VFEFL: Privacy-Preserving Federated Learning against Malicious Clients via Verifiable Functional Encryption

Nina Cai[a], Jinguang Han[a,b,*] and Weizhi Meng[c]

[a]School of Cyber Science and Engineering, Southeast University, Nanjing, 210096, China
[b]State Key Laboratory of Internet Architecture, Tsinghua University, Beijing, 100084, China
[c]School of Computing and Communications, Lancaster University, Lancaster, United Kingdom

ARTICLE INFO

Keywords:
Federated learning
Functional enctyption
Data privacy
Verifiability
Robustness

ABSTRACT

Federated learning is a promising distributed learning paradigm that enables collaborative model training without exposing local client data, thereby protecting data privacy. However, it also brings new threats and challenges. The advancement of model inversion attacks has rendered the plaintext transmission of local models insecure, while the distributed nature of federated learning makes it particularly vulnerable to attacks raised by malicious clients. To protect data privacy and prevent malicious client attacks, this paper proposes a privacy-preserving Federated Learning framework based on Verifiable Functional Encryption (VFEFL), without a non-colluding dual-server assumption or additional trusted third-party. Specifically, we propose a novel Cross-Ciphertext Decentralized Verifiable Functional Encryption (CC-DVFE) scheme that enables the verification of specific relationships over multi-dimensional ciphertexts. This scheme is formally treated, in terms of definition, security model and security proof. Furthermore, based on the proposed CC-DVFE scheme, we design a privacy-preserving federated learning framework that incorporates a novel robust aggregation rule to detect malicious clients, enabling the effective training of high-accuracy models under adversarial settings. Finally, we provide the formal analysis and empirical evaluation of VFEFL. The results demonstrate that our approach achieves the desired privacy protection, robustness, verifiability and fidelity, while eliminating the reliance on non-colluding dual-server assumption or trusted third parties required by most existing methods.

1. Introduction

Federated learning [1] is an advanced and promising distributed learning paradigm that enables multiple clients to collaboratively train models without sharing local data. It embodies the principles of focused data collection and minimization, thereby mitigating many of the systemic privacy risks and costs inherent in traditional, centralized machine learning approaches. With the enactment of privacy protection and data security regulations in various countries, as well as the increase in public privacy awareness, the application of federated learning has gradually expanded e. g., healthcare, finance, and edge computing.

Although the federated learning framework offers a degree of privacy protection for client data during training, it brings new privacy and security issues [2, 3]. One primary concern is the potential privacy leakage through the local models uploaded by clients. With the advent and ongoing advancement of attack techniques, such as model inversion attacks [4, 5, 6], submitting local models in plaintext form is no longer regarded as a secure method. Accordingly, obfuscation and encryption mechanisms have been adopted to address this issue. Another threat is attacks from malicious clients. Due to its distributed architecture, federated learning is susceptible to adversarial manipulation by malicious clients. A malicious client may make the accuracy of the global model drop rapidly by uploading malicious models, or destroy the decryption by uploading a malicious decryption key or a wrong ciphertext. The above issues are interrelated and important for federated learning systems to achieve their core goal (training high quality models while preserving data privacy). Privacy-preserving mechanisms (such as encryption schemes) can potentially lead to difficulties in detection and verification, while detecting and verifying encrypted models of the clients may compromise specific information. Existing solutions [7, 8] cannot be deployed in the basic federated learning architecture (one server and multiple clients), and need help from two non-colluding servers or additional trusted third parties, which introduces additional limitations and consumption.

Therefore it is very important and challenging to design a federated learning scheme that preserves privacy and resists malicious client attacks in the basic federated learning architecture. This paper proposes a privacy-preserving Federated Learning scheme based on Verifiable Functional Encryption (VFEFL). Our contributions can be summarized as follows:

1. We employ a verifiable functional encryption scheme to encrypt local models in the federated learning, ensuring data privacy and correctness during encryption and decryption. The scheme supports verifiable evaluation of relations over multidimensional ciphertexts, making it well suited for federated learning scenarios. Furthermore, we formalize its definition and security model, and reduce its security in the proposed model to well-established computational hardness assumptions.

*Corresponding author. E-mail: jghan@seu.edu.cn
nncai@seu.edu.cn (N. Cai); jghan@seu.edu.cn (J. Han); weizhi.meng@ieee.org (W. Meng)
ORCID(s): 0009-0000-3132-5797 (N. Cai)

2. We propose a privacy-preserving federated learning scheme named VFEFL to protect local model privacy and defend against malicious clients. Compared with existing approaches, it enables verifiable client-side aggregated weights and can be integrated into standard federated learning architectures to enhance trust. Theoretical analysis demonstrates that VFEFL provides strong guarantees in both privacy and robustness.
3. We implement a prototype of VFEFL and conduct comprehensive experiments to evaluate its fidelity, robustness, and efficiency. Robustness is evaluated under both targeted and untargeted poisoning attacks. Experimental results demonstrate that VFEFL effectively defends against such attacks while preserving model privacy.

2. RELATED WORK

2.1. Functional Encryption

Functional encryption (FE) [9, 10] is a paradigm supports selective computation on encrypted data, while traditional public key encryption only implements all-or-nothing decryption method. An authority in a FE scheme can generate restricted decryption keys to allow receivers to learn specific functions of the encrypted messages and nothing else. Goldwasser [11] first introduced the definition of multi-input functional encryption (MIFE), which extends functional encryption to support multi-input functions. Abdalla [12] constructed the first MIFE for a non-trivial functionality with polynomial security loss under falsifiable assumptions for a super-constant number of slots.

Although the input can be large, the original definition of FE limits the source of the encrypted data to a single client, which is not compatible with many real-world scenarios where data must be collected from multiple sources. Multi-client functional encryption (MCFE) [13] was proposed to addresses this issue. Further, decentralized multi-client functional encryption (DMCFE) [13] eliminated reliance on a trusted party. In the DMCFE scheme instantiated over an asymmetric pairing group, clients can locally generate their own keys without relying on a trusted third party.

The decentralized architecture of DMCFE introduces additional challenges arising from potentially dishonest clients, which motivates the use of verifiable functional encryption (VFE) [14] as a solution. Nguyen et al. [15] proposed a verifiable DMCFE scheme for inner products, along with its formal definitions and security proofs.

2.2. Federated Learning

Federated Learning against Malicious Clients. Robust aggregation rules are one of the most common means of resisting malicious clients in federated learning [3]. Next, we review some of the existing Byzantine robust aggregation rules (ARG).

- **FedAvg [1].** FedAvg directly computes the weighted sum of all local model to obtain the global model, where the weight of each client is determined by the proportion of its local dataset size to the total dataset size. However, FedAvg cannot resist malicious client attacks, in which an adversary can manipulate the global model by submitting crafted local models.

- **Krum [16].** This AGR selects global model update based on the squared distance score. The global model is selected as either the one with the lowest score or the average of several lowest-scoring models.

- **FLTrust [17].** The method requires the server to maintain a small clean dataset, which is used to train a trusted model update g_0. A trust score for each local model update g_i is then computed by evaluating the cosine similarity between g_i and g_0, followed by applying a ReLU operation. The score is used as the aggregation weight of the local model in the aggregation process.

- **AGR in FheFL [8].** In FheFL [8], the aggregation methods use the Euclidean distance between the local model W_i^t and the global model W^{t-1} of the previous round as a discriminator of the quality of the local model W_i^t.

- **AGR in BSR-FL [18].** This aggregation method constructs a qualified aggregation model using secure cosine similarity and an incentive-based Byzantine robustness strategy. In BSR-FL, the global model from the $(t-1)$-th iteration, denoted by W^{t-1}, serves as the baseline model. For each local update W_i^t, the trust score cs_i^t is computed as the cosine similarity between W_i^t and W^{t-1}, followed by a transformation using a parameterized sigmoid function centered at 0.5. However, BSR-FL does not verify the magnitude of local model updates, which leaves the system vulnerable to spoofing attacks.

Existing aggregation rules are either not robust enough to resist adaptive attacks [17], or too complex to be applicable for fast verification of zero-knowledge proofs. Therefore, we propose a new aggregation rule that is applicable to zero-knowledge proofs and provides an effective defense against attacks such as adaptive attacks.

Privacy-Preserving FL against Malicious Clients. In order to make Privacy-Preserving Federated Learning (PPFL) resistant to the behavior of malicious participants, some works have been proposed to implement the defense strategies in PPFL. The relevant schemes can be categorized into 3 groups according to privacy techniques. A comparison of these previous schemes with our method is summarized in Table 1.

- **Differential Privacy Based FL.** The Biscotti[19] combined differential privacy, Shamir secret shares (SS) and Multi-Krum aggregation rule to realize a private and secure federated learning scheme based on the blockchain. SecProbe[20] aggregated local

Table 1
Comparison with other schemes

Proposed Works	Techniques	Robustness	Self-Contained	Verifiability	Fidelity
Biscotti[19]	DP+SS	✓	✗	✓	✗
SecProbe[20]	DP	✓	✓	✗	✗
ESFL[21]	DP	✓	✓	✗	✗
SecureFL[22]	PLHE+SS	✓	✗	✓	✓
AegisFL[7]	HE	✓	✗	–	✓
RVPFL[23]	HE	✓	✗	✓	✓
FheFL[8]	FHE	✓	✓	✗	✓
FEFL[24]	2DMCFE	✗	✓	✗	✓
BSR-FL[18]	NIFE	✓	✗	✗	✓
Ours	CC-DVFE	✓	✓	✓	✓

Note: **Self-contained** means the scheme works in a single-server setup without trusted third parties. **Verifiability** ensures the client follows protocol specifications. **Fidelity [17]** means no accuracy loss when no attack exists.

models based on the computed utility scores and applies a functional mechanism to perturb the objective function to achieve privacy. Miao et al.[21] proposed an efficient and secure federated learning (ESFL) framework based on DP. Since the nature of differential privacy techniques is to hide the sensitive information of an individual by introducing random noise in the data or computational results, it cannot avoid affecting the accuracy of the model.

- **Homomorphic Encryption Based FL.** SecureFL [22] implemented the FLTrust construction using packed linear homomorphic encryption, which ensures stronger security. However, in this scheme, the client distributes the local gradient to two servers through secret sharing, and the privacy protection of the gradient must rely on two non-colluding servers. AegisFL [7] improved the computational efficiency of multiple aggregation rules under ciphertexts by optimizing the homomorphic encryption, but it also fails to eliminate dependency on two non-colluding servers. The schemes [25, 26, 23] enhance privacy in federated learning using homomorphic encryption and related techniques. However, they all rely on trusted third entities or additional participants, limiting their applicability. FheFL [8] built the scheme on homomorphic encryption in a single server. But a trusted key distribution process is necessary to ensure the secure generation of pairwise keys among users. The scheme does not validate the client behavior, allowing malicious clients to disrupt decryption through incorrect keys. Such schemes often employ homomorphic encryption to ensure privacy, allowing aggregation to be performed directly on ciphertexts. Decryption typically requires a trusted party or two non-colluding servers. Shared decryption keys risk exposing honest clients' data to key holders through collusion or eavesdropping.

- **Functional Encryption Based FL.** Chang et al.[24] constructed a privacy-preserving federated learning scheme FEFL via functional encryption. Qian et al. [27] designed another privacy-preserving federated learning scheme by combining functional encryption with multiple cryptographic primitives. However, both schemes do not address the detection of malicious clients. The BSR-FL scheme [18] combined functional encryption and blockchain technology to construct an efficient Byzantine robust privacy-preserving federated learning framework. Nevertheless, the scheme still requires two non-colluding entities (the task publisher and the server) and does not validate the magnitude of the submitted local models.

