret sharing technique into our designed mechanism. In conclusion, our contributions can be summarized as follows.

- *Collusion-resistant verification*. We propose a lightweight commitment scheme using irreversible gradient transformation to protect clients' privacy. To prevent the manipulative aggregated results from passing verification with an overwhelming probability, we design an efficient verification mechanism based on optimized Lagrange interpolation. Compared with existing works that only consider collusion-resistant privacy preservation, our VCD-FL can also achieve collusion-resistant verification.
- *Identifying malicious behavior*. Although existing studies can detect whether aggregated results are forged, they are able to do very little to reveal the underlying reasons. To make the security precautions more targeted, we establish malicious behavior detection rules, which can help defenders to determine if the AS is involved in collusion attacks for passing verification or if it is just a lazy server that returns incorrect results to save computation overhead.
- *Supporting federation dynamic*. Considering that some clients might go offline as a result of some reason such as network anomaly, crash, and power outage, we integrate our proposed verification mechanism with Shamir's threshold secret sharing scheme [23] for tolerating a certain number of clients dropping out. It can eliminate the effect of federation dynamics on FL, and take little impact on the privacy of the remaining clients.
- *Lower computation and communication overheads*. We reduce the computation overhead in [19] by designing a new method to generate interpolation points for Lagrange interpolation. Moreover, we further reduce the communication overhead by introducing the gradient compression algorithm [24]. Extensive experiments conducted on real-world data demonstrate that our VCD-FL is more practical.

C. Organization

The remainder of this paper is organized as follows. We briefly introduce some preliminaries in Section II. In Section III, we present the system overview of our VCD-FL. In Section IV, we elaborate on the system design of our VCD-FL. Theoretical analysis and experimental evaluation are respectively discussed in V and Section VI. In Section VII, we describe the related work. Finally, we conclude the paper in Section VIII.

II. Preliminaries

In this section, we present some preliminaries needed for the understanding of our VCD-FL. To facilitate readability, we list some main notations and their descriptions in Table I.

TABLE I: List of Notations

Notations	Descriptions
$\mathbb{P} = \{P_i\}_{i=1}^N$	the client set
D_i	the local dataset owned by P_i
$s_{i,j}$	a pairwise seed between P_i and P_j
ρ_i	an additional random seed of P_i
$\mathbb{A}_i = \{A_i(k)\}_{k=1}^{\lceil \frac{d}{M} \rceil}$	a pseudo-random sequence of P_i
$\mathbb{Z} = \{a_i\}_{i=1}^{\lceil \frac{d}{M} \rceil (M+1)}$	a random integer sequence
$\mathbf{g}_i, \mathbf{g}_{i,[k]}$	the raw gradient and the k-th grouped gradient
$\mathbf{g'}_i, \mathbf{g'}_{i,[k]}$	the noised gradient and the k-th grouped gradient
$\boldsymbol{U}$	a singular square matrix with $M \times M$
$\mathbf{C}_{i,[k]}$	the commitment of $\mathbf{g}_{i,[k]}$
$f_{i,[k]}(x), f'_{i,[k]}(x)$	the correct and false Lagrange interpolation function
$\mathbf{B}_{i,[k]}$	coefficients of $f_{i,[k]}(x)$
$b_{j,\langle i,[k] \rangle}$	the j-th coefficient in $\mathbf{B}_{i,[k]}$
$f_{[k]}(x)$	the aggregated interpolation function of the k-th group
$\mathbf{R}$	the aggregated random vector
$\mathbf{S}$	the result of $\mathbf{R} \cdot \boldsymbol{U}$
V_i	the verification value of P_i

* In this paper, we use (x) to represent the input of the interpolation function, $[k]$ to represent the group number of the k-th group, $\langle i, [k] \rangle$ to represent the k-th group of client P_i.

A. Federated Learning

FL is a distributed machine learning framework, which enables clients to collaboratively train a joint global model without sharing each local dataset [4], [5], [25]. Specifically, suppose there is a set $\mathbb{P} = \{P_i \mid i = 1, 2, \cdots, N\}$ with N clients and each client $P_i \in \mathbb{P}$ owns its private dataset D_i. At each iteration t, each client P_i downloads the latest global model $\mathbf{w}^{t-1}$ from the AS and iteratively conducts local model updating with stochastic gradient descent (SGD) algorithm [26] as

$$\mathbf{w}_i^t = \mathbf{w}^{t-1} - \eta_i \mathbf{g}_i^{t-1}, \tag{1}$$

where η_i is the local learning rate and $\mathbf{g}_i^{t-1} = \nabla_{\mathbf{w}} \ell(\mathbf{w}^{t-1}; d_i)$, where $\ell(\cdot)$ is the loss function on a single randomly example $d_i \in D_i$ with a variety of forms [25]. To improve the convergence rate, it is usually to use a mini-batch example $D'_i \subseteq D_i$ to compute the stochastic gradient. Finally, the AS collects those updated local models and aggregates them with the common FedAvg [4] as

$$\mathbf{w}^t = \sum_{i=1}^{N} \frac{|D_i'|}{|\sum_{i=1}^{N} D_i'|} \mathbf{w}_i^t. \tag{2}$$

B. Lagrange Interpolation

Lagrange interpolation refers to a method that can construct a polynomial accurately through all those given data points. Formally, let a set of n data points be $\{(x_i, y_i)\}_{i=1}^n$, where x_i for $i \in \{1, 2, \cdots, n\}$ are all distinct, we can fit a unique polynomial with the degree no greater than $n-1$ as

$$L(x) = \sum_{i=0}^{n} y_i L_i(x), \tag{3}$$

where the polynomials $L_i(x)$ are defined as

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}, \ i \in \{1, 2, \ldots, n\}. \tag{4}$$

Obviously, $L_i(x)$ has the property of

$$L_i(x_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{5}$$

Thus, the Lagrange polynomial $L(x)$ satisfies that $L(x_i) = y_i$. Recall that Fu *et al.* [19] first proposed to use Lagrange polynomial to verify the aggregated results from the AS in FL. They split each blind gradient into m parts as interpolation values and a random integer sequence is generated as the corresponding interpolation point. For the gradient with d-dimension, it needs to do splitting operations with md times and calculate d interpolation polynomials, which would cause significant computation overhead. Therefore, we reduce the computation and communication overheads by optimizing Lagrange interpolation using gradient grouping and compression for achieving efficient validation.

C. Commitment Scheme

A commitment scheme is a general function, which enables a committer to commit a message for verification without revealing any details. Specifically, it takes as inputs the message to be committed and a one-time pad, and as output a commitment to be publicly posted on a bulletin board. The one-time pad acts as the decommitment, which should be kept secret until commitment opening. Any compute-bound verifier believes the commitment by checking its correctness with the committed message and one-time pad. Note that exiting works [18], [20] use homomorphic hash commitment to achieve verifiable FL, which still incurs high computational complexity. In this paper, we design a lightweight commitment scheme for gradients by irreversible gradient transformation while protecting clients' privacy.

III. System Overview

In this section, we first introduce the design goals of our VCD-FL, and then provide an overview of the system model of VCD-FL and define the threat model.

A. Design Goals

To address the issues mentioned in Section I-A, we aim to design verifiable, collusion-resistant, and dynamic FL. The main design goals of our VCD-FL are as follows.

- *Robust result verification.* Our VCD-FL should guarantee the robustness of correctness verification, which can support not only collusion-resistant privacy preservation but also collusion-resistant verification. In addition, it should guarantee the correctness of dynamic FL caused by clients dropping out unexpectedly.
- *Malicious behavior classification.* Our VCD-FL should discover the underlying reasons for the incorrect aggregation result, which is helpful for taking targeted punishments and measures. That is to identify whether the AS is lazy for saving overhead or the AS colludes with some corrupt clients for collusion attacks.
- *Efficient model operations.* Our VCD-FL should enable clients to efficiently perform model operations including local model training and aggregated result verification. Besides, the communication overhead should be reduced to accelerate the operations somehow.
- *Lightweight privacy preservation.* Our VCD-FL should protect clients' privacy against inference attacks and collusion attacks while reducing some computation-intensive operations to enhance practicality.

B. System Model

For the design goals, we depict the system model of our proposed VCD-FL in Fig. 1, which is consisted of three entities, namely the Trusted Authority (TA), the Clients, and the AS.

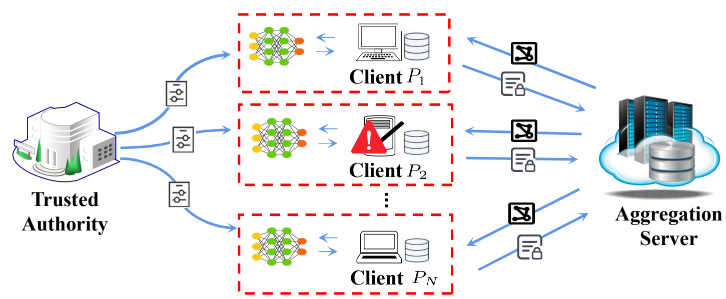


Fig. 1. System model of VCD-FL

- *TA* is mainly responsible for system initialization, which takes PRG as the pseudo-random generator and distributes parameters used in our VCD-FL to clients, including a pairwise seed, a singular square matrix, a pseudo-random vector sequence, and an integer sequence. It is considered to be trustworthy, which will neither participate in the federated training nor leak related private information.
- *Clients* with some common interest can join together for a specific model. Each client first downloads global parameters from the AS, then performs local model training on its owned private dataset, and finally uploads

the coefficients of grouped Lagrange polynomials as ciphertexts to the AS. Once the AS returns the aggregated result, each client can verify its correctness and decide whether to accept or reject the update. Note that clients that try to get sensitive information are honest-but-curious and any of them might drop out during the training.

- *AS* takes charge of ciphertexts collection and aggregation, and then distributes the aggregated result in each iteration to clients for verification. It is deemed to be malicious, which would launch inference attacks for prying into privacy or forgery attacks for disrupting availability.

C. Threat Model

In our VCD-FL, we define the threat model as that the TA is *trustworthy*, clients are *honest-but-curious*, and the AS is *malicious*. Specifically, the TA is only to generate and distribute parameters, which will not collude with others to reveal clients' privacy. Clients strictly perform operations in accordance with the pre-defined FL protocol, but try to infer some private information during the model training [18], [20]. The AS is considered to be an active adversary, which is out of control and manipulates aggregated results to disrupt model availability. Here, we sort the capabilities of the AS into two categories as follows.

- *Weak attack models.* The AS with weak capabilities is just to be a lazy server, which reduces the number of iterations or just aggregates partially collected gradients to save computation overhead. It would launch inference attacks to determine if the raw training dataset contains some specific data or even reconstruct sensitive attributes.
- *Strong attack models.* The AS with strong capabilities would try its best to hide the modifications to the aggregated result. It would collude with some clients to falsify the aggregated result to deceive others. Even worse, it might inveigle clients to expose more private information by using a well-designed aggregated result.

IV. Our VCD-FL Constructions

In this section, to overcome existing schemes that cannot achieve collusion-resistant verification and collusion attack detection, we detail our VCD-FL constructions for implementing the design goals in Fig. 2. Specifically, the main steps of our VCD-FL consist of initialization, local model training, ciphertext aggregation, and aggregated result verification.

A. Initialization

To begin with, TA initializes FL profiles and generates parameters needed in our VCD-FL, which is summarized in Algorithm 1. We notice that we can reduce the overhead of initialization in [18], [20], [22] by removing the negotiation with key agreement. That is, TA directly generates a pairwise seed $s_{i,j}$ between any two clients P_i and P_j for masking the gradient. To deal with the dropout problem, TA first generates an additional random seed ρ_i for P_i and then distributes shares of ρ_i to each client by using Shamir's threshold secret sharing scheme [23]. To enhance the interpolation accuracy by narrowing the scale gap between gradients and random numbers, a

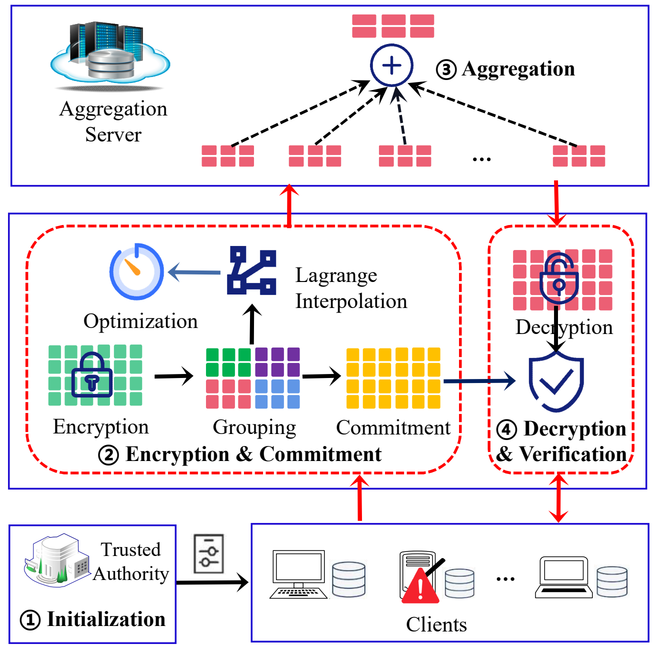


Fig. 2. Overview of our VCD-FL construction

Algorithm 1: Initialization

Input: PRG.
Output: $s_{i,j}$, $\mathbb{A}_i$, $\mathbb{Z}$, shares of ρ_i, $\boldsymbol{U}_{M\times M}$.

Generate a pairwise seed $s_{i,j}$ between P_i and P_j;
Generate an additional random seed ρ_i for P_i;
Compute a normalized sequence $\mathbb{A}_i$ for P_i as $\mathbb{A}_i \leftarrow \frac{\mathsf{PRG}(\rho_i)}{\max\{|\mathsf{PRG}(\rho_i)|\}}$;
for $j = 1$ *to* N **do**
  Get T-out-of-N shares of ρ_i from TA as $\{(P_j, \rho_{ij})\}_{P_j \in \mathbb{P}} \leftarrow \mathsf{Share}(T, \mathbb{P}, \rho_i)$;
end
Generate a random integer sequence $\mathbb{Z}$ and a singular square matrix U;

---

sequence set $\mathbb{A}_i$ generated by $\mathsf{PRG}(\rho_i)$ for verification should be normalized as

$$\mathbb{A}_i \leftarrow \frac{\mathsf{PRG}(\rho_i)}{\max\{|\mathsf{PRG}(\rho_i)|\}}, i \in \{1, 2, \ldots, N\}. \tag{6}$$

Here, we improve the VFL [19] by grouping gradient elements instead of splitting them. Each gradient $\mathbf{g}_i$ with d-dimension is divided into $\lceil \frac{d}{M} \rceil$ groups, each group contains M gradient elements. If the number of gradient elements in the last group is less than M, we will add padding with 0 to the rest. To make our VCD-FL verifiable, TA needs to generate a random integer sequence $\mathbb{Z} = \{a_i | i = 1, 2, \cdots, \lceil \frac{d}{M} \rceil (M+1)\}$ as the interpolation point set and a singular square matrix $\boldsymbol{U}$ with $M \times M$ for commitment generation.

B. Local Model Training

In this phase, each client $P_i \in \mathbb{P}$ first initializes its local model by downloading the latest global model, and then

Algorithm 2: Encryption and Commitment

Input: $\mathbf{g}_i$, $\boldsymbol{U}$, $s_{i,j}$, $\mathbb{A}_i$.
Output: Coefficient $\mathbf{B}_i$, Commitment $\mathbf{C}_i$.

Blind the gradient $\mathbf{g}_i$ as

$$\mathbf{g'}_i \leftarrow \mathbf{g}_i + \sum_{P_i \in \mathbb{P}, i<j} \mathsf{PRG}(s_{i,j}) - \sum_{P_i \in \mathbb{P}, i>j} \mathsf{PRG}(s_{i,j});$$

Divide $\mathbf{g}_i$ and $\mathbf{g'}_i$ into $\lceil \frac{d}{M} \rceil$ groups, where the k-group are $\mathbf{g}_{i,[k]}$ and $\mathbf{g'}_{i,[k]}$, where $k \in \{1, 2, \cdots, \lceil \frac{d}{M} \rceil\}$;
if $|\mathbf{g}_{i,[\lceil \frac{d}{M} \rceil]}| < M$ *or* $|\mathbf{g'}_{i,[\lceil \frac{d}{M} \rceil]}| < M$ **then**
  Add the padding with 0 to the rest;
end
for $k = 1$ *to* $\lceil \frac{d}{M} \rceil$ **do**
  Compute the k-th group commitment $\mathbf{C}_{i,[k]}$ as $\mathbf{C}_{i,[k]} \leftarrow \boldsymbol{U} \cdot \mathbf{g}_{i,[k]}$;
  Generate the k-th group Lagrange interpolation set as $\{(a_{(k-1)(M+1)+j}, \mathbf{g'}_i((k-1)(M+1) + j)), (a_{k(M+1)}, A_i(k))\}$, where $j \in \{1, 2, \cdots, M\}$;
  Perform Lagrange interpolation to get $f_{i,[k]}(x)$ as $f_{i,[k]}(x) \leftarrow \sum_{j=(k-1)(M+1)+1}^{k(M+1)-1} L_{j,[k]}(x)\mathbf{g'}_i(j) + L_{k(M+1),[k]}(x)A_i(k)$, where $L_{j,[k]}(x) = \prod_{h=(k-1)(M+1)+1, h\neq j}^{kM+k} \frac{x-a_h}{a_j-a_h}$;
  Extract coefficients $\mathbf{B}_{i,[k]}$ of the k-th group interpolation function in ascending order as $\mathbf{B}_{i,[k]} \leftarrow \left(b_{0,\langle i,[k]\rangle}, b_{1,\langle i,[k]\rangle}, \ldots, b_{M,\langle i,[k]\rangle}\right)$;
end
return $\mathbf{B}_i = (\mathbf{B}_{i,[k]})_{k=1}^{\lceil \frac{d}{M} \rceil}$, $\mathbf{C}_i = (\mathbf{C}_{i,[k]})_{k=1}^{\lceil \frac{d}{M} \rceil}$.

iteratively performs local model training on $D'_i \subseteq D_i$ with mini-batch gradient descent to compute the gradient $\mathbf{g}_i$ as

$$\mathbf{g}_i = \nabla_{\mathbf{w}} \ell(\mathbf{w}; D'_i), \tag{7}$$

where $\ell(\mathbf{w}; D'_i) = \frac{1}{|D'_i|} \sum_{(x_j, y_j) \in D'_i} (y_j - F(\mathbf{w}; x_j))^2$ and $F(\cdot)$ is the prediction function.

Then, P_i will perform gradient encryption, grouping, and commitment in turn, which are described in Algorithm 2.

*1) **Gradient Encryption**:* To achieve secure aggregation of gradients, we combine the single-masking protocol and optimized Lagrange interpolation to protect gradient privacy against collusion attacks. Inspired by [18], [22], based on the ordered subscripts of clients, we first use the single-masking protocol to blind each client P_i's local gradient $\mathbf{g}_i$ as

$$\mathbf{g'}_i = \mathbf{g}_i + \sum_{P_i \in \mathbb{P}, i<j} \mathsf{PRG}(s_{i,j}) - \sum_{P_i \in \mathbb{P}, i>j} \mathsf{PRG}(s_{i,j}). \tag{8}$$

It should be pointed out that compared with [18], [22], our VCD-FL employs the TA to directly distribute the pairwise seed $s_{i,j}$, which removes the complicated negotiations among clients. Moreover, it can resist inference attacks caused by the leakage of the original gradient due to the dropout misjudgment and threshold secret sharing in [18], [20].