3. PROBLEM STATEMENT

3.1. System Model

VFEFL is applicable to a wide range of privacy-preserving federated learning settings and does not require any trusted third party. Its architecture involves two main roles: a server $\mathcal{S}$ and n clients $(C_i)_{i\in[n]}$.

Server: The server, equipped with a clean dataset D_0 and substantial computational resources, is responsible for detecting maliciously encrypted local models submitted by clients, aggregating the models from clients, and returning the aggregated model to the clients for further training. This process is iteratively performed until a satisfactory global model is achieved.

Clients: Clients have their own datasets and a certain amount of computational power, and it is expected that multiple clients will cooperate to train the model together without revealing their own data and model information.

The notations in the paper are described in Table 2

3.2. Design Goals

VFEFL is designed to provide the following performance and security guarantees:

- **Privacy.** VFEFL should ensure that the server or channel eavesdroppers do not have access to the client's local model to reconstruct the client's private data.

Table 2
Notations and Descriptions

Notations	Descriptions
n	the number of clients
m	the dimension of model
t	training epoch
lr	local learning rate
R_l	the number of local iterations
ℓ	the label in the DVFE
W^t	global model
W_0	initial global model
W_i^t	local model
D_0	root dataset
D_i	local dataset
g, w_i, u, v	the generators of $\mathbb{G}_1$
$h, \hat{v}_1, \hat{v}_2$	the generators of $\mathbb{G}_2$

Note: The superscripts of $W_i^t, \vec{x}_i^t, D_i$ denote iterations and the subscripts denote client identifiers.

- **Byzantine Robustness.** VFEFL can resist Byzantine attacks such as Gaussian attack [28], the adaptive attack [17] and Label Flipping attack, and obtain a high-quality global model.
- **Self-contained.** VFEFL can be deployed in the basic federated learning framework, i.e., without relying on two non-colluding servers and additional third parties, thus gaining more general applicability.
- **Verifiability.** VFEFL ensures clients' outputs comply with the protocol, including ciphertexts and functional key shares, preventing malicious clients from framing honest ones or disrupting aggregation. It also ensures system continuity despite partial data corruption during transmission.
- **Fidelity.** VFEFL architecture causes no loss of accuracy in the absence of attacks.

We will proof the above properties in Section 7.

3.3. Security Model

The protocol π securely realizes a function f if, for any adversary $\mathcal{A}$, there exists no environment $\mathcal{E}$ can distinguish whether it is interacting with π and $\mathcal{A}$ or with the ideal process for f and ideal adversary $\mathcal{A}^*$ with non-negligible probability. Formally, let $\text{REAL}_{\mathcal{A}}^{\pi}(\lambda, \hat{x})$ and $\text{IDEAL}_{\mathcal{A}^*}^{f}(\lambda, \hat{x})$ denote the interaction views of $\mathcal{E}$ in the real-world and ideal-world models, respectively. For any $\mathcal{E}$, if the following holds:

$$\text{REAL}_{\mathcal{A}}^{\pi}(\lambda, \hat{x}) \overset{c}{\approx} \text{IDEAL}_{\mathcal{A}^*}^{f}(\lambda, \hat{x}),$$

where $\overset{c}{\approx}$ denotes computational indistinguishability, λ represents the security parameter, and $\hat{x}$ includes all inputs.

In the design goal of VFEFL, privacy is defined that neither the server nor eavesdroppers can obtain the local models of clients. In the execution of VFEFL protocols, an attacker may compromise either some clients or the server, thereby gaining access to encrypted data and intermediate results. With them, the attacker seeks to extract private information. Thus, we define the ideal function of VFEFL as f_{VFEFL}. The privacy of VFEFL can be formally described as:

$$\text{REAL}_{\mathcal{A}}^{\text{VFEFL}}(\lambda, \hat{x}_{\mathcal{S}}, \hat{x}_{\mathcal{C}}, \hat{y}) \overset{c}{\approx} \text{IDEAL}_{\mathcal{A}^*}^{f_{\text{VFEFL}}}(\lambda, \hat{x}_{\mathcal{S}}, \hat{x}_{\mathcal{C}}, \hat{y}),$$

where $\hat{y}$ denotes intermediate results, $\hat{x}_{\mathcal{S}}, \hat{x}_{\mathcal{C}}$ denote the inputs of $\mathcal{S}$ and $\mathcal{C}$, respectively. The formal proof as discussed in Section 7.1.

3.4. Threat Model

We consider the honest-but-curious server, which executes the protocol faithfully but may leverage techniques such as gradient inversion attacks on the client's model to infer private data. We categorize the clients $\mathcal{C}$ into two groups: honest clients $\mathcal{HC}$ and malicious clients $\mathcal{CC}$. An honest client honestly trains a local model based on its real dataset in each training iteration, encrypts the model, generates functional key share and the proofs of them. A malicious client may submit random or carefully crafted model parameters to degrade the accuracy of the global model, or submit incorrect encrypted models or functional key shares to cause decryption failure. Additionally, a malicious client may eavesdrop on an honest client's secrets by colluding with others.

Our system requires at least two clients to be honest to ensure security, even in the presence of collusion.

4. PRELIMINARIES

4.1. Groups

Prime Order Group. We use a prime-order group generator $\texttt{GGen}$, a probabilistic algorithm which inputs a security parameter λ, and outputs a description $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is an additive cyclic group of order p, p is a 2λ-bit prime, and g is the generator of $\mathbb{G}$. For any $a \in \mathbb{Z}_p$, given g^a, it is computationally hard to determine a, a problem known as the discrete logarithm problem. For any $\vec{a} = \{a_1, \cdots, a_n\} \in \mathbb{Z}_p^n$, $\vec{g} = \{g_1, \cdots, g_n\} \in \mathbb{G}^n$, we denote $\vec{g}^{\vec{a}} = \prod_{i\in[n]} g_i^{a_i}$.

Pairing group. We use a pairing group generator $\texttt{PGGen}$, which inputs a security parameter λ and returns a description $\mathcal{PG} = \left(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, h, e\right)$ of asymmetric pairing groups where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are additive cyclic groups of order p for a 2λ-bit prime p, g and h are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable (non-degenerate) bilinear map. For $a, b \in \mathbb{Z}_p$, given g^a, h^b, one can efficiently compute $e(g, h)^{ab}$ using the pairing e.

Class Group. We consider a Decisional Diffie-Hellman (DDH) group that contains a subgroup where the discrete logarithm (DL) problem is easy[29], which can be instantiated from class groups of imaginary quadratic fields. Let GenClassGroup be a pair of algorithms $(\texttt{Gen}, \texttt{Solve})$. The $\texttt{Gen}$[30] algorithm is a group generator which takes as inputs a security parameter λ and a prime q and outputs the parameters $\mathcal{DG} = \left(\tilde{s}, g, f, g_q, \hat{G}, G, F, \hat{G}^q\right)$. The set $(\hat{G}, \cdot)$ is a finite abelian group of order $q\hat{s}$ where the bitsize of $\hat{s}$ is a function of λ, with $gcd(q, \hat{s}) = 1$. $\tilde{s}$ is an upper bound

of $\hat{s}$. It is also required that one can efficiently recognise valid encodings of elements in $\hat{G}$. The set $(F, \cdot)$ is the unique cyclic subgroup of $\hat{G}$ of order q, generated by f. The set $(G, \cdot)$ is a cyclic subgroup of $\hat{G}$ of order qs where s divides $\hat{s}$. $G^q = \{x^q, x \in G\}$ is the subgroup of order s of G, it holds that $G = G^q \times F$. Moreover, the DL problem is easy in F, with an efficient algorithm `Solve` [29]. According to the Hard subgroup membership (HSM) assumption [30], it is hard to distinguish random elements of G from those of G^q.

4.2. Zero Knowledge Proof

A zero-knowledge proof for L is a protocol between prover $\mathcal{P}$ and verifier $\mathcal{V}$ where $\mathcal{P}$ convinces $\mathcal{V}$ that a common input $\varphi(w)$ without revealing information about a witness w. A zero-knowledge proof should satisfy three properties: soundness, completeness and perfect honest-verifier zero-knowledge. We denote by

$$\mathsf{PoK}\{(w) : \varphi(w)\}$$

a proof of knowledge of a secret value w such that the public predicate $\varphi(w)$ holds.

Notably, an interactive proof-of-knowledge protocol can be transformed into a non-interactive proof-of-knowledge through the Fiat-Shamir heuristic [31].

4.3. Cross-Ciphertext Decentralized Verifiable Functional Encryption

Verifiable functional encryption (VFE) [14] was introduced to capture ciphertexts and decryption keys' correctness under fine-grained control. Informally, VFE requires that for any ciphertext C that passes a public verification procedure, there exists a unique message m such that, for any valid function description f and any corresponding function key dk_f that also pass public verification, the decryption algorithm outputs $f(m)$ when given C, dk_f, and f as inputs.

Nguyen et al.[15] proposed a decentralized multi-client verifiable functional encryption scheme. The scheme focuses on verifying properties of individual ciphertexts. In our work, we propose a verifiable functional encryption scheme that enables cross-ciphertext verification, allowing the validation of specific relations among multiple ciphertexts in various application scenarios. Next, we give the formal definition of our Cross-Ciphertext Decentralized Verifiable multi-client Functional Encryption for inner product (CC-DVFE).

Definition 1. *A Cross-Ciphertext Decentralized Verifiable Multi-Client Functional Encryption for Inner Product among a set of n clients $(C_i)_{i\in[n]}$ with a m-dimensional vector on $\mathcal{M}^m$ contains eight algorithms.*