Then, each client P_i leverages the advantages of Lagrange interpolation to deal with the blinded gradient for collusion-resistant verification. We improve the VFL [19] by grouping gradient elements rather than splitting them. Specifically, we adopt the same partition method described in Section IV-A to group the blinded gradient $\mathbf{g'}_i$. Each client P_i generates the k-th grouped Lagrange interpolation set as that the first M points are $\{(a_{(k-1)(M+1)+j}, \mathbf{g'}_i((k-1)(M+1)+j)) \mid j = 1, 2, \cdots, M\}$ and the $(M+1)$-th point is $(a_{k(M+1)}, A_i(k))$, where $k \in \{1, 2, \cdots, \lceil \frac{d}{M} \rceil\}$. Therefore, the function $f_{i,[k]}$ is computed on the k-th grouped Lagrange interpolation set as

$$f_{i,[k]}(x) = \sum_{j=(k-1)(M+1)+1}^{k(M+1)-1} L_{j,[k]}(x)\mathbf{g'}_i(j) + L_{k(M+1),[k]}(x)A_i(k), \tag{9}$$

where $L_{j,[k]}(x) = \prod_{h=(k-1)(M+1)+1, h\neq j}^{kM+k} \frac{x-a_h}{a_j-a_h}$.

Finally, according to the group indication, P_i uploads the assembled coefficient vector $\mathbf{B}_i$ as the gradient ciphertext to the AS as

$$\mathbf{B}_i = \left(\mathbf{B}_{i,[1]}, \mathbf{B}_{i,[2]}, ..., \mathbf{B}_{i,[\lceil \frac{d}{M} \rceil]}\right),$$

where each $\mathbf{B}_{i,[k]}$ denotes these $M+1$ coefficients extracted from $f_{i,[k]}(x)$ in ascending order according to x as

$$\mathbf{B}_{i,[k]} = (b_{0,\langle i,[k]\rangle}, b_{1,\langle i,[k]\rangle}, \ldots, b_{M,\langle i,[k]\rangle}).$$

Obviously, the confidentiality of the Lagrange interpolation set can enhance gradient privacy. Even if it leaks, as long as at least two clients do not collude with the AS, our VCD-FL can guarantee the gradient can hardly be deduced. The reason is that $s_{i,j}$ cannot be known to derive the gradient based on equation (8). Moreover, given the d-dimensional gradient, the number of Lagrange polynomials for interpolation is about $(M+1)\lceil \frac{d}{M} \rceil$, while it is $(M+1)d$ in the VFL [19]. Because our VCD-FL does grouping instead of splitting, the computation and communication overheads can be reduced. More details will be discussed in Section VI.

*2) **Commitment Generation**:* To protect gradient privacy while preventing those corrupt clients from proselytizing during the correctness verification of the aggregated result, we propose a lightweight commitment scheme, which reduces heavy computations in [18], [20] by using irreversible gradient transformation instead of cryptographic proof. Considering that the matrix $\boldsymbol{U}$ will be large if the gradient dimension d is big, we first divide $\mathbf{g}_i$ into $\lceil \frac{d}{M} \rceil$ groups, where each group contains M gradient elements. If the number of gradient elements in the $\lceil \frac{d}{M} \rceil$-th group is less than M, it will be filled with 0 to the rest. That is, $\mathbf{g}_i = (\mathbf{g}_{i,[1]}, \mathbf{g}_{i,[2]}, \ldots, \mathbf{g}_{i,[\lceil \frac{d}{M} \rceil]})$, where $\mathbf{g}_{i,[k]} = (\mathbf{g}_i((k-1)M+1), \mathbf{g}_i((k-1)M+2), \ldots, \mathbf{g}_i(kM))^T$.

Next, each client P_i makes the commitment for $\mathbf{g}_{i,[k]}$ as

$$\mathbf{C}_{i,[k]} = \boldsymbol{U} \cdot \mathbf{g}_{i,[k]}, \tag{10}$$

where $\boldsymbol{U}$ with $M \times M$ is irreversible for ensuring the gradient privacy and $\mathbf{g}_{i,[k]}$ is the M-dimensional column vector of the k-th gradient group.

It is important to note that the multiplication computation for commitment generation is lightweight and can be further

Algorithm 3: Interpolation Optimization

Input: $\mathbf{g}_i$, $\mathbf{G}_i$.
Output: Optimized $\mathbf{g}_i$.

if *epoch*=1 **then**
  Initialize $\mathbf{G}_i$ as $\mathbf{G}_i \leftarrow \mathbf{0}_{d\times 1}$;
end
Compute $\mathbf{G}_i$ as $\mathbf{G}_i \leftarrow \mathbf{g}_i + 0.5 \cdot \mathbf{G}_i$;
Select $p\%$ gradient elements from $\mathbf{G}_i$ as $\mathbf{g}_i \leftarrow \mathsf{Sparse}(\mathbf{G}_i, p\%)$;
Set the rest elements in $\mathbf{g}_i$ to 0;
Update $\mathbf{G}_i$ as $\mathbf{G}_i \leftarrow \mathbf{G}_i - \mathbf{g}_i$;
return $\mathbf{g}_i$.

processed in parallel, which can significantly increase efficiency. Subsequently, P_i will broadcast $\mathbf{C}_i = (\mathbf{C}_{i,[k]})_{k=1}^{\lceil \frac{d}{M} \rceil}$ and receive commitments from other clients before uploading $\mathbf{B}_i$ to the AS, which cannot convince those honest clients to accept the forged result. More proof will be presented in Section V.

*3) **Interpolation Optimization**:* To reduce the interpolation frequency while not compromising the model accuracy, we introduce deep gradient compression [24] to get an optimized gradient. Lagrange interpolation will be performed on top of the gradient sparsification. The interpolation optimization is described in Algorithm 3. Specifically, we adopt the same way proposed in [24] to compute the cumulative gradient $\mathbf{G}_i$. To solve the staleness issue, we generally use a momentum factor with 0.5 to compute $\mathbf{G}_i$ as

$$\mathbf{G}_i = \mathbf{g}_i + 0.5 \cdot \mathbf{G}_i, \tag{11}$$

where the initial value of $\mathbf{G}_i$ is set to 0.

Afterward, each client P_i will get the optimized gradient $\mathbf{g}_i$ for the input of Algorithm 2 by selecting $p\%$ elements from $\mathbf{G}_i$ with larger absolute values. The selected elements are put in the same place in $\mathbf{g}_i$, and the rest elements in $\mathbf{g}_i$ are set to 0. To avoid losing information, each unselected element from $\mathbf{G}_i$ will accumulate locally until its absolute value is large enough. Those selected elements in $\mathbf{G}_i$ will be reset to 0. Apparently, the optimized gradient $\mathbf{g}_i$ will greatly reduce the interpolation computation overhead. Because those interpolation points with $\mathbf{g'}_i(j) = 0$ will have no effect upon $\mathbf{B}_{i,[k]}$, P_i only needs to compute $L_{j,[k]}(x)$ with $\mathbf{g'}_i(j) \neq 0$. Compared with Algorithm 2 in which P_i originally requires compute $L_{j,[k]}(x)$ of $M+1$ times in total, our Algorithm 3 makes P_i only need to compute $L_{j,[k]}(x)$ of $\frac{p\% \cdot M+1}{M+1}$ times in total. More details on overhead comparisons will be discussed in Section V.

In addition, we find that the expensive Lagrange interpolation operation in gradient encryption also incurs high overhead. Therefore, we propose grouping gradient elements instead of splitting them to reduce the interpolation frequency. Theoretically, the interpolation computation overhead in our VCD-FL is about $\frac{1}{M}$ of the VFL [19] under the same M, where M is an integer that determines the degree of Lagrange interpolation function.

C. Ciphertext Aggregation

We consider that the ciphertext aggregation operation runs in a synchronous network. That is, the AS will perform aggregation until it receives the ciphertext $\mathbf{B}_i$ from each client P_i and compute $\mathbf{B}$ as

$$\mathbf{B} = \sum_{i=1}^{N} \mathbf{B}_i = (\sum_{i=1}^{N} \mathbf{B}_{i,[1]}, \ldots, \sum_{i=1}^{N} \mathbf{B}_{i,[\lceil \frac{d}{M} \rceil]}). \tag{12}$$

Afterward, the AS distributes $\mathbf{B}$ to each client P_i. Note that because the ciphertext $\mathbf{B}_i$ is computed by $\mathbf{g'}_i$ and $\mathbb{A}_i$, our VCD-FL can guarantee the original gradient $\mathbf{g}_i$ not being inferred only if the AS collude with no more than $N-2$ clients. Compared with [18], [20], our VCD-FL can overcome the privacy leakage issue of the single-masking protocol while supporting some clients who drop out for some reason during the training process. That is because our VCD-FL adopts threshold secret sharing [23] to ρ_i rather than $s_{i,j}$. According to equation (8), even if the AS colludes with $N-2$ clients, it can hardly get the $s_{i,j}$ between the remaining two clients, which can prevent the AS from getting the gradient.

D. Aggregated Result Verification

In this phase, each client P_i uses the received $\mathbf{B}$ to get the gradient aggregated result and verifies its correctness with previous commitments. The overall verification process of our VCD-FL is summarized in Algorithm 4.

*1) **Gradient Decryption**:* To get the aggregated result of gradients, P_i first reconstructs the aggregated interpolation function $f_{[k]}(x)$ of the k-th group with $\mathbf{B}_{[k]}$ as

$$f_{[k]}(x) = \sum_{m=1}^{M+1} \mathbf{B}_{[k]}(m) x^{M-m+1}, \tag{13}$$

where $\mathbf{B}_{[k]}(m)$ denotes the m-th element in $\mathbf{B}_{[k]}$ and $k \in \{1, 2, \ldots, \lceil \frac{d}{M} \rceil\}$.

Then, P_i reconstructs the aggregated result $\mathbf{g}$ of gradients with $f_{[k]}(x)$ by taking the integer sequence $\mathbb{Z}$ as input, and removing the inserted sequence $\mathbb{A}_i$ and the padding with 0 in the $\lceil \frac{d}{M} \rceil$-th group if exits. That is,

$$\begin{aligned} \mathbf{g} &= \sum_{i=1}^{N} \mathbf{g}_i = \sum_{i=1}^{N} \mathbf{g'}_i \\ &= ((f_{[k]}(a_{(k-1)M+k}), \ldots, f_{[k]}(a_{k(M+1)-1}))), \end{aligned} \tag{14}$$

where $k \in \{1, 2, \ldots, \lceil \frac{d}{M} \rceil\}$.

*2) **Result Verification**:* To verify the correctness of the aggregated result $\mathbf{g}$ while protecting gradient privacy, the basic idea is that judge whether the following equation actually holds,

$$\mathbf{R} \cdot \mathbf{g} = \sum_{i=1}^{N} \mathbf{R} \cdot \mathbf{g}_i, \tag{15}$$

where $\mathbf{R}$ is a d-dimensional vector.

Apparently, if $\mathbf{g} \neq \sum_{i=1}^{N} \mathbf{g}_i$, the equation is not satisfied unless $\mathbf{g}$ is crafted to be in the same hyper-plane. However, this

Algorithm 4: Decryption and Verification

Input: $\mathbf{B}$, $\mathbb{Z}$, $\mathbf{C} = \{\mathbf{C}_i\}_{i=1}^{N}$, $\mathbb{A} = \{\mathbb{A}_i\}_{i=1}^{N}$.
Output: The aggregated result $\mathbf{g}$.

for $k = 1$ *to* $\lceil \frac{d}{M} \rceil$ **do**
  Recover the aggregated function $f_{[k]}(x)$ as $f_{[k]}(x) \leftarrow \sum_{m=1}^{M+1} \mathbf{B}_{[k]}(m) x^{M+1-m}$;
  for $m = (k-1)M + 1$ *to* kM **do**
    Compute the aggregated gradient with $\mathbb{Z}$ as $\mathbf{g}(m) \leftarrow f_{[\lceil \frac{m}{M} \rceil]}(a_{m+\lceil \frac{m}{M} \rceil - 1})$;
  end
end
for $P_i \in \mathbb{P}$ **do**
  Generate a random vector $\mathbf{R}_i$;
  Broadcast $\mathbf{R}_i$ to the other clients;
  Compute $\mathbf{R}$ as $\mathbf{R} \leftarrow (\sum_{i=1}^{N} r_{i,1}, \sum_{i=1}^{N} r_{i,2}, \ldots, \sum_{i=1}^{N} r_{i,M\cdot\lceil \frac{d}{M} \rceil})$;
  Divide $\mathbf{R}$ into $\lceil \frac{d}{M} \rceil$ groups in the same way as $\mathbf{g}_i$;
  Compute the k-th group checksum $v_{i,k} \leftarrow \mathbf{R}_{[k]} \cdot \mathbf{C}_{i,[k]}$;
  Compute V_i for verification as $V_i \leftarrow \sum_{k=1}^{\lceil \frac{d}{M} \rceil} v_{i,k}$;
  Compute the k-group of $\mathbf{S}$ as $\mathbf{S}_{[k]} \leftarrow \mathbf{R}_{[k]} \cdot \boldsymbol{U}$;
  Check the following two equations
  $f_{[k]}(a_{(M+1)k}) \stackrel{?}{=} \sum_{i=1}^{N} A_i(k),\ k \in \{1, 2, \ldots, \lceil \frac{d}{M} \rceil\}$;
  $\sum_{m=1}^{d} \mathbf{S}(m)\mathbf{g}(m) \stackrel{?}{=} \sum_{i=1}^{N} V_i$;
  if Rule 1 *holds* **then**
    The AS is considered to be trustworthy;
    return $\mathbf{g}$;
  end
  if Rule 2 *holds* **then**
    The AS is considered to be a weak attacker;
    Exit;
  end
  if Rule 3 *holds* **then**
    The AS is considered to be a strong attacker;
    Exit;
  end
end

basic verification mechanism cannot resist collusion attacks. That is because $\mathbf{R}$ is not generated randomly in this case, and those corrupt clients in collusion can craftily design $\mathbf{R}$ or manipulate $\mathbf{R} \cdot \mathbf{g}_i$ to help the malicious AS bypass the verification.

To alleviate this issue, we design an efficient verification mechanism on the basis of the previously generated commitment. Specifically, P_i first distributes the other clients with $\mathbb{A}_i$ and a random row vector $\mathbf{R}_i = (r_{i,1}, r_{i,2}, \ldots, r_{i,M\cdot\lceil \frac{d}{M} \rceil})$. Then, P_i can compute $\mathbf{R}$ as

$$\mathbf{R} = \sum_{i=1}^{N} \mathbf{R}_i = (\sum_{i=1}^{N} r_{i,1}, \sum_{i=1}^{N} r_{i,2}, \ldots, \sum_{i=1}^{N} r_{i,M\cdot\lceil \frac{d}{M} \rceil}). \quad (16)$$

It is obvious that $\mathbf{R}$ is unpredictable and cannot be manipulated as long as any client does not take part in collusion. To prevent the corrupt clients from helping the malicious AS bypass the verification, P_i then groups $\mathbf{R}$ in the same way as that of gradient $\mathbf{g}_i$ and computes a random group row vector $\mathbf{S}$ as the validation coefficient vector as

$$\mathbf{S} = (\mathbf{S}_{[1]}, \mathbf{S}_{[2]}, \ldots, \mathbf{S}_{[\lceil \frac{d}{M} \rceil]}),$$

where the k-th group vector $\mathbf{S}_{[k]}$ is computed as $\mathbf{S}_{[k]} = \mathbf{R}_{[k]} \cdot \boldsymbol{U}$, and $\boldsymbol{U}$ is the same singular square matrix with $M \times M$.

The verification process is on the basis of the previously distributed commitment $\mathbf{C}_i$. To achieve efficient verification, P_i computes the checksum $v_{i,k}$ of each group in parallel. For the k-th group, $v_{i,k}$ can be computed as

$$v_{i,k} = \mathbf{R}_{[k]} \cdot \mathbf{C}_{i,[k]},\ k \in \{1, 2, \ldots, \lceil \frac{d}{M} \rceil\}. \quad (17)$$

Finally, P_i computes V_i as $V_i = \sum_{k=1}^{\lceil \frac{d}{M} \rceil} v_{i,k}$ and releases $\mathbb{A}_i = \{A_i(k)\}_{k=1}^{\lceil \frac{d}{M} \rceil}$ for verification.

With the above information, our VCD-FL establishes the first set of malicious behavior detection rules to identify and differentiate the type of attack models as defined in Section III-C, which alleviates the ambiguous issue in [18]–[20] for targeted precautions. Specifically, P_i checks the following two equations, respectively as

$$f_{[k]}(a_{(M+1)k}) \stackrel{?}{=} \sum_{i=1}^{N} A_i(k),\ k \in \{1, 2, \ldots, \lceil \frac{d}{M} \rceil\}, \quad (18)$$

$$\sum_{m=1}^{d} \mathbf{S}(m)\mathbf{g}(m) \stackrel{?}{=} \sum_{i=1}^{N} V_i. \quad (19)$$

It is worth noting that $f_{[k]}(a_{(M+1)k})$ and $\mathbf{g}(m) = f_{[\lceil \frac{m}{M} \rceil]}(a_{m+\lceil \frac{m}{M} \rceil - 1})$ are calculated by the returned ciphertext $\mathbf{B}$ from the AS, and $\mathbf{S}(m)$ denotes the m-th element in $\mathbf{S}$. In short, equation (18) can be used to just first check the accuracy of the aggregation, and equation (19) serves to further validate whether or not the AS has colluded with clients. Therefore, our VCD-FL defines the rules as

- **Rule 1:** If both equation (18) and equation (19) hold, the AS is considered to be trustworthy.
- **Rule 2:** If both equation (18) and equation (19) do not hold, the AS is considered to be a weak attacker.
- **Rule 3:** If equation (18) holds and equation (19) does not hold, the AS is considered to be a strong attacker.

The entire verification process is summarized in Algorithm 4. We can conclude that the AS is trustworthy for passing the aggregation verification if and only if **Rule 1** holds. Otherwise, the aggregation verification fails, and the type of malicious AS with different attack capabilities defined by the attack models in Section III-C can be distinguished respectively using **Rule 2** and **Rule 3**. More detailed explanations will be proved in Section V.

E. Supporting Dynamic Verification

An important effect on the verification is the federation dynamics. Due to some reason such as network anomaly, crash, and power outage, there is a possibility that some clients might drop out during the training process. Existing works [18], [20], [22] have proposed to subtract off the masks before gradient aggregation with the double-masking protocol. However, they guarantee gradient privacy on the basis of the assumption that any client would never reveal both kinds of shares for the same client. It will be infeasible in our VCD-FL because some clients might collude with the AS. Our VCD-FL can guarantee privacy if no more than $N-2$ clients collude with the AS. Here, we analyze the verification process on the basis of equation (18) and equation (19) when some clients drop out during the training process.