- $\mathtt{SetUp}(\lambda) \rightarrow pp$*: Takes as input the security parameter λ and outputs the public parameters pp.*
- $\mathtt{KeyGen}(pp) \rightarrow \left((ek_i, sk_i)_{i\in[n]}, vk_{CT}, vk_{DK}, pk\right)$*: Each client C_i generates own encryption key ek_i and secret key sk_i, and eventually outputs the verifiation key for ciphertexts vk_{CT} and for functional key share vk_{Dk}, and the public key pk.*
- $\mathtt{Encrypt}\left(ek_i, \vec{x}_i, aux, \ell, (\ell_{Enc,j})_{j\in[m]}, pp\right) \rightarrow (C_{\ell,i}, \pi_{CT,i})$ *: Takes as input an encryption key ek_i, a vector $\vec{x}_i$ to encrypt, the auxiliary information aux used in the ciphertext proof protocol, a label ℓ and a set encryption labels $\left(\ell_{Enc,j}\right)_{j\in[m]}$ that ensures the ciphertexts of different dimensions in each round cannot be illegally aggregated. Outputs the ciphertext $C_{\ell,i}$ and a zero knowledge proof $\pi_{CT,i}$.*
- $\mathtt{VerifyCT}\left((C_{\ell,i})_{i\in[n]}, \pi_{CT}, vk_{CT}\right) \rightarrow 1$ ***or*** 0*: Takes as input the ciphertexts $(C_{\ell,i})_{i\in[n]}$ and proofs π_{CT} from n client and the verification keys vk_{CT}. It outputs 1 for accepting or 0 with a set of malicious clients $\mathcal{CC}_{CT} \neq \emptyset$ for rejecting.*
- $\mathtt{DKeyGenShare}\left(sk_i, pk, y_i, \ell_y, pp\right) \rightarrow \left(dk_{y,i}, \pi_{DK,i}\right)$*: Takes as input the secret key sk_i of a client, public key pk, a value y_i and a function label ℓ_y. Outputs a functional decryption key share $dk_{y,i}$ and a zero knowledge proof $\pi_{DK,i}$.*
- $\mathtt{VerifyDK}\left((dk_{y,i})_{i\in[n]}, \pi_{DK}, vk_{DK}, pp\right) \rightarrow 1$ ***or*** 0*: Takes as input functional decryption key shares $(dk_{y,i})_{i\in[n]}$ and proofs π_{DK} from n clients and a verification key vk_{DK}. It outputs 1 for accepting or 0 with a set of malicious clients $\mathcal{CC}_{DK} \neq \emptyset$ for rejecting.*
- $\mathtt{DKeyComb}\left((dk_{y,i})_{i\in[n]}, \ell_y, pk\right) \rightarrow dk$*: Takes as input all of the functional decryption key shares $(dk_{y,i})_{i\in[n]}$, a function label ℓ_y and public key pk. Outputs the functional decryption key dk.*
- $\mathtt{Decrypt}\left(\left(C_{\ell,i}\right)_{i\in[n]}, dk, \vec{y}\right) \rightarrow (\langle \vec{X}_j, \vec{y}\rangle)_{j\in[m]}$ ***or*** $\perp$*: Takes as input the m-dimensions vector ciphertext $\left(C_{\ell,i}\right)_{i\in[n]}$ from n client , a functional decryption key dk and the function vector $\vec{y}$. Outputs $(\langle \vec{X}_j, \vec{y}\rangle)_{j\in[m]}$ or $\perp$, where $\vec{X}_j = (x_{1,j}, x_{2,j}, \ldots, x_{n,j})$ denotes the vector formed by the j-th component from each of the n users.*

Correctness. Given any set of message $\left(\vec{x}_1, \cdots, \vec{x}_n\right) \in \mathcal{M}^{n\times m}$ and $\vec{x}_0 \in \mathcal{M}^m$, and any function vector $\vec{y} \in \mathbb{Z}_p^n$, We say that an CC-DVFE scheme is correct if and only if following equation holds.

$$\Pr\left[\begin{array}{l|l} & \mathtt{SetUp}(\lambda) \rightarrow \mathrm{pp} \\ & \mathtt{KeyGen}(pp) \rightarrow ((ek_i, sk_i)_{i\in[n]}, \\ \mathtt{DKeyComb}(& \quad vk_{CT}, vk_{DK}, pk), \\ \quad (dk_{y,i})_{i\in[n]}, \ell_y, & \forall i \in [n],\ \mathtt{Encrypt}(ek_i, \vec{x}_i, \\ \quad pk) \rightarrow dk & \quad aux, \ell, (\ell_{Enc,j})_{j\in[m]}, pp) \\ \mathtt{Decrypt}((C_{\ell,i})_{i\in[n]}, & \quad \rightarrow (C_{\ell,i}, \pi_{CT,i}) \\ \quad dk, \vec{y}) \rightarrow & \forall i \in [n],\ \mathtt{DKeyGenShare}(\\ \quad \left(\langle \vec{X}_j, \vec{y}\rangle\right)_{j\in[m]} & \quad sk_i, pk, y_i, \ell_y, pp) \\ & \quad \rightarrow (dk_{y,i}, \pi_{\mathrm{DK},i}) \\ & \mathtt{VerifyCT}((C_{\ell,i})_{i\in[n]}, \pi_{CT}, \\ & \quad vk_{CT}) \rightarrow 1 \\ & \mathtt{VerifyDK}((dk_{y,i})_{i\in[n]}, \pi_{DK}, \\ & \quad vk_{DK}, pp) \rightarrow 1 \end{array}\right] = 1.$$

4.4. Our New Aggregation Rule

Inspired by the aggregation rule of FLTrust [17], we similarly introduce root dataset D_0 to bootstrap trust. However, FLTrust contains cosine similarity operations that are difficult to complete the verification using a simple zero-knowledge proof, which is avoided in our new aggregation rule.

A malicious client can arbitrarily manipulate the direction and magnitudes of the local model to degrade the accuracy of the global model, so our robust aggregation rule restricts both ways. First, we assume that the server has a small clean dataset D_0 and train on it to get a baseline model W_0^t. Without considering the magnitudes of the model, we use the inner product of the local client model W_i^t and the baseline model W_0^t to measure the directional similarity between them. In order to avoid malicious users to enhance the attack effect of malicious models by amplifying the model vector, we incorporate the inner product value of the local client model W_i^t itself into the robustness aggregation rule. We further consider that when the inner product value is negative, the model still has some negative impact on the global model of the aggregation, so we use the Relu function to further process the inner product value. The Relu function is defined as follows:

$$\mathrm{ReLu}\,(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{others} \end{cases}.$$

We define the robust aggregation rule as follows:

$$W^* = \sum_{i\in[n]} \mathrm{Relu}\left(\frac{\langle W_i^t, W_0^t\rangle}{\langle W_i^t, W_i^t\rangle}\right) W_i^t$$

At the same time, in order to avoid the global model unable to statute to a good level of accuracy, we normalize each epoch of global model to a certain magnitude. We use the W_0^t as a reference to normalize each epoch of global model. That is, the global gradient W^t of for the t-th iteration is defined as follows:

$$W^t = \frac{||W_0^t||}{||W^*||} W^*$$

Let $y_i = \mathrm{Relu}\left(\frac{\langle W_i^t, W_0^t\rangle}{\langle W_i^t, W_i^t\rangle}\right)$, and W^* can be obtained from the functional encryption for inner product. The robustness of this aggregation rule is proved in Section 7.

5. OUR DECENTRALIZED VERIABLE FUNCTIONAL ENCRYPTION

5.1. Construction of CC-DVFE

Although VFE schemes supporting weight aggregation for encrypting local models are now readily available, their application to privacy-preserving and robust federated learning faces the following two challenges. The first one is that the current ciphertext verification in VFE supports range proofs for individual elements but cannot verify Byzantine robustness rules. Secondly, the construction of zero-knowledge proofs for the Byzantine robust aggregation rule faces compatibility issues: existing inner-product arguments [32] and functional encryption schemes cannot be directly combined for verification. Specifically, the current inner-product argument fails to achieve the zero-knowledge property, and the ciphertext structure in functional encryption does not support direct verification of Byzantine robust aggregation rules.

To address these limitations, we propose a novel verifiable functional encryption scheme for the new robust aggregation rule. Next we describe the complete CC-DVFE scheme construction, with eight algorithms. The flowchart of the algorithm CC-DVFE is shown in Figure 1.

- $\mathtt{SetUp}(\lambda) \rightarrow pp$**:** Takes as input the security parameter λ and outputs the public parameters $pp = (\mathcal{PG}, \mathcal{DG}, h_p, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_1', \mathcal{H}, \ell_D)$, where $\mathtt{PGGen}(\lambda) \rightarrow \mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, h, e)$ is a pairing group, g the generator of $\mathbb{G}_1$, h the generator of $\mathbb{G}_2$, $\mathtt{GenClassGroup}$ $(\lambda, p) \rightarrow \mathcal{DG} = (\tilde{s}, f, \hat{h}_p, \hat{G}, F, \hat{G}^p)$ is a DDH group with an easy DL subgroup, $\hat{t} \leftarrow \mathcal{D}_p$ is sampled, and $h_p = \hat{h}_p^{\hat{t}}$ is set. Four full-domain hash functions are defined as follows:

  $$\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^2, \quad \mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{G}_2^2,$$
  $$\mathcal{H}_1' : \{0,1\}^* \rightarrow \mathbb{G}_1, \quad \mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p.$$

  And $\ell_D = (\{0,1\}^*)$ is the initialization label.

- $\mathtt{KeyGen}(pp) \rightarrow \left((ek_i, sk_i)_{i\in[n]}, vk_{CT}, vk_{DK}, pk\right)$**:** It is executed by each client C_i to generate own encryption key ek_i and secret key sk_i, and eventually outputs a verifiation key for ciphertexts vk_{CT} and functional key share vk_{Dk}, and a public key pk. A client selects $\hat{k}_i = (\hat{k}_{i,1}, \hat{k}_{i,2}) \xleftarrow{\$} \mathbb{Z}_p^2$, $t_i = (t_{i,1}, t_{i,2}) \xleftarrow{\$} \mathcal{D}_p^2$ and $s_i = (s_{i,1}, s_{i,2}) \xleftarrow{\$} \mathbb{Z}_p^2$, then computes $T_i = \left(h_p^{t_{i,b}}\right)_{b\in[2]}$ and

  $$d_i = \left(f^{\hat{k}_{i,b}} \cdot \left(\prod_{i<i^*}^{n} T_{i^*,b} \cdot \prod_{i>i^*}^{n} T_{i^*,b}^{-1}\right)^{t_{i,b}}\right)_{b\in[2]}$$

  and commits s_i as

  $$\mathrm{com}_i = (\upsilon)^{s_i}$$

  where $\upsilon = \mathcal{H}_1(\ell_D) \in \mathbb{G}_1^2$. For each client C_i, encryption key is $ek_i = s_i$, secret key is $sk_i = (s_i, \hat{k}_i, t_i)$, and public key is $pk = \left(T_i,\ d_i\right)_{i\in[n]}$. The verification key for ciphertexts is $vk_{CT} = \left(\mathrm{com}_i\right)_{i\in[n]}$, while for functional keys it is $vk_{DK} = \left(T_i, d_i, \mathrm{com}_i\right)_{i\in[n]}$.

- $\mathtt{Encrypt}(ek_i, \vec{x}_i, \vec{x}_0, \ell, (\ell_{Enc,j})_{j\in[m]}, pp) \rightarrow (C_{\ell,i}, \pi_{CT,i})$ **:** Takes as input an encryption key ek_i, a vector $\vec{x}_i$ to encrypt, a vector $\vec{x}_0$ used in the ciphertext

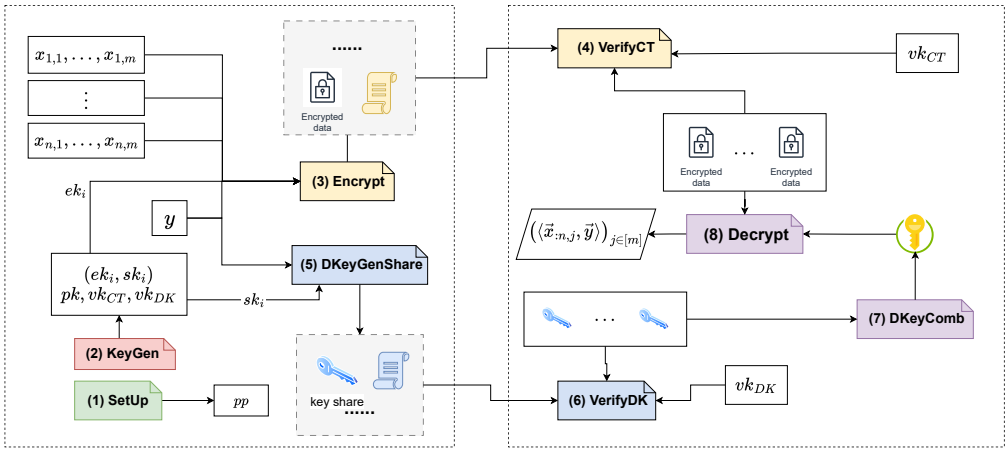


Figure 1: The workflow of our CC-DVFE

proof protocol, a label ℓ and a set encryption labels $(\ell_{Enc,j})_{j\in[m]}$ that ensures the ciphertexts of different dimensions in each round cannot be illegally aggregated. The ciphertext is $C_{\ell,i} = \left((u)^{s_i} \cdot (w_j)^{x_{i,j}}\right)_{j\in[m]}$ where $u = \mathcal{H}_1(l) \in \mathbb{G}_1^2$, $w_j = \mathcal{H}_1'(\ell_{Enc,j}) \in \mathbb{G}_1$. It re-computes com_i and $V = \prod_{j\in[m]} C_{\ell,i,j}$, then generates a proof $\pi_{CT,i}$ for the relation $\mathcal{R}_{Encrypt}$ on input $\left(V, \text{com}_i, \ell, \ell_{Enc}, y_i, \vec{x}_0\right)$. The $\pi_{CT,i}$ is described as

$$\text{PoK}\left\{(s_i, \vec{x}_i): \begin{array}{r} V_i = (u^m)^{s_i} \cdot \vec{w}^{\vec{x}_i} \wedge \\ y_i = \dfrac{\langle \vec{x}_i, \vec{x}_0 \rangle}{\langle \vec{x}_i, \vec{x}_i \rangle} \wedge \\ \text{com}_i = (v)^{s_i}. \end{array}\right\}$$

where $\vec{w}^{\vec{x}_i} = \prod_{j\in[m]} (w_j)^{x_{i,j}}$. Outputs ciphertext $C_{\ell,i}$ and proof $\pi_{CT,i}$.

- **`VerifyCT`**$((C_{\ell,i})_{i\in[n]}, \pi_{CT}, vk_{CT}) \to 1$ **or** 0**:** Takes as input the ciphertexts $(C_{\ell,i})_{i\in[n]}$ and proofs π_{CT} from n client and the verification keys vk_{CT}. For $i \in [n]$: it verifies $\pi_{CT,i}$ for the relation $\mathcal{R}_{Encrypt}$ and outputs 1 for accepting or 0 with a set of malicious clients $\mathcal{CC}_{CT} \neq \emptyset$ for rejecting.

- **`DKeyGenShare`**$(sk_i, pk, y_i, \ell_y, pp) \to (dk_{y,i}, \pi_{DK,i})$**:** Takes as input a client secret key $sk_i = (s_i, \hat{k}_i, t_i)$, public key pk, the function coefficient y_i and a function label ℓ_y. It computes dk_i as

$$dk_i = \left((\hat{v}_b)^{\hat{k}_i} \cdot h^{s_{i,b} \cdot y_i}\right)_{b\in[2]}$$

where $\hat{v}_b = \mathcal{H}_2(\ell_{y,b}) \in \mathbb{G}_2^2, b \in [2]$. It re-computes com_i and generates a proof $\pi_{DK,i}$ for the relation $\mathcal{R}_{DKeyGenShare,i}$ on input $\left(pk, com, dk_{y,i}, \ell_y\right)$. The $\pi_{DK,i}$ is described as

$$\text{PoK}\left\{(s_i, t_i, \hat{k}): \begin{array}{r} T_i = h_p^{t_i} \wedge \\ d_i = (f^{\hat{k}_b} K_{\sum,i,b}^{t_{i,b}})_{b\in[2]} \wedge \\ dk_i = \left((\hat{v}_b)^{\hat{k}} \cdot h^{s_{i,b} \cdot y_i}\right)_{b\in[2]} \wedge \\ \text{com}_i = (v)^{s_i} \end{array}\right\}$$

where $K_{\sum,i,b} = \prod_{i<i^*}^{n} T_{i^*,b} \cdot \left(\prod_{i>i^*}^{n} T_{i^*,b}\right)^{-1} \in G$, $\hat{v}_b = \mathcal{H}_2(\ell_{y,b}), b \in [2], v = \mathcal{H}_1(\ell_D)$. Outputs a functional decryption key share $dk_{y,i}$ and $\pi_{DK,i}$.

- **`VerifyDK`**$((dk_{y,i})_{i\in[n]}, \pi_{DK}, vk_{DK}, pp) \to 1$ **or** 0**:** Takes as input decryption key shares $(dk_{y,i})_{i\in[n]}$, proof π_{DK} from n clients and the verification key vk_{DK}. For $i \in [n]$: it verifies $\pi_{DK,i}$ for the relation $\mathcal{R}_{DKeyGenShare}$ and outputs 1 for accepting or 0 with a set of malicious clients $\mathcal{CC}_{DK} \neq \emptyset$ for rejecting.

- **`DKeyComb`**$((dk_{y,i})_{i\in[n]}, \ell_y, pk) \to dk$**:** Takes as input all of the functional decryption key shares $(dk_{y,i})_{i\in[n]}$, a function label ℓ_y and public key pk. It computes $f^d = \left(\prod_{i\in[n]} d_{i,b}\right)_{b\in[2]}$ and solves the discrete logarithm problem on $\mathcal{DG}$ to obtain d, then computes h^{dk_y} as

$$h^{dk_y} = \left(\frac{\prod_{i\in[n]} dk_{i,b}}{(\hat{v}_b)^d}\right)_{b\in[2]}$$

where $\hat{v}_b = \left(\mathcal{H}_2(\ell_{y,b})\right), b \in [2]$. Outputs the functional decryption key $dk = h^{dk_y}$.

- **`Decrypt`**$\left(\left(C_{\ell,i}\right)_{i\in[n]}, dk, y\right) \to \left(\langle \vec{X}_j, \vec{y} \rangle\right)_{j\in[m]}$ **or** $\perp$ **:** Takes as input n client with m-dimensions vector ciphertext $\left(C_{\ell,i}\right)_{i\in[n]}$, a functional decryption key h^{dk_y} and the function vector y. It computes $u = (u_1, u_2) = \mathcal{H}_1(l)$, and $\left(E_j\right)_{j\in[m]} = \left(e(w_j, h)\right)_{j\in[m]}$, where $w_j =$

$\mathcal{H}_1'(\ell_{Enc,j})$. For $j \in [m]$, it gets

$$E_j^{\langle \vec{X}_j, \vec{y} \rangle} = \frac{\prod_{i\in[n]} e(c_{l,i,j}, h^{y_i})}{e(u_1, h^{dk_{y,1}})e(u_2, h^{dk_{y,2}})}$$

and solves the discrete logarithm in basis E_j to return $\langle \vec{X}_j, \vec{y} \rangle$, where $\vec{X}_j = (x_{1,j}, x_{2,j}, \ldots, x_{n,j}) \in \mathbb{R}^n$ denotes the vector consisting of the j-th component of each of n clients. It outputs $(\langle \vec{X}_j, \vec{y} \rangle)_{j\in[m]}$ or $\perp$.

Correctness. We have

$$\begin{aligned} f^{\sum_{i\in[n]} \hat{k}_i} &= \prod_{i\in[n]} d_i \\ d &= \sum_{i\in[n]} \hat{k}_i = \log_f f^{\sum_{i\in[n]} \hat{k}_i} \\ h^{dk_y} &= \left(\frac{\prod_{i\in[n]} dk_{i,b}}{(\hat{v}_b)^d} \right)_{b\in[2]} \\ &= \left(h^{\sum_{i\in[n]} s_{i,b}\cdot y_i} \right)_{b\in[2]} \end{aligned}$$

$$\begin{aligned} E_j^{\langle \vec{X}_j, \vec{y} \rangle} &= \frac{\prod_{i\in[n]} e(c_{l,i,j}, h^{y_i})}{e(u_1, h^{dk_{y,1}})e(u_2, h^{dk_{y,2}})} \\ &= \frac{\prod_{i\in[n]} e(u^{s_i} \cdot (w_j)^{x_{i,j}}, h^{y_i})}{e(u_1, h^{dk_{y,1}})e(u_2, h^{dk_{y,2}})} \\ &= e(w_j, h)^{\langle \vec{X}_j, \vec{y} \rangle} \end{aligned}$$

where $\vec{X}_j = (x_{1,j}, x_{2,j}, \ldots, x_{n,j}) \in \mathbb{R}^n$ denotes the vector composed of the j-th parameter from n clients, and then solves the discrete logarithm in basis E_j to return $\langle \vec{X}_j, \vec{y} \rangle$. Performing m decryption operations on $\left(\left(C_{\ell,i,j} \right)_{i\in[n]} \right)_{j\in[m]}$ returns the result $\left(\langle \vec{X}_j, \vec{y} \rangle \right)_{j\in[m]}$.

Theorem 1. *The decentralized verifiable multi-client functional encryption for inner product is static-IND-secure under the DDH, multi-DDH and HSM assumption (see full version [33] for specific details). More precisely, we have*

$$\begin{aligned} \mathrm{Adv}^{\mathrm{sta-ind}}_{\mathrm{CC-DVFE}}(t, q_E, q_K) &\le q_E \mathrm{Adv}^{\mathrm{zk}}_{\prod_{Encrypt}}(t) \\ &+ q_K \mathrm{Adv}^{\mathrm{zk}}_{\prod_{DKeyGenShare}}(t) + \mathrm{Adv}^{\mathrm{std-ind}}_{\mathrm{LDSUM}}(t, q_K) \\ &+ 2q_E(2\mathrm{Adv}^{ddh}_{\mathbb{G}_1}(t) + \frac{1}{p}) + 2\mathrm{Adv}^{ddh}_{\mathbb{G}_1}(t + 4q_E \times t_{\mathbb{G}_1}) \end{aligned}$$

where

- $\mathrm{Adv}^{\mathrm{sta-ind}}_{\mathrm{CC-DVFE}}(t, q_E, q_K)$ *is the best advantage of any PPT adversary running in time t with q_E encryption queries and q_K key share queries against the static-IND-secure game of the CC-DVFE scheme;*
- $\mathrm{Adv}^{\mathrm{zk}}_{\prod_{Encrypt}}(t)$ *is the best advantage of any PPT adversary running in time t against the zero-knowledge property of the $\prod_{Encrypt}$ scheme;*
- $\mathrm{Adv}^{\mathrm{zk}}_{\prod_{DKeyGenShare}}(t)$ *is the best advantage of any PPT adversary running in time t against the zero-knowledge property of the $\prod_{DKeyGenShare}$ scheme;*
- $\mathrm{Adv}^{\mathrm{std-ind}}_{\mathrm{LDSUM}}(t, q_K)$ *is the best advantage of any PPT adversary running in time t with q_K encryption queries against the IND-security game of the LDSUM scheme [15].*
- $\mathrm{Adv}^{ddh}_{\mathbb{G}_1}(t)$ *is the best advantage of breaking the DDH assumption in $\mathbb{G}_1$ within time t.*

The proof is provided in the full version [33].