In our VCD-FL, it is obvious that $f_{[k]}(a_{(M+1)k})$ and $\mathbf{g}$ are computed using $\mathbf{B}$ returned by the AS, which is not affected by offline clients. Due to the fact that the gradient commitment $\mathbf{C}_i$ has been distributed prior to the aggregation and $\mathbf{R}_i$ has already been broadcast, the computational process of $\mathbf{S}$ will not be impacted. When P_i has dropped out, to maintain the verification process with the remaining clients without being affected by the loss of $\mathbb{A}_i$, we use Shamir's threshold secret sharing scheme [23] to share ρ_i of P_i in the form of T-out-of-N. Each client $P_j \in \mathbb{P}$ can get a share ρ_{ij}. This allows ρ_i to be recovered even if P_i drops out during the verification, as long as the minimum number of clients remains alive is no less than T. Hence, even if some clients fail to send $\mathbb{A}_i$ on time, we can get $\mathbb{A}_i$ with the recovered ρ_i to verify whether or not equation (18) holds.

Furthermore, as long as there is a client that does not take part in collusion, a forged aggregated result can be detected by equation (19) in our VCD-FL. Because $\mathbf{S}$ generated by the unpredictable $\mathbf{R}$ is random, even $N-1$ clients collude with the AS to forge the aggregated result, it can hardly make equation (19) hold. A more detailed formal proof will be presented in Section V.

V. Theoretical and Comparative Analysis

In this section, we theoretically prove the effectiveness of our VCD-FL in terms of correctness, verifiability, collusion resistance, and dynamics. Afterward, we conduct a comprehensive comparison analysis with those related works.

A. Correctness

Our VCD-FL defines correctness as ensuring that clients get the correct aggregated result from the AS for updating their local models if each entity performs its operations honestly. More formally, we have the following **Theorem** 1.

Theorem 1: If the AS performs aggregation operations honestly in our VCD-FL, the correct aggregated result will pass the verification.

Proof: If the AS performs aggregation operations honestly in our VCD-FL, each client $P_i \in \mathbb{P}$ can obtain the correct aggregated result only if both equation (18) and equation (19) hold.

For equation (18), according to the correct $\mathbf{B}$ returned by the AS, each client P_i can recover the aggregated interpolation function of the k-th group $f_{[k]}(x)$. Because $f_{[k]}(x) = \sum_{i=1}^{N} f_{i,[k]}(x)$ and $A_i(k) = f_{i,[k]}(a_{(M+1)k})$ according to equation (9), we can get

$$f_{[k]}(a_{(M+1)k}) = \sum_{i=1}^{N} f_{i,[k]}(a_{(M+1)k}) = \sum_{i=1}^{N} A_i(k), \quad (20)$$

where $k \in \{1, 2, \ldots, \lceil \frac{d}{M} \rceil\}$.

For equation (19), we first compute the k-grouped aggregated gradients $\mathbf{g}_{[k]}$ as $\mathbf{g}_{[k]} = \sum_{i=1}^{N} \mathbf{g}_{i,[k]}$, where $\mathbf{g}_{i,[k]} = (f_{i,[k]}(a_{(k-1)M+k}), \ldots, f_{i,[k]}(a_{k(M+1)-1}))$. Then, we can compute

$$\begin{aligned}\sum_{i=1}^{N} V_i &= \sum_{i=1}^{N} \sum_{k=1}^{\lceil \frac{d}{M} \rceil} v_{i,k} = \sum_{i=1}^{N} \sum_{k=1}^{\lceil \frac{d}{M} \rceil} \mathbf{R}_{[k]} \cdot \mathbf{C}_{i,[k]} \\ &= \sum_{i=1}^{N} \sum_{k=1}^{\lceil \frac{d}{M} \rceil} \mathbf{S}_{[k]} \cdot \mathbf{g}_{i,[k]} = \sum_{k=1}^{\lceil \frac{d}{M} \rceil} \mathbf{S}_{[k]} \cdot \sum_{i=1}^{N} \mathbf{g}_{i,[k]} \\ &= \sum_{m=1}^{d} \mathbf{S}(m) f_{[\lceil \frac{m}{M} \rceil]}(a_{m+\lceil \frac{m}{M} \rceil -1}) + \sum_{m=d+1}^{\lceil \frac{d}{M} \rceil \cdot M} \mathbf{S}(m) \cdot 0 \\ &= \sum_{m=1}^{d} \mathbf{S}(m) f_{[\lceil \frac{m}{M} \rceil]}(a_{m+\lceil \frac{m}{M} \rceil -1}) = \sum_{m=1}^{d} \mathbf{S}(m)\mathbf{g}(m).\end{aligned} \quad (21)$$

Therefore, according to the deduction of equation (20) and equation (21), we can conclude that both equation (18) and equation (19) hold. That will mean **Rule 1** is satisfied. Therefore, if the AS performs aggregation operations honestly in our VCD-FL, the correct aggregated result will pass the verification. ■

B. Verifiability

Our VCD-FL defines verifiability as the ability of each client to independently verify the correctness of the aggregated result under the two defined attack models.

To distinguish the false result from the true $\mathbf{B}$ returned by the AS, here we use $\Delta\mathbf{B}$ ($\Delta\mathbf{B} \neq \mathbf{0}$) to represent the modification of the aggregated result. Therefore, the false result $\mathbf{B}'$ is $\mathbf{B}' = \mathbf{B} + \Delta\mathbf{B}$. According to the aforementioned instructions in Section IV-D, P_i recovers the false aggregated interpolation function $f'_{[k]}(x)$ of the k-th group with $\mathbf{B}'_{[m]} \in \mathbf{B}$ as

$$\begin{aligned}f'_{[k]}(x) &= \sum_{m=1}^{M+1} \mathbf{B}'_{[k]}(m) x^{M+1-m} \\ &= \sum_{m=1}^{M+1} (\mathbf{B}_{[k]}(m) + \Delta\mathbf{B}_{[k]}(m)) x^{M+1-m} \\ &= f_{[k]}(x) + \sum_{m=1}^{M+1} \Delta\mathbf{B}_{[k]}(m) x^{M+1-m},\end{aligned} \quad (22)$$

where $\mathbf{B}'_{[m]}$ represents the m-th group vector with $M+1$ elements and $k \in \{1, 2, \ldots, \lceil \frac{d}{M} \rceil\}$.

It is quite clear that the malicious AS can manipulate the aggregated result by adjusting $\Delta\mathbf{B}$ under the defined threat models in Section III-C. Therefore, we can draw the following **Theorem 2**.

Theorem 2: If the AS returns a false aggregated result in our VCD-FL, our detection rules can identify the false aggregated result under the defined threat models with an overwhelming probability.

Proof: If the AS attempts to evade the detection rules successfully, it needs to ensure both equation (18) and equation (19) hold with $f'_{[k]}(x)$.

For equation (18), according to equation (22), it is significant that it holds only if

$$\sum_{m=1}^{M+1} \Delta\mathbf{B}_{[k]}(m) a_{(M+1)k}^{M+1-m} = 0,\ k \in \{1, 2, \ldots, \lceil \frac{d}{M} \rceil\}. \quad (23)$$

If $a_{(M+1)k}$ is kept secret from the AS, it has been proved that equation (23) is impossible [19]. However, in our VCD-FL, those corrupt clients might collude with the AS to obtain $a_{(M+1)k}$, which makes equation (23) hold with an overwhelming probability by returning a crafted $\Delta\mathbf{B}$. Our VCD-FL can detect and identify the type of collusion behaviors, which will be proved in **Theorem** 3.

For equation (19), each client recovers $\mathbf{g}'$ with the returned $\mathbf{B}'$ from the AS, and the attackers aim to pass the verification by essentially making equation (24) hold, that is

$$\sum_{m=1}^{d} \mathbf{S}(m)\mathbf{g}'(m) = \sum_{i=1}^{N} V_i = \sum_{m=1}^{d} \mathbf{S}(m)\mathbf{g}(m). \quad (24)$$

Therefore, it is equivalent to

$$\sum_{m=1}^{d} \mathbf{S}(m)[\mathbf{g}'(m) - \mathbf{g}(m)] = 0, \quad (25)$$

where $\mathbf{g}'(m) - \mathbf{g}(m)$ represents the modification by the attackers, which can be controlled by the malicious AS. However, $\mathbf{S}$ is generated only after receiving the aggregated result from the AS. Because $\mathbf{R}$ is unpredictable, manipulating each element $\mathbf{S}(m) \in \mathbf{S}$ is nearly impossible. Therefore, as long as there is any client in $\mathbb{P}$ that does not collude with the malicious AS, the probability that equation (25) holds will be extremely low.

To sum up, we can conclude that once the malicious AS returns a false aggregated result, our detection rules can identify it with an overwhelming probability. The collusion identification will be explained in **Theorem** 3. ■

C. Collusion Resistance

To make the crafted aggregated result pass verification, the malicious AS might collude with some corrupt clients. This will make the VFL [19] fail due to the leakage of the interpolation sequences. We have demonstrated that our VCD-FL can guarantee gradient privacy only if the AS colludes with no more than $N-2$ clients. Here, we prove that our VCD-FL is collusion-resistant and capable of identifying the type of collusion behaviors, as presented in **Theorem** 3.

Theorem 3: If the AS colludes with $N-1$ clients at most, the forged aggregated result by collusion attacks can be detected in our VCD-FL with an overwhelming probability.

Proof: Without loss of generality, we assume that $\{P_i\}_{i=1}^{N'}$ collude with the AS, where $N' \leq N-1$. To make the forged aggregated result pass the verification, they are in collusion to forge some information to make equation (18) and equation (19) hold.