Theorem 2. *The decentralized verifiable multi-client functional encryption for inner product supports verifiability in the random oracle (see full version [33] for specific details). More precisely, we have*

$$\begin{aligned} &\mathrm{Adv}^{verif}_{CC\text{-}DVFE}\left(t, q_C, q_F\right) \le \epsilon_1 + \\ &q_C \cdot \max\left\{ \mathrm{Adv}^{snd}_{\prod_{Encrypt}}(t), q_F \cdot \mathrm{Adv}^{snd}_{\prod_{DKeyGenShare}}(t) \right\} \end{aligned}$$

where

- $\mathrm{Adv}^{verif}_{CC\text{-}DVFE}\left(t, q_C, q_F\right)$ *is the best advantage of any PPT adversary running in time t against the verifiability game with q_C corruption queries and q_F functions for the finalization phase;*
- $\mathrm{Adv}^{snd}_{\prod_{Encrypt}}(t)$ *is the best advantage of any PPT adversary running in time t against the soundness of $\prod_{Encrypt}$;*
- $\mathrm{Adv}^{snd}_{\prod_{DKeyGenShare}}(t)$ *is the best advantage of any PPT adversary running in time t against the soundness of $\prod_{DKeyGenShare}$.*

The proof is provided in the full version [33].

The zero-knowledge proof algorithms related to `Encrypt` and `VerifyCT` are instantiated in the full version [33]. while those related to `DKeyGenShare` and `VerifyDK` are instantiated in in the full version [33]. The above two zero-knowledge proofs are transformed into the non-interacting zero-knowledge proofs using the Fiat-shamir transform [34], which can be applied to the CC-DVFE scheme.

6. VFEFL DESIGN

6.1. High-Level Description of VFEFL

The overview of VFEFL is depicted in the Figure 2. It utilizes the CC-DVFE to achieve a privacy and robustness FL among Server $\mathcal{S}$ and multiple clients $\left(C_i\right)_{i\in[n]}$. Specifically, VFEFL consists of three phases.

1. **Setup Phase**: $\mathcal{S}$ initializes the the public parameters pp, the set of clients $\mathcal{CS}$, and the global initial model W_0, which are sent to $\mathcal{CS}$ (Step (1)). Each client independently generates its keys, broadcasts the corresponding the public keys and verification keys (Step (2)).

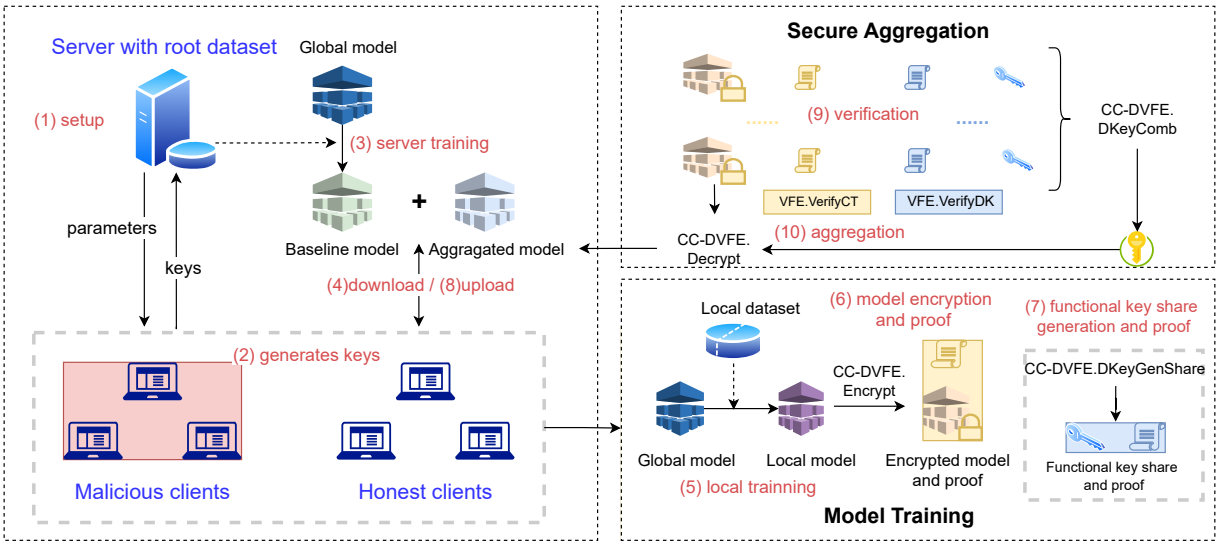


Figure 2: The workflow and main components of VFEFL

2. **Model Training Phase**: In the t-th training epoch, denoting the global model from the previous round as W^{t-1}, $\mathcal{S}$ updates the baseline model W_0^t based on D_0 (Step (3)) and broadcasts it. Each participant downloads the W_0^t and W^{t-1} (Step (4)), trains a local model W_i^t from W^{t-1} based on the local dataset D_i (Step (5)), encrypts it (Step (6)), and generates the functional key share (Step (7)) along with corresponding proofs, which are then sent to $\mathcal{S}$ (Step (8)).
3. **Secure Aggregation Phase**: $\mathcal{S}$ collects and verifies all encrypted models $\left(E_i\right)_{i\in\mathcal{CS}}$ and functional key shares with the proofs (Step (9)). If all verifications pass, the process moves to the next step; otherwise, the server $\mathcal{S}$ requires some clients to resend the ciphertext or key share based on the verification results and re-verifies them. $\mathcal{S}$ aggregates the authenticated functional key share, decrypts the ciphertexts, and normalizes the decryption results to get the final global update W^t (Step (10)) and send it to the $\mathcal{CS}$

6.2. Setup Phase

The server $\mathcal{S}$ collects all participating clients to initialize the client set $\mathcal{CS}$ and randomly sets the initial model parameters W_0. Then, the server runs the `CC-DVFE.SetUp` algorithm to generate the public parameters pp and broadcasts $(\mathcal{CS}, W_0, pp)$. Each client $\mathcal{C}_i$ interactively executes `CC-DVFE.KeyGen`(pp) and eventually generate its keys pair $(ek_i = s_i,\ sk_i = (s_i, \hat{k}_i, t_i),\ pk_i = (T_i, d_i))$, and verification keys $(vk_{CT,i} = \mathrm{com}_i, vk_{CT,i} = (T_i, d_i, \mathrm{com}_i))$ according to $\mathcal{CS}$. Each client sends the public key pk_i, the verification keys $vk_{CT,i}$ and $vk_{DK,i}$ to $\mathcal{S}$.

6.3. Model Training Phase

Model training consists of three parts: local training, model encryption and proof and functional key share generation and proof.

1. **Local Training:** In the t-th training iteration, after retrieving the global model W^{t-1} from the server, each client $\mathcal{C}_i$ optimizes the local model using its private dataset D_i by minimizing the corresponding loss function Loss to obtain the updated model W_i^t. For benign clients, the local model is updated using stochastic gradient descent (SGD) for R_l iterations with a local learning rate lr. Malicious clients may choose a random vector as their model, or deliberately craft it so that the global model deviates as much as possible from the correct direction. Similarly, $\mathcal{S}$ trains the baseline model W_0^t for this epoch t based on the root dataset D_0 and broadcast it to the $\mathcal{C}$.
2. **Model Encryption and Proof:** $\mathcal{C}_i$ encrypts and generates a proof of it before uploading. Specifically, for an honest client, $\mathcal{C}_i$ applies the `CC-DVFE.Encrypt`$(ek_i, W_i^t, W_0^t, \ell, \left(\ell_{Enc,j}\right)_{j\in[m]}, pp)$ algorithm for encryption and gains $E_{\ell,i}$ and $\pi_{CT,i}$. Then $\mathcal{C}_i$ sends then $\pi_{CT,i}$ to the $\mathcal{S}$. For malicious clients, they may not use the correct encryption key or the correct encryption algorithm for encryption in order to disrupt the aggregation process.
3. **Functional Key Share Generation and Proof:** For an honest client, $\mathcal{C}_i$ takes y_i from the previous step and processes it using the ReLU function, resulting in $y_i' = \mathrm{ReLU}(y_i)$. $\mathcal{C}_i$ executes `CC-DVFE.DKeyGenShare`$(sk_i, pk, y_i', \ell_y, pp)$ to generate the corresponding decryption key share dk_i and the proof $\pi_{DK,i}$ for it. For a malicious client, $\mathcal{C}_i$ may try to use an invalid secret key or provide a malformed decryption key share, aiming to compromise the decryption result.

6.4. Secure Aggregation Phase

The secure aggregation phase consists of three stages: ciphertext verification , functional key share verification and aggregation.

1. **Ciphertext Verification:** In this phase, the server $\mathcal{S}$ verifies all encrypted models $\left(E_{\ell,i}\right)_{i\in\mathcal{CS}}$ from clients using $\texttt{CC-DVFE.VerifyCT}\left((C_{\ell,i})_{i\in\mathcal{CS}}, \pi_{CT,i}, vk_{CT}\right)$. If all ciphertexts are valid, the process continues. Otherwise, the server identifies malicious clients $\mathcal{CC}_{CT}$, and requires all the clients in $\mathcal{CC}_{CT}$ to regenerate the ciphertext.
2. **Functional Key Share Verification:** In this phase, the server $\mathcal{S}$ collects and verifies all functional key shares from clients by executing $\texttt{CC-DVFE.VerifyDK}$ $((dk_{y,i})_{i\in\mathcal{CS}}, \pi_{DK,i}, vk_{DK}, pp)$. If all ciphertexts pass the verification, the process proceeds to the next phase. Otherwise, the set of malicious clients $\mathcal{CC}_{DK}$ whose key shares failed verification is identified and $\mathcal{S}$ requires all clients in $\mathcal{CC}_{DK}$ to regenerate the functional key share.
3. **Aggregation:** In this phase, the server $\mathcal{S}$ performs $\texttt{CC-DVFE.DKeyComb}\left(\left(dk_{y,i}\right)_{i\in\mathcal{CS}}, \ell_y, pk\right)$ to aggregate the verified functional keys shares and obtain the decryption key dk, and $\texttt{CC-DVFE.Decrypt}(\left(E_{\ell,i}\right)_{i\in\mathcal{CS}}, dk,$ $\vec{y})$ is executed to decrypt the verified models to gain W^*. Then $\mathcal{S}$ further normalizes W^* to obtain W^t according to the baseline model W_0^t. Finally, $\mathcal{S}$ obtains the global model W^t for the epoch t and send it to the $\mathcal{CS}$ for the next training epoch utill the global model converges.

VFEFL repeats the model training and secure aggregation phases until the global model converges, resulting in the final global model.

7. Analysis

In this section, we focus on a theoretical analysis of the design goals defined in 3.2, including privacy, robustness, and verifiability, among others and provide formal arguments.

7.1. Privacy

Theorem 3. *According to the security model of VFEFL, we say this VFEFL is secure if any view* $\boldsymbol{REAL}_{\mathcal{A}}^{VFEFL}$ *of* $\mathcal{A}$ *in the real world is computationally indistinguishable from* $\boldsymbol{IDEAL}_{\mathcal{A}*}^{f_{\text{VFEFL}}}$ *of* $\mathcal{A}^*$ *in the ideal world, namely*

$$\boldsymbol{REAL}_{\mathcal{A}}^{\text{VFEFL}}(\lambda, \hat{x}_{\mathcal{S}}, \hat{x}_{\mathcal{C}}, \hat{y}) \overset{c}{\approx} \boldsymbol{IDEAL}_{\mathcal{A}^*}^{f_{\text{VFEFL}}}(\lambda, \hat{x}_{\mathcal{S}}, \hat{x}_{\mathcal{C}}, \hat{y}),$$

where $\hat{y}$ *denotes intermediate results,* $\hat{x}_{\mathcal{S}}, \hat{x}_{\mathcal{C}}$ *denote the inputs of* $\mathcal{S}$ *and* $\mathcal{C}$.

Proof 1. *To prove the security of VFEFL, it suffices to demonstrate the confidentiality of the privacy-preserving defense strategy, as it is the only component involving the computation of private data of clients, for the unauthorized entities* $\mathcal{S}^*$ *(the server or eavesdrops). For the curious entities* $\mathcal{S}^*$, $\boldsymbol{REAL}_{\mathcal{A}}^{\text{VFEFL}}$ *contains intermediate parameters and encrypted local models* $(E_i^t)_{i=[n]}$ *with corresponding proofs* $(\pi_{CT,i})_{i=[n]}$, *collected during the execution of the VFEFL protocols. Additionally, we construct a PPT simulator* $\mathcal{A}^*$ *to execute* f_{VFEFL}, *simulating each step of the privacy-preserving defensive aggregation strategy. The detailed proof is provided below.*

Hyb1. *We initialize a set of system parameters based on distributions indistinguishable from those in* $\boldsymbol{REAL}_{\mathcal{A}}^{\text{VFEFL}}$ *during the actual protocol execution, including public parameters pp, initial models* W_0, *and the keys.*

Hyb2. *In this hybrid, we alter the behavior of the simulated client* C_i^*, *having it use the selected random vector* Θ_i^t *as its local model* W_i^t. *As only the original contents of ciphertexts have changed, the static-IND security property of VFEFL guarantees that* $\mathcal{S}^*$ *cannot distinguish the view of* Θ_i^t *from the view of original* W_i^t. *And in* $\pi_{Encrypt,i}$, *the* y_i *also be sent to the server. Existing reconstruction attacks [35] cannot recover the model vector with limited information of* $\|W_i^t\|$. y *can be viewed as a variant of* $\|W_i^t\|$ *and* $cosine(W_i^t, W_0^t)$ *(which is also revealed in [18]) when baseline models* W_0^t *are known. Thus, It is a negligible security threat to* W_i^t.

Hyb3. *In this hybrid, the server uses zero-knowledge* $\pi_{Encrypt}$ *of encrypted random vector* Θ_i^t *instead of blinded models* W_i^t *to verify the ciphertexts to obtain the result of* 1 *or* 0, *without knowing the further information of inputs. Hence, this hybrid is indistinguishable from the previous one.*

Hyb4. *In this hybrid, the decryption key is computed by* $\texttt{CC-DVFE.DKeyComb}$ *and the aggregated model* W^**(before normalization) is computed by the* $\texttt{CC-DVFE.Decrypt}$ *algorithm.* $\mathcal{S}^*$ *can obtain* $\Theta^* = \sum_{i\in\mathcal{C}}\left(y_i\Theta_i^t\right)$. *In addition, the local model* Θ_i^t *is encrypted while the randomness of these functional key shares is still preserved. This randomness ensures that the data is not compromised during the aggregation and decryption process. The probability of* $\mathcal{S}^*$ *guessing the specific* Θ_i^t *correctly is negligible. Hence, this hybrid is indistinguishable from the previous one.*

In ***Hyb4****,* $\mathcal{A}^*$ *holds the view* $\boldsymbol{IDEAL}_{\mathcal{A}^*}^{f_{\text{VFEFL}}}$, *which is converted from* $\boldsymbol{REAL}_{\mathcal{A}}^{\text{VFEFL}}$ *and is indistinguishable from it. Thus, the above analysis demonstrates that the output of* $\boldsymbol{IDEAL}_{\mathcal{A}^*}^{f_{\text{VFEFL}}}$ *is computationally indistinguishable from the output of* $\boldsymbol{REAL}_{\mathcal{A}}^{\text{VFEFL}}$. *Therefore, we conclude the proof of the privacy of VFEFL.*

7.2. Robustness

We provide provable guarantees on the Byzantine robustness of the proposed defensive aggregation. It is shown that under the following assumptions, the difference between the global model W^t learned under attack by the aggregation condition and the optimal global model W^{opt} is bounded. Next, we present the necessary assumptions, followed by our main theorem.

Assumption 1. The expected loss function $F(w)$ is μ-strongly convex and differentiable over the space Θ with an L-Lipschitz continuous gradient. Formally, for any $w, \hat{w} \in \Theta$, the following conditions hold:

$$F(w) \geq F(\hat{w}) + \langle \nabla F(\hat{w}), (w - \hat{w}) \rangle + \frac{\mu}{2}\|w - \hat{w}\|^2.$$

$$\|\nabla F(w) - \nabla F(\hat{w})\| \leq L\|w - \hat{w}\|.$$

Assumption 2. The local training dataset D_i (for $i = 1, 2, \ldots, n$) of each client and the root dataset D_0 are independently sampled from the distribution $\mathcal{X}$.

Theorem 4. *For any number of malicious clients, the difference between the global model W^t learned by VFEFL and the optimal global model W^{opt} (in the absence of attacks) is bounded.*

$$\|W^t - W^{opt}\| \leq$$
$$3\left(\sqrt{(1 - 2\eta\mu + \eta^2 L^2)}\right)^{lr} \|W_0 - W^{opt}\| + 2\|W^{opt}\|$$

where η is the local learning rate and W_0 is the initial model. When $\|\sqrt{(1 - 2\eta\mu + \eta^2 L^2)}\| < 1$, we have $\mathrm{Lim}_{t\to\infty} |W^t - W^{opt}\| \leq 2\|W^{opt}\|$.

The proof is provided in the full version [33].

7.3. Verifiability, self-contained and fidelity

The verifiability of VFEFL is inherently dependent on the verifiability of CC-DVFE Scheme. This dependency ensures two critical properties: it prevents malicious clients from framing honest ones, and ensures that authenticated ciphertexts and keys can be correctly decrypted. Thus, the clients successfully proves to the server that both the inputs and behavior follow the protocol specification without compromising the privacy of the model.

The proposed scheme is self-contained, as it can be deployed in the basic federated learning framework without two non-colluding servers and any additional trusted third parties. Furthermore, it achieves high fidelity by ensuring VFEFL introduces no noise during aggregation, thereby preserving the accuracy of the global model in the absence of attacks. This fidelity is validated through experiments in Section 8, demonstrating the scheme's reliability in maintaining model performance.

8. EXPERIMENTS

8.1. Experimental Settings

In this section, we evaluate the model accuracy, Byzantine robustness and efficiency of VFEFL.

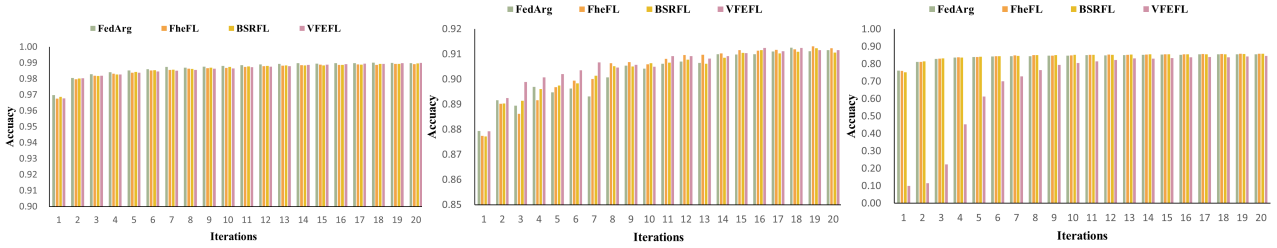
1) Implementation. We conduct our experiments on a local machine equipped with the 13th Gen Intel(R) Core(TM) i5-1340P CPU@1.90GHz and 16GB RAM. We also approximate floating-point numbers to two decimal places and convert them into integers for cryptographic computations, a common practice in cryptographic schemes. The final decryption step involves computing a discrete logarithm, which introduces a substantial computational overhead. Specifically, the problem reduces to solving for x in the equation $a = g^x$, where g is an element of a bilinear group and x is a relatively small integer. To efficiently tackle this, we utilize the baby-step giant-step algorithm [36], which is known for its effectiveness in computing discrete logarithms in such contexts.

2) Dataset and Models. On data set allocation, we set aside a portion of the dataset as a trusted root dataset D_0 for the server and then randomly divide the remaining data into clients. We use multiple datasets from different domains, and evaluate the accuracy of the final global model on the entire test set. The publicly available datasets are the following.

- **MNIST:** The dataset consists of handwritten digits, organized into 10 classes, with 60,000 samples for training and 10,000 samples for testing. Each sample is a grayscale image with a resolution of 28×28 pixels.
- **Fashion-MNIST:** The dataset consists of grayscale images of fashion items, divided into 10 categories. It provides 60,000 training samples and 10,000 test samples, with each image having dimensions of 28×28 pixels.
- **CIFAR-10:** This dataset consists of color images belonging to 10 distinct classes, with 50,000 training samples and 10,000 test samples. Each image has a resolution of 32×32 pixels in RGB format.

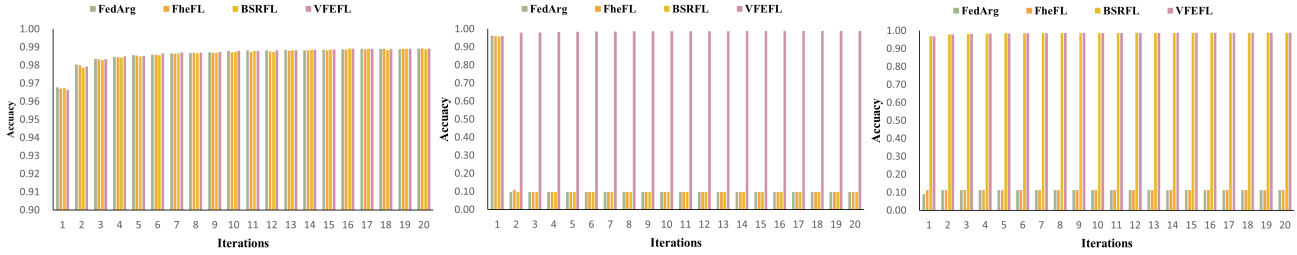
3) Attack Scenario. We investigate four types of attacks: Gaussian attack, Scaling attacks, Adaptive attack and Label Flip attack. In our experiments, 20% of clients are designated as malicious under untargeted poisoning attacks, while this proportion is reduced to 10% for targeted poisoning attacks. The specific details of each attack are outlined as follows:

- **Gaussian Attack (GA) [28] :** Malicious clients may make the server's global model less accurate by uploading random models. This attack randomly generates local models on compromised clients, which is designed using the approach and parameters from [28].
- **Scaling Attack (SA):** On the basis of the above Gaussian attack, the malicious client enhances the attack by amplifying its own model parameters by a scaling factor $\lambda \gg 1$ before submitting them to the server, thereby disproportionately influencing the global model. Based on prior work, we set the scaling factor $\lambda = n$, where n represents the number of clients.
- **Adaptive Attack (AA) [17]:** For each aggregation rule, we define a corresponding objective function that quantifies the gap between the global model aggregated solely from honest clients and the global model after the participation of malicious clients and the solution of the optimization problem is used as a model for the attack, enabling the malicious client models to maximize the attack's effectiveness. We use the default parameter settings in [17] and the inverse direction of the average of all models for each dimension as the initial malicious model.
- **Label Flipping Attack (LFA):** Our work adopts the label-flipping attack strategy outlined in [17]. Specifically, in all experiments, the malicious clients flip its



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10

Figure 3: Model accuracy on three datasets in the absence of attacks: (a) MNIST, (b) Fashion-MNIST, and (c) CIFAR-10.