As proved in **Theorem** 2, equation (18) holds with an overwhelming probability by returning a crafted $\Delta\mathbf{B}$ with the exception of $N' = 0$. That is if there is no client in collusion, equation (18) holds with a negligible probability. Here, we prove that equation (19) is impossible even if $N' = N-1$ clients collude with the AS. To make equation (19) hold, the goal of N' clients in collusion is to control V' as

$$\begin{aligned} V' &= \sum_{m=1}^{d} \mathbf{S}(m)\mathbf{g}'(m) - \sum_{i=1}^{N'} V_i \\ &= \sum_{m=1}^{d} \mathbf{S}(m)\mathbf{g}'(m) - \sum_{i=1}^{N'} \sum_{m=1}^{d} \mathbf{S}(m)\mathbf{g}_i(m) \\ &= \sum_{m=1}^{d} \mathbf{S}(m)(\mathbf{g}'(m) - \sum_{i=1}^{N'} \mathbf{g}_i(m)). \end{aligned} \quad (26)$$

Recall that $\mathbf{S}$ depends on $\mathbf{R}$, which are calculated only after getting the false aggregated result $\mathbf{g}'$. It has been analyzed that $\mathbf{R}$ is unpredictable as long as a client in $\mathbb{P}$ at least does not take in collusion. Therefore, even if N' clients are in collusion to craft $\mathbf{g}'$, it is impossible to determine $\mathbf{S}$. As a result, it can control V' with a negligible probability even if $N' = N-1$ clients take in collusion.

Therefore, we can conclude that the detection rules in our VCD-FL can not only detect the forged aggregated result by collusion attacks with an overwhelming probability but also distinguish the type of attack models. ■

D. Dynamics

Dynamics in our VCD-FL refers to the fact that a certain percentage of clients dropping out would not affect the privacy of the remaining clients or the correctness of gradient aggregation verification. As for privacy, we have proved that our VCD-FL can guarantee the gradient $\mathbf{g}_i$ not being reverted as long as the number of clients in collusion is no more than $N-2$. Here we demonstrate the correctness, as shown in **Theorem** 4.

Theorem 4: If clients drop out during the verification process, our VCD-FL can still work as long as the number of dropped clients is no more than $N-T$.

Proof: Our VCD-FL works if and only if both equation (18) and equation (19) hold. For equation (18), if the number of dropped clients is no more than $N-T$, then our VCD-FL can recover ρ_i with T online clients using T-out-of-N threshold secret sharing [23]. According to equation (6), our VCD-FL can compute $\mathbb{A}_i$ for verifying the correctness of equation (18).

For equation (19), even if $N-T$ clients drop out, the online client can still calculate $\mathbf{R}$. That is because each client P_i has owned $\{\mathbf{R}_j\}_{j=1,j\neq i}^{N}$, which are distributed from the other clients. According to Algorithm 2, each client P_i makes a commitment $\mathbf{C}_i$ and distributes it to the others before uploading $\mathbf{B}_i$ to the AS, thus P_i can calculate V_i even if

TABLE II: Verifiable FL schemes: A comprehensive comparison

Requirements	VerifyNet [18]	VFL [19]	VeriFL [20]	Our VCD-FL
Privacy-preserving	Double-masking	Single-masking with two seeds	Double-masking	Single-masking with a seed
Verifiability	Homomorphic hash	Lagrange interpolation	Homomorphic hash	Lagrange interpolation and Commitment
Convergence stability	✔	✗	✔	✔
Collusion-resistant Verification	✗	✗	✗	✔
Collusion-detection	✗	✗	✗	✔
Dynamic Verification	✔	✗	✗	✔
High-accuracy	✔	✗	✔	✔

$N-T$ clients drop out according to equation (17). Likewise, according to Algorithm 4, the unpredictable $\mathbf{S}$ can also be computed. Finally, the equation (19) can be verified.

Therefore, we can conclude that our VCD-FL can effectively support dynamic verification as long as the number of dropped clients is no more than $N-T$. ■

E. Comparison

We compare our VCD-FL with exiting verifiable FL schemes [18]–[20], which is shown in Table II. To protect gradient privacy, our VCD-FL adopts the single-masking protocol with a seed rather than two seeds in [19] to blind raw gradients while reducing communication overhead. It alleviates the assumption that any client would never reveal both online shares and offline shares for the same client [22], and solves the privacy leakage issue in [18], [20] by applying threshold secret sharing [23] to ρ_i rather than $s_{i,j}$. To guarantee verifiability while protecting gradient privacy, we propose a lightweight commitment scheme, which reduces heavy computations in [18], [20] by using irreversible gradient transformation instead of cryptographic proof.

We find that all these works assume that the AS is honest-but-curious, which neglects collusion attacks during the verification process. Some corrupt clients might help the malicious AS to make the falsified aggregated result pass verification. To the best of our knowledge, our VCD-FL is the first work to achieve collusion-resistant verification and collusion attack detection. Besides, our VCD-FL can support dynamic verification as long as the number of dropped clients is no more than $N-T$, while guaranteeing the gradient $\mathbf{g}_i$ not being reverted as long as the number of clients in collusion is no more than $N-2$. Compared with [19], our VCD-FL can provide better convergence stability and higher accuracy. That is because the encoding scheme used in [19] will significantly reduce gradient precision.

VI. Evaluation

In this section, we evaluate the performance of our VCD-FL in terms of effectiveness, computation overhead, and communication overhead.

A. Experimental Setup

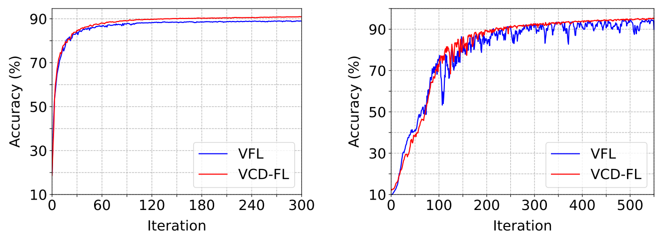
We conduct the performance evaluation of our VCD-FL based on a prototype implementation. Clients and the AS in our VCD-FL are simulated on a 64-bit laptop that has Inter(R) Core(TM) i7-9750H, 2.6GHz CPU, GTX 1660Ti GPU, and 16GB RAM based on Windows 10. Then, we implement our VCD-FL with Python 3.8.8, Pytorch 1.8.1, and Numpy 1.19.2. The local training process is simulated in a multi-processing manner.

We perform all the experiments on MNIST dataset [27] for classification tasks. MNIST is a handwritten image dataset, which contains 60,000 training samples and 10,000 test samples. Each sample is a digital grayscale image of 28×28 pixels, which represents a handwritten number between 0 and 9.

We take a multi-layer perceptron (MLP) and a convolution neural network (CNN) as the training model for our VCD-FL, respectively. Specifically, the architecture of the MLP is configured as three fully-connected layers with 784(input)-128(hidden)-10(output). The number of parameters for the MLP is (784+1)×128+(128+1)×10=101,770. The architecture of the CNN is configured as two convolution layers with 5×5 convolution kernels, where the first is with 10 channels and the second is with 20 channels, and each is followed by 2×2 max pooling layer. Following the convolutional layers, there is a fully connected layer with 50 neurons and an output layer. The number of parameters for the CNN is 21,780. We conduct the local model training on a mini-batch of 100 randomly selected samples to balance accuracy and efficiency.

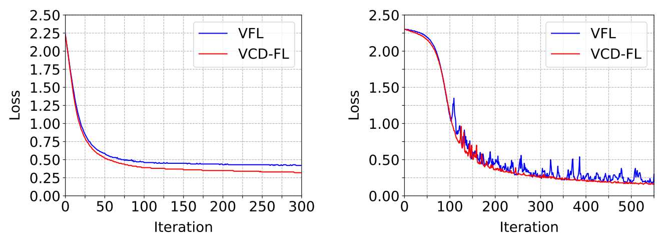
B. Effectiveness

According to Table II, only the VFL [19] and our VCD-FL adopt Lagrange interpolation to guarantee verifiability. Hence, we analyze the effectiveness of our VCD-FL in terms of accuracy and loss, as well as make comparisons with the VFL. Fig. 3 shows the accuracy under different iterations by taking the MLP and the CNN as the training model respectively. It can be seen that the accuracy of our VCD-FL has advantages over the VFL. Specifically, the accuracy of the MLP is shown in Fig. 3(a). After 300 iterations, the accuracy of our VCD-FL can reach about 90.92%, while the VFL is about 88.90%. Likewise, the accuracy of the CNN is shown in Fig. 3(b). After 500 iterations, the accuracy of our VCD-FL can reach about 94.83%, while the VFL is about 93.38%. There are two reasons for this. On one hand, the VFL adopts an encoding scheme that converts a gradient to an integer using a rounding method, which will reduce the gradient accuracy. On the other hand, our VCD-FL adopts deep gradient compression [24] to enhance accuracy and efficiency even further. Meanwhile, we find that compared with our VCD-FL, the VFL would cause a certain vibration phenomenon during the convergence process, especially for the CNN. This is also caused by the encoding scheme, which loses some gradient accuracy.



(a) (b)

Fig. 3. Accuracy comparison between the VFL [19] and our VCD-FL. (a) Accuracy of MLP. (b) Accuracy of CNN.



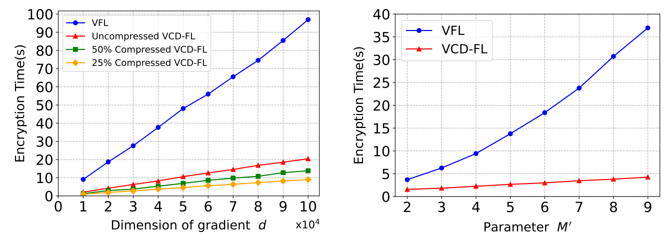
(a) (b)

Fig. 4. Loss comparison between the VFL [19] and our VCD-FL. (a) Loss of MLP. (b) Loss of CNN.