(a) GA (b) SA (c) AA

Figure 4: Model accuracy under different attacks on MNIST: (a) GA, (b) SA, and (c) AA.

label l to $M - l - 1$, where M is the total number of labels and $l \in \{0, 1, \cdots, M - 1\}$.

4) Evaluation Metrics. In this experiment, we use the accuracy (AC) of the global model as the primary performance metric. For the LF attack, in addition to model accuracy, we also consider the attack success rate (ASR) as an evaluation criterion. The attack success rate is defined as the probability that a sample with label l is misclassified as $M - l - 1$.

8.2. Experimental Results

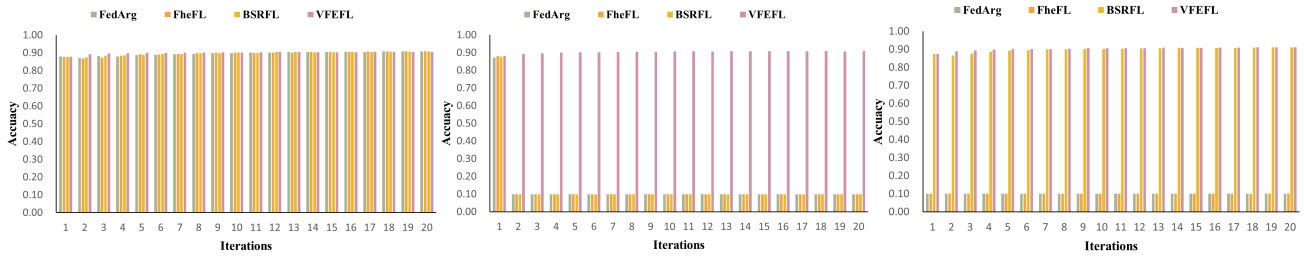
VFEFL achieves the three defense goals: First, VFEFL achieves fidelity by maintaining testing error rates comparable to FedAvg under benign conditions. Second, it demonstrates robustness by effectively countering attacks from malicious clients. Finally, its efficiency remains within acceptable bounds, ensuring feasibility for real-world deployment.

First, when there is no attack, our scheme's accuracy rates similar to FedAvg , thereby fulfilling the fidelity goal. As we can see in the Figure 3, the accuracy of our scheme is largely in line with FedAvg across multiple datasets. For instance, on MNIST, the accuracy of both FedAvg and VFEFL is close to 99%. Our aggregation rules take into account all client updates when there is no attack, so the accuracy of the global model is guaranteed. In the comparison in the table, we chose comparison schemes that are all capable of implementing fidelity protection.

Second, the proposed scheme demonstrates strong robustness against various attacks. As shown in Figures 4, 5, and 6, across three datasets, the global model trained with our method consistently maintains high accuracy under Gaussian, Scaling, and Adaptive attacks. In particular, the accuracy drop under Scaling attacks is less than 1%. Although the three baseline methods are also robust to Gaussian attacks, they are vulnerable to Scaling attacks due to the lack of constraints on local model magnitudes. This allows adversaries to amplify their influence by submitting models with large norms. In contrast, our scheme employs a robust aggregation rule that verifies and restricts local model norms, thereby sustaining high accuracy even under Scaling attacks. Under Adaptive attacks, both BSR-FL and VFEFL achieve high final model accuracy. These results demonstrate that VFEFL provides strong resilience against various untargeted poisoning attacks.

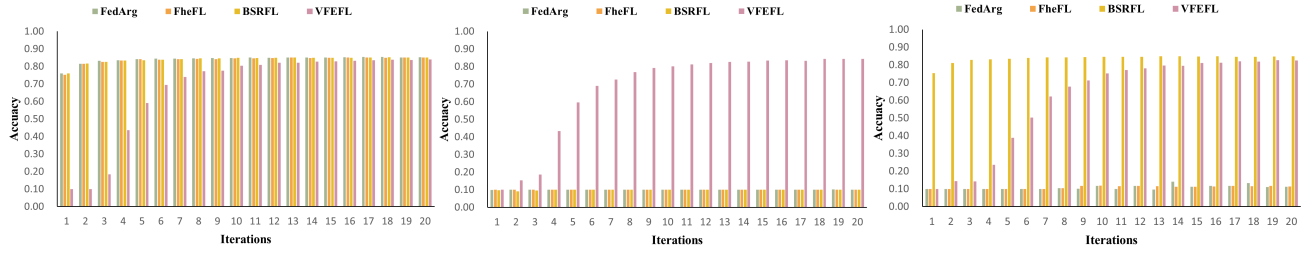
As shown in Figures 7, 8, and 9, VFEFL consistently achieves high accuracy under LF attacks, demonstrating strong robustness. On the CIFAR10 dataset (Fig. 9), model convergence is relatively slower due to the aggregation rule normalizing each local model's norm based on the baseline. This design mitigates malicious clients' attempts to amplify local model parameters, as evidenced by the scaling attack results in Figures 4b, 5b, and 6b, but increases convergence time. Nevertheless, after approximately 20 iterations, our scheme attains satisfactory accuracy. Moreover, on MNIST and Fashion-MNIST, the global model maintains high classification accuracy and exhibits superior defense and stability against attacks compared to other methods. These results further confirm the robustness and effectiveness of our approach in adversarial settings.

Finally, we conduct a comprehensive evaluation of the computational efficiency of VFEFL. First, a performance



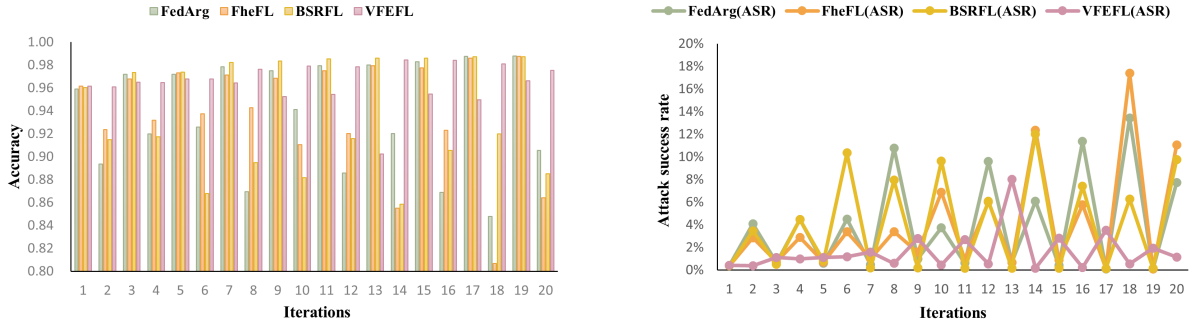
(a) GA (b) SA (c) AA

Figure 5: Model accuracy under different attacks on Fashion-MNIST: (a) GA, (b) SA, and (c) AA.



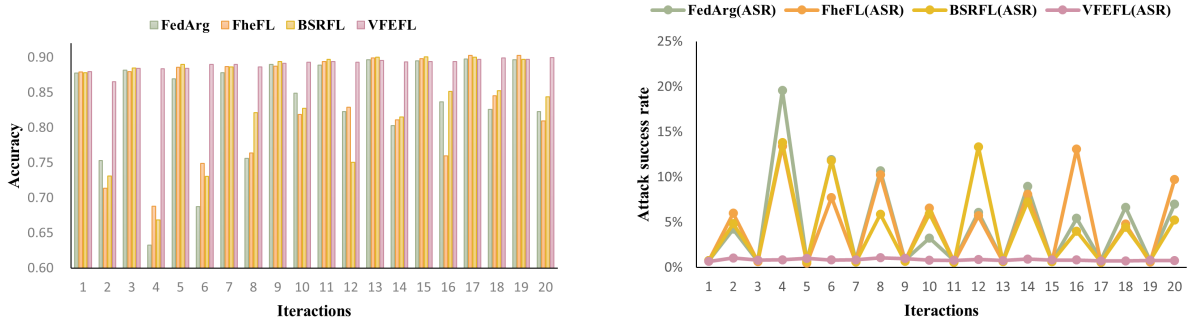
(a) GA (b) SA (c) AA

Figure 6: Model accuracy under different attacks on CIFAR-10: (a) GA, (b) SA, and (c) AA.



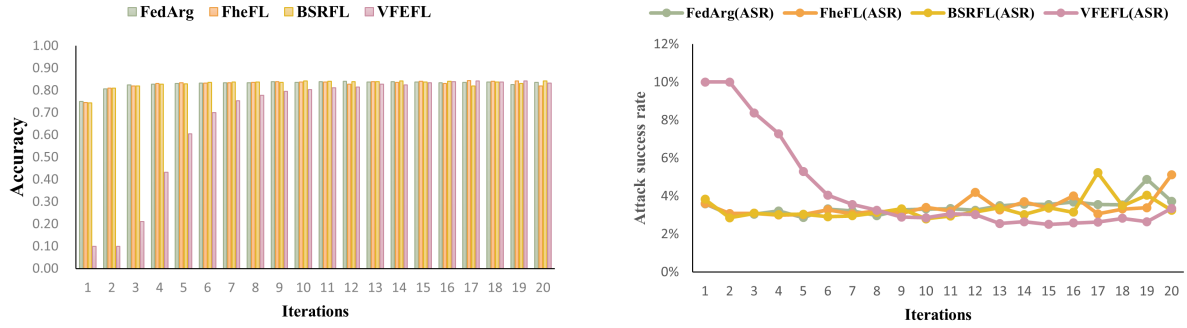
(a) AC under LF attacks (b) ASR under LF attacks

Figure 7: AC and ASR under LF attacks on MNIST: (a) accuracy (AC) and (b) attack success rate (ASR).



(a) AC under LF attacks (b) ASR under LF attacks

Figure 8: AC and ASR under LF attacks on Fashion-MNIST: (a) accuracy (AC) and (b) attack success rate (ASR).



(a) AC under LF attacks

(b) ASR under LF attacks

Figure 9: AC and ASR under LF attacks on CIFAR-10: (a) accuracy (AC) and (b) attack success rate (ASR).

Table 3
Time Consumption of CC-DVFE Under Different Dimensions

Operations		m = 10	m = 50	m = 100	m = 500	m = 1000
Setup		9.60 s	4.85 s	9.30 s	14.89 s	10.59 s
Key Generation		1.37 s	1.31 s	1.32 s	1.30 s	1.28 s
Encryption	Hash to point	0.02 s	0.11 s	0.22 s	1.06 s	2.09 s
	Encryption	0.01 s	0.02 s	0.04 s	0.21 s	0.48 s
	Ciphertext Proof	0.02 s	0.06 s	0.11 s	0.48 s	1.10 s
Verify Ciphertext		0.03 s	0.07 s	0.12 s	0.49 s	1.18 s
DKeyGenShare	Key Generation	0.01 s	0.02 s	0.02 s	0.02 s	0.02 s
	Key Proof	0.14 s	0.15 s	0.14 s	0.14 s	0.15 s
Verify Functional Key		0.22 s	0.23 s	0.22 s	0.22 s	0.24 s
Key Combination		0.02 s	0.02 s	0.02 s	0.02 s	0.02 s
Decryption		0.14 s	0.93 s	1.49 s	7.01 s	13.29 s

Table 4
Time Consumption of VFEFL Per Iteration (LeNet5, MNIST, $n = 10$, $m = 61706$)

	Operations	Time (s)
Setup Phase	Setup	87.54
	Communication	0.43
Model Training Phase	Local Training	99.39
	Hash to Pairing	128.61
	Model Encryption and Proof	87.55
	Key Share Generation and Proof	0.16
	Communication	12.38
Secure Aggregation Phase	Ciphertext Verification	593.93
	Key Share Verification	2.35
	Aggregation	1014.62

test of CC-DVFE is conducted, and the results are shown in Table 3. The time required for the Setup phase exhibits some uncertainty due to the need for trial and error in generating appropriate prime numbers within the GenClassGroup. The runtime of KeyGen, DKeyGenShare and VerifyDK scales linearly with the number of clients, while that of Encrypt, VerifyCT and Decrypt increases with the ciphertext dimensionality. In the Encrypt, hash to point operation is relatively time-consuming, however, it only needs to be executed once during the system construction. In this experiment, the Encrypt, KeyGen, and related verification operations are recorded as the average time for each client.

We further evaluate the time required to complete a full training iteration using the MNIST dataset under the VFEFL, and the relevant results are demonstrated in Table 4. In practical experiments, the Setup phase also includes some preprocessing operations related to solving the discrete logarithm problem. The communication time of the Setup Phase refers to the time taken for the server to send the initial model and the baseline model to the client, and that of the Model Training Phase refers to the time taken for the client to upload the encrypted model, the functional key share, and corresponding proofs to the server. These communication times are averaged over multiple clients. In the Secure Aggregation Phase, the server performs the verification of all ciphertexts and key shares, and the total verification time is computed. Despite the introduction of cryptographic operations, the overall training efficiency of the federated learning framework is still within an acceptable range, which verifies that the proposed scheme can effectively take into

account the computational performance while guaranteeing security.

9. CONCLUSION

In this paper, we propose a novel federated learning framework, VFEFL, to address the limitation that existing frameworks depend on the assumption of two non-colluding servers or trusted third parties. Specifically, we propose a new aggregation rule and a decentralized verifiable functional encryption scheme, and rigorously prove their security. Furthermore, experiments on real datasets demonstrate that our scheme achieves robustness and fidelity comparable to existing methods, while eliminating reliance on such trust assumptions and providing strong verification guarantees.

Sample CRediT author statement

Nina Cai: Conceptualization, Methodology, Software, Investigation, Validation, Formal analysis, Writing - Original Draft, Writing - Review & Editing. **Jinguang Han:** Conceptualization, Writing - Review & Editing, Supervision, Funding acquisition. **Weizhi Meng:** Writing - Review & Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grant 62372103, the Natural Science Foundation of Jiangsu Province under Grant BK20231149, the Jiangsu Provincial Scientific Research Center of Applied Mathematics under Grant BK20233002, and the Start-up Research Fund of Southeast University under Grant RF1028623300.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: AISTATS, PMLR, 2017, pp. 1273–1282.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, Found. Trends Mach. Learn. (2021) 1–210.

[3] Z. Alebouyeh, A. J. Bidgoly, Benchmarking robustness and privacy-preserving methods in federated learning, Future Gener. Comput. Syst. 155 (2024) 18–38.

[4] L. Zhu, Z. Liu, S. Han, Deep leakage from gradients, Advances in neural information processing systems 32 (2019).

[5] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: CCS 2015, Association for Computing Machinery, New York, NY, USA, 2015, pp. 1322–1333. URL: https://doi.org/10.1145/2810103.2813677. doi:10.1145/2810103.2813677.

[6] Z. Li, J. Zhang, L. Liu, J. Liu, Auditing privacy defenses in federated learning via generative gradient leakage, in: CVPR, 2022, pp. 10122–10132. doi:10.1109/CVPR52688.2022.00989.

[7] D. Chen, H. Qu, G. Xu, AegisFL: Efficient and flexible privacy-preserving Byzantine-robust cross-silo federated learning, in: ICML 2024, PMLR, 2024, pp. 7207–7219. URL: https://proceedings.mlr.press/v235/chen24ag.html.

[8] Y. Rahulamathavan, C. Herath, X. Liu, S. Lambotharan, C. Maple, Fhefl: Fully homomorphic encryption friendly privacy-preserving federated learning with byzantine users, 2024. URL: https://arxiv.org/abs/2306.05112. arXiv:2306.05112.

[9] D. Boneh, A. Sahai, B. Waters, Functional encryption: Definitions and challenges, in: TCC 2011, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 253–273.

[10] D. Boneh, A. Sahai, B. Waters, Functional encryption: a new vision for public-key cryptography, Commun. ACM 55 (2012) 56–64.

[11] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, H.-S. Zhou, Multi-input functional encryption, in: EUROCRYPT, Springer, 2014, pp. 578–602.

[12] M. Abdalla, R. Gay, M. Raykova, H. Wee, Multi-input inner-product functional encryption from pairings, in: EUROCRYPT, Springer, 2017, pp. 601–626.

[13] J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, D. Pointcheval, Decentralized multi-client functional encryption for inner product, in: ASIACRYPT 2018, Springer, 2018, pp. 703–732.

[14] S. Badrinarayanan, V. Goyal, A. Jain, A. Sahai, Verifiable functional encryption, in: ASIACRYPT 2016, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016, pp. 557–587.

[15] D. D. Nguyen, D. H. Phan, D. Pointcheval, Verifiable decentralized multi-client functional encryption for inner product, in: CT-RSA 2023, Springer, 2023, pp. 33–65.

[16] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: Byzantine tolerant gradient descent, in: NeurIPS 2017, Curran Associates, Inc., 2017.

[17] X. Cao, M. Fang, J. Liu, N. Z. Gong, Fltrust: Byzantine-robust federated learning via trust bootstrapping, in: NDSS 2021, 2021. doi:10.14722/ndss.2021.24434.

[18] H. Zeng, J. Li, J. Lou, S. Yuan, C. Wu, W. Zhao, S. Wu, Z. Wang, Bsr-fl: An efficient byzantine-robust privacy-preserving federated learning framework, IEEE Trans. Comput. (2024) 2096–2110.

[19] M. Shayan, C. Fung, C. J. M. Yoon, I. Beschastnikh, Biscotti: A blockchain system for private and secure federated learning, IEEE Trans. Parallel Distrib. Syst. (2021) 1513–1525.

[20] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, Y. Chen, Privacy-preserving collaborative deep learning with unreliable participants, IEEE Trans. Inf. Forens. Secur. (2020) 1486–1500.

[21] Y. Miao, R. Xie, X. Li, Z. Liu, K.-K. R. Choo, R. H. Deng, Efficient and secure federated learning against backdoor attacks, IEEE Trans. Dependable Secure Comput. (2024) 4619–4636.

[22] M. Hao, H. Li, G. Xu, H. Chen, T. Zhang, Efficient, private and robust federated learning, in: ACSAC 2021, ACM, New York, NY, USA, 2021, pp. 45–60. URL: https://doi.org/10.1145/3485832.3488014. doi:10.1145/3485832.3488014.

[23] Z. Lu, S. Lu, X. Tang, J. Wu, Robust and verifiable privacy federated learning, IEEE Trans. Artif. Intell. (2024) 1895–1908.

[24] Y. Chang, K. Zhang, J. Gong, H. Qian, Privacy-preserving federated learning via functional encryption, revisited, IEEE Trans. Inf. Forens. Secur. (2023) 1855–1869.

[25] Q. Wei, G. Rao, X. Wu, Contract-based hierarchical security aggregation scheme for enhancing privacy in federated learning, J. Inf. Secur. Appl. 85 (2024) 103857.

[26] W. Xu, X. Xu, H. Wang, Fed-ha: A privacy-preserving robust federated learning scheme based on homomorphic assessment, J. Inf.

Secur. Appl. 93 (2025) 104144.
[27] X. Qian, H. Li, M. Hao, G. Xu, H. Wang, Y. Fang, Decentralized multi-client functional encryption for inner product with applications to federated learning, IEEE Trans. Dependable Secure Comput. (2024) 5781–5796.
[28] M. Fang, X. Cao, J. Jia, N. Gong, Local model poisoning attacks to byzantine-robust federated learning, in: USENIX Sec., 2020, pp. 1605–1622.
[29] G. Castagnos, F. Laguillaumie, Linearly homomorphic encryption from ddh, in: CT-RSA 2015, Springer International Publishing, Cham, 2015, pp. 487–505.
[30] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, I. Tucker, Two-party ecdsa from hash proof systems and efficient instantiations, in: CRYPTO 2019, Springer-Verlag, Berlin, Heidelberg, 2019, pp. 191–221. doi:10.1007/978-3-030-26954-8_7.
[31] A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature problems, in: CRYPTO 1986, Springer-Verlag, Berlin, Heidelberg, 1987, pp. 186–194.
[32] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, Bulletproofs: Short proofs for confidential transactions and more, in: IEEE S&P 2018, 2018, pp. 315–334. doi:10.1109/SP.2018.00020.
[33] N. Cai, J. Han, W. Meng, Vfefl: Privacy-preserving federated learning against malicious clients via verifiable functional encryption, 2025. URL: https://arxiv.org/abs/2506.12846. arXiv:2506.12846.
[34] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in: CCS 1993, Association for Computing Machinery, 1993, p. 62–73. URL: https://doi.org/10.1145/168588.168596. doi:10.1145/168588.168596.
[35] J. Xu, C. Hong, J. Huang, L. Y. Chen, J. Decouchant, Agic: Approximate gradient inversion attack on federated learning, in: SRDS 2022, IEEE, 2022, pp. 12–22.
[36] D. Terr, A modification of shanks' baby-step giant-step algorithm, Math. Comput. (2000) 767–773.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: AISTATS, PMLR, 2017, pp. 1273–1282.
[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, Found. Trends Mach. Learn. (2021) 1–210.
[3] Z. Alebouyeh, A. J. Bidgoly, Benchmarking robustness and privacy-preserving methods in federated learning, Future Gener. Comput. Syst. 155 (2024) 18–38.
[4] L. Zhu, Z. Liu, S. Han, Deep leakage from gradients, Advances in neural information processing systems 32 (2019).
[5] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: CCS 2015, Association for Computing Machinery, New York, NY, USA, 2015, pp. 1322–1333. URL: https://doi.org/10.1145/2810103.2813677. doi:10.1145/2810103.2813677.
[6] Z. Li, J. Zhang, L. Liu, J. Liu, Auditing privacy defenses in federated learning via generative gradient leakage, in: CVPR, 2022, pp. 10122–10132. doi:10.1109/CVPR52688.2022.00989.
[7] D. Chen, H. Qu, G. Xu, AegisFL: Efficient and flexible privacy-preserving Byzantine-robust cross-silo federated learning, in: ICML 2024, PMLR, 2024, pp. 7207–7219. URL: https://proceedings.mlr.press/v235/chen24ag.html.
[8] Y. Rahulamathavan, C. Herath, X. Liu, S. Lambotharan, C. Maple, Fhefl: Fully homomorphic encryption friendly privacy-preserving federated learning with byzantine users, 2024. URL: https://arxiv.org/abs/2306.05112. arXiv:2306.05112.
[9] D. Boneh, A. Sahai, B. Waters, Functional encryption: Definitions and challenges, in: TCC 2011, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 253–273.
[10] D. Boneh, A. Sahai, B. Waters, Functional encryption: a new vision for public-key cryptography, Commun. ACM 55 (2012) 56–64.
[11] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, H.-S. Zhou, Multi-input functional encryption, in: EUROCRYPT, Springer