Besides, we also conduct experiments on the loss of the VFL [19] and our VCD-FL with the corresponding MLP and CNN. The loss is measured by the widely-used cross-entropy function for the multi-class classifier and the results are shown in Fig. 4. It can be seen that compared with the VFL, our VCD-FL causes less loss. On the whole, the loss of the MLP is shown in Fig. 4(a). After 300 iterations, the loss of our VCD-FL reduces to 0.323, while the VFL is about 0.422. Likewise, the loss of the CNN is shown in Fig. 4(b). After 500 iterations, the loss of our VCD-FL drops to 0.176, while the VFL is about 0.208. The reasons for these are the same as that for the accuracy, which have been discussed above.

C. Computation Overhead

As we described in Section IV-C, the AS that is responsible for ciphertext aggregation only needs to perform $\sum_{i=1}^{N} \mathbf{B}_i$, where $\mathbf{B}_i$ is uploaded by each $P_i \in \mathbb{P}$. Apparently, the computation overhead of the AS is trivial to our VCD-FL. Here, we mainly evaluate the computation overhead of our VCD-FL on clients in terms of encryption overhead and decryption overhead. Different from those schemes that upload encrypted gradients to the AS, our VCD-FL uploads the coefficient vectors as ciphertexts instead. Hence, we evaluate the computation overhead of a client dealing with a gradient that has different dimensions. It is worth noting that the computation overhead mainly relies on the complexity of the Lagrange interpolation process. To guarantee the fairness of comparison and describe conveniently, we uniformly mark $m-1$ in the VFL [19] and M in our VCD-FL as parameter M', where $m-1$ originally refers to the size of sequences for spitted gradient interpolation and M originally denotes the number of gradient elements in each group. In this way, we can make comparisons under the same degree of interpolation function, which eliminates the influence of symbols on evaluation results. We have run the process of encryption and decryption 10 times to get the average.



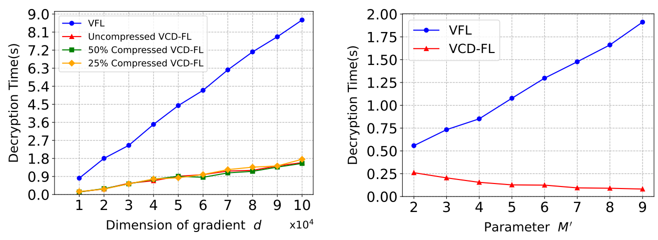
(a) (b)

Fig. 5. Encryption overhead of a client. (a) Encryption overhead with different d of a gradient. (b) Encryption overhead with different M'.

*1) **Encryption Overhead**:* As we discussed above, the encryption overhead of our VCD-FL and the VFL [19] is mainly determined by Lagrange interpolation computation. Recall that for a gradient with d dimensions, the VFL splits each element in a gradient into M' parts and computes the Lagrange interpolation function of degree M' with $M'+1$ points, while our VCD-FL divides d elements in a gradient into $\lceil \frac{d}{M'} \rceil$ groups and each group determines the Lagrange interpolation function of degree M' with $M'+1$ points. Given an element in a gradient, our VCD-FL only needs to compute a polynomial, whereas $M'+1$ polynomials need to be computed in the VFL. Therefore, the total encryption overhead of our VCD-FL is theoretically $(d+\lceil \frac{d}{M'} \rceil)M'$, while $(M'+1)dM'$ in the VFL. As a result, we can conclude that our VCD-FL has a total encryption overhead of approximately $\frac{(d+\lceil \frac{d}{M'} \rceil)M'}{(M'+1)dM'} \approx \frac{1}{M'}$ of that of the VFL.

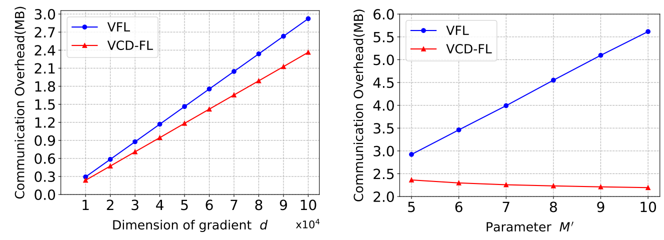
To further support the conclusion in practice, we evaluate the encryption overhead as the growth in dimension d of a gradient. The results are shown in Fig. 5(a). It is significant that the encryption time increases as d grows. When $M'=4$ and $d=100,000$, the encryption overhead of our VCD-FL is about 20.439s while that of the VFL is about 96.914s. Here, the encryption overhead of our VCD-FL is about a fifth of that of the VFL, which is slightly lower than the theoretical value analyzed above. That is because the VFL also includes an encoding using the Chinese Remainder Theorem (CRT) during the training process, which slightly increases the computation overhead.

By introducing gradient compression algorithm [24] described in Algorithm 3, our VCD-FL only needs to upload $p\%$ elements in a gradient to the AS. Therefore, the computation overhead of our VCD-FL theoretically reduces by $p\%$ at most. As depicted by Fig. 5(a), the encryption time decreases with the compression ratio increases under the same d. By using compression rate with 50% and 25% respectively, the encryption time under $d=100,000$ is further reduced from 20.439s to 13.807s and 8.992s, accordingly.



(a) (b)

Fig. 6. Decryption overhead of a client. (a) Decryption overhead with different dimensions d of a gradient. (b) Decryption overhead with different parameters M'.



(a) (b)

Fig. 7. Communication overhead between a client and the AS. (a) Communication overhead with different dimensions d of a gradient. (b) Communication overhead with different parameters M'.

TABLE III: Communication overhead comparison between our VCD-FL and the VFL [19]

Model	Number of Parameters	Communication Overhead	
		VFL	VCD-FL
MLP	$101,770$	2.975 MB	2.405 MB
CNN	$21,780$	0.637 MB	0.516 MB

Besides, given $d = 10,000$, we investigate the relationship between the parameter M' and the total encryption overhead. Fig. 5(b) shows that as M' grows, the encryption time of the VFL almost increases quadratically, while work done only rises linearly in our VCD-FL. When $M' = 9$, the encryption overhead of our VCD-FL is about 4.256s while that of the VFL is about 36.972s. The reason is that our VCD-FL adopts gradient grouping rather than gradient splitting, which will make the impact on interpolation polynomial computation be limited. As for an element in the gradient, each time M' increases by one, one more multiplication calculation for the corresponding interpolation polynomial computation. Therefore, the growth rate of computation overhead of our VCD-FL is nearly 1 but that of the VFL is about $2M' + 1$. It is observed that the total encryption overhead approximately increases linearly in our VCD-FL but quadratically in the VFL.

*2) **Decryption Overhead:*** The decryption overhead is the total of the overhead due to the aggregated result computation and commitment verification, as shown in Fig. 6(a). It can be seen that the decryption time almost linearly increases as d grows. When $M' = 4$ and $d = 100,000$, the decryption overhead of our VCD-FL without any compression is about 1.633s while that of the VFL [19] is about 8.704s. There are two reasons for this. On the one hand, because the aggregated result $\mathbf{g}$ is obtained by taking as input $\mathbb{Z}$, and removing $\mathbb{A}_i$ and the padding with 0, the computation overhead increases linearly as d grows. On the other hand, according to equation (18) and equation (19), the overhead of commitment verification is almost linear with d. In addition, because the decryption operation entirely depends on the input of $\mathbb{Z}$, the impact of Algorithm 3 using gradient compression algorithm [24] on the decryption time will be limited. The experimental results depicted by Fig. 6(a) further supports our theoretical analysis.

Likewise, given a fixed $d = 10,000$, we further investigate the impact of M' on decryption overhead. As depicted in Fig. 6(b), the decryption time of the VFL [19] almost linearly increases while that of our VCD-FL decreases with M' grows. The reason is that the decryption overhead is largely decided by the aggregated result computation. It depends on the number of interpolation points, which is about $(M' + 1)d$ in the VFL and $d + \lceil \frac{d}{M'} \rceil$ in our VCD-FL. As for commitment verification, the overhead varies for the same reason. It is significant that the decryption overhead of the VFL linearly increases while that of our VCD-FL decreases as M' grows.

D. Communication Overhead

To conveniently compare our VCD-FL with the VFL [19], we only evaluate the communication overhead between a client and the AS. Here, we measure the communication overhead by the size of uploaded information. The comparative experimental results under the MLP and the CNN between our VCD-FL and the VFL [19] are presented in Table III. As we discussed, the communication overhead mainly depends on d and M'. When $M' = 5$, which means both generate interpolation polynomials with degree 5, the communication overhead of the VFL under the MLP with $d = 101,770$ is about 2.975 MB while that of our VCD-FL is about 2.405 MB. By using the CNN with $d = 21,780$, the communication overhead of the VFL is about 0.637 MB while that of our VCD-FL is about 0.516 MB. The reason is that recall the gradient encryption process in Algorithm 2, given a d-dimensional gradient, our VCD-FL needs to upload $d + \lceil \frac{d}{M'} \rceil$ numbers to the AS, while d numbers uploaded in the VFL. However, because the VFL adopts the CRT to encode each number into a large integer, the size of each number in our VCD-FL is much less than that in the VFL. On the whole, the communication overhead of our VCD-FL is less than that of the VFL.

To further support the conclusion, given a fixed $M' = 5$, we investigate the communication overhead under different dimensions d of a gradient, and the result is depicted in Fig. 7(a). It can be seen that the communication overhead linearly increases as d grows. According to the above analysis, in our VCD-FL, the size of each uploaded number takes 32 bit and the communication overhead is about $(d + \lceil \frac{d}{M'} \rceil) \times 32$ bit. In the VFL [19], it converts each uploaded number in a gradient into a finite domain by multiplying with a scale factor and truncating the remaining fractional part [28]. Note that the ciphertext size of the VFL is determined by the size of each gradient, the size of the finite domain, and M', whose maximum is about $(M'+1)\times 32$ bit in theory. Because $(d+\lceil\frac{d}{M'}\rceil) < (M'+1)d$ for $M' > 1$, the communication overhead of our VCD-FL is less than that of the VFL. In fact, the size of generated ciphertexts in the experiment is smaller than the maximum. Therefore, the reduction rate of the communication overhead is not so much as the maximum.

Besides, given $d = 10,000$, we also conduct a comparative experiment of the communication overhead between our VCD-FL and the VFL [19] under different parameters M'. The result is shown in Fig. 7(b). It can be seen that the communication overhead of the VFL increases while that of our VCD-FL decreases as M' grows. That is because as M' grows, the magnitude of the algebraic structure ring increases, which takes more bits. However, it does not exist in our VCD-FL and the increase of M' decreases the number of groups, which results in the decrease of uploaded numbers and reduces the communication overhead of our VCD-FL.

VII. Related Work

In this section, we briefly review the state-of-the-art research on verifiable FL. Generally, the AS may manipulate the aggregated result unintentionally or intentionally, misleading the training models. How to validate the correctness of the aggregated result returned from the AS among those joint clients for model training that do not fully trust each other is crucial to the success of FL. Regarding this issue, most existing works that focus on verifiable FL aim to solve the problems such as privacy [18], [19], [29], performance [20], and auditability [21], [30].

One type is the centralized verifiable FL. The original verifiable FL is proposed in [18], which guarantees the verifiability using the generated cryptographic proof by the AS and the privacy by the proposed double-masking protocol. Fu *et al.* [19] proposed to use Lagrange interpolation for verifiability, and blinding technology for collusion-resistant privacy preservation. Zhang *et al.* [29] used a bilinear aggregate signature to verify the correctness of the aggregated result from the AS, and combined the CRT and the Paillier homomorphic encryption to protect privacy. To guarantee the verifiability of FL while improving the performance, Guo *et al.* [20] optimized the secure aggregation protocol in [22] by the proposed gradient hash commitment and amortized verification mechanism.

The other type is the distributed verifiable FL. Considering that the centralized AS might cause issues such as single-point failure, model trustability, and privacy leakage, blockchain as an underlying trust-building machine has been introduced into verifiable FL [21], [30]. Especially, Weng *et al.* [21] proposed an incentive mechanism based on blockchains to achieve verifiable FL. Peng *et al.* [30] proposed to select an effective committee based on blockchains for collective model aggregation and verifiability. However, the potential drawbacks of blockchains such as efficiency and scalability make these schemes impractical.

To the best of our knowledge, all these works are vulnerable to collusion attack verification. The corrupt clients might assist with the AS to make the falsified aggregated results pass verification. In addition, the computation and communication overheads caused by some operations with high complexity are still very expensive.

VIII. Conclusions

In this paper, we have proposed VCD-FL, which is verifiable, collusion-resistant, and dynamic federated learning. To guarantee verifiability while protecting gradient privacy during the training, we have proposed an efficient verification mechanism combined with a novel blinding protocol and a lightweight commitment scheme. To the best of our knowledge, our VCD-FL is the first work that can not only resist collusion-resistant verification but also support differentiated threat models using our proposed malicious behavior detection rules. Compared with existing works, our VCD-FL can reduce computation and communication overheads, while achieving stable and high-accuracy model training, and tolerating a certain number of clients dropping out.

References

[1] EU, "General data protection regulation (gdpr)," https://gdpr-info.eu/.

[2] the State of California Department of Justice, "California consumer privacy act (ccpa)," https://www.oag.ca.gov/privacy/ccpa.

[3] the Peoples Republic of China, "Chinas personal information protection law-an overview," https://usercentrics.com/knowledge-hub/china-personal-information-protection-law/.

[4] B. McMahan, E. Moore, D. Ramage *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of the 20th International Conference on Artificial Intelligence and Statistics*, Florida, USA, 2017, pp. 1273–1282.

[5] Q. Yang, Y. Liu, T. Chen *et al.*, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.

[6] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. of the 33rd Conference on Neural Information Processing Systems*, Vancouver, Canada, 2019, pp. 14 747–14 756.

[7] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning," in *Proc. of the 40th IEEE Symposium on Security & Privacy*, San Francisco, USA, 2019, pp. 739–753.

[8] C. Ma, J. Li, M. Ding *et al.*, "On safeguarding privacy and security in the framework of federated learning," *IEEE Network*, vol. 34, no. 4, pp. 242–248, 2020.

[9] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proc. of the 24th ACM SIGSAC Conference on Computer and Communications Security*, Dallas, USA, 2017, pp. 603–618.

[10] Z. Wang, M. Song, Z. Zhang *et al.*, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. of the 38th IEEE Conference on Computer Communications*, Paris, France, 2019, pp. 2512–2520.

[11] R. Shokri, M. Stronati, C. Song *et al.*, "Membership inference attacks against machine learning models," in *Proc. of the 38th IEEE Symposium on Security & Privacy*, San Jose, USA, 2017, pp. 3–18.

[12] L. Melis, C. Song, E. De Cristofaro *et al.*, "Exploiting unintended feature leakage in collaborative learning," in *Proc. of the 40th IEEE Symposium on Security & Privacy*, San Francisco, USA, 2019, pp. 691–706.

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

[14] C. Zhang, S. Li, J. Xia *et al.*, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. of the 2020 USENIX Annual Technical Conference*, Boston, USA, 2020, pp. 493–506.

[15] L. Sun and L. Lyu, "Federated model distillation with noise-free differential privacy," in *Proc. of the 30th International Joint Conference on Artificial Intelligence*, Montreal, Canada, 2020, pp. 1563–1570.

[16] L. Sun, J. Qian, and X. Chen, "Ldp-fl: Practical private aggregation in federated learning with local differential privacy," in *Proc. of the 30th International Joint Conference on Artificial Intelligence*, Montreal, Canada, 2021, pp. 1571–1578.
[17] K. Wei, J. Li, M. Ding *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.
[18] G. Xu, H. Li, S. Liu *et al.*, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2020.
[19] A. Fu, X. Zhang, N. Xiong *et al.*, "Vfl: A verifiable federated learning with privacy-preserving for big data in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3316–3326, 2022.
[20] X. Guo, Z. Liu, J. Li *et al.*, "Verifl: Communication-efficient and fast verifiable aggregation for federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1736–1751, 2021.
[21] J. Weng, J. Weng, J. Zhang *et al.*, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2438–2455, 2021.
[22] K. Bonawitz, V. Ivanov, B. Kreuter *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.
[23] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[24] Y. Lin, S. Han, H. Mao *et al.*, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. of the 6th International Conference on Learning Representations*, Vancouver, Canada, 2018, pp. 1–14.
[25] W. Y. B. Lim, N. C. Luong, D. T. Hoang *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.
[26] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. of 19th International Conference on Computational Statistics*, Paris, France, 2010, pp. 177–186.
[27] Y. LeCun, C. Cortes, and C. J.C. Burges, "The mnist database of handwritten digits," http://yann.lecun.com/exdb/mnist/.
[28] M. Hao, H. Li, X. Luo *et al.*, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2020.
[29] X. Zhang, A. Fu, H. Wang *et al.*, "A privacy-preserving and verifiable federated learning scheme," in *Proc. of IEEE International Conference on Communications*, Dublin, Ireland, 2020, pp. 1–6.
[30] Z. Peng, J. Xu, X. Chu *et al.*, "Vfchain: Enabling verifiable and auditable federated learning via blockchain systems," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 173–186, 2022.

Sheng Gao received the Ph.D. degree in computer science and technology from Xidian University, Xi'an, China, in 2014. He is now a Professor and a Ph.D. Supervisor with the School of Information, Central University of Finance and Economics. He has authored or co-authored over 5 books and published over 50 papers in refereed international journals and conferences, such as IEEE Transactions on Information Forensics and Security, IEEE Transactions on Vehicular Technology, IEEE Transactions on Services Computing, and IEEE Transactions on Wireless Communications. His current research interests include blockchain, data security, and privacy computing.

Jingjie Luo received the B.S. degree from Nanjing Normal University, Nanjing, China in 2020. Currently, he is pursuing the M.S. degree in the School of Information, Central University of Finance and Economics. His current research focuses on federated learning and privacy protection.

Jianming Zhu received the Ph.D. degree in computer application technology from Xidian University, Xi'an, China, in 2004. He is working as a Professor with the School of Information, Central University of Finance and Economics. From September 2008 to March 2009, he was a Research Fellow with the University of Texas at Dallas, USA. His research interests include wireless network security, data privacy, and blockchain.

Xuewen Dong received the B.E., M.S., and Ph.D. degrees in computer science and technology from Xidian University, Xi'an, China, in 2003, 2006, and 2011, respectively. From 2016 to 2017, he was with Oklahoma State University, OK, USA, as a Visiting Scholar. Now, he is a professor in the School of Computer Science at Xidian University. His research interests include blockchain, wireless network security, and AI security.

Weisong Shi (Fellow, IEEE) received the B.S. degree in computer engineering from Xidian University, Xian, China, in 1995, and the Ph.D. degree in computer engineering from the Chinese Academy of Sciences in 2000. He is currently a Professor and Chair of the Department of Computer and Information Sciences at the University of Delaware (UD). Before joining UD, he was a professor of Computer Science at Wayne State University (2002-2022) and served in multiple administrative roles, including the Associate Dean for Research and Graduate Studies at the College of Engineering and Interim Chair of Computer Science. He has published more than 270 articles in peer-reviewed journals and conferences. Particularly his pioneering paper entitled "Edge Computing: Vision and Challenges" has been cited more than 5000 times. He is an IEEE Fellow and a Distinguished Scientist of ACM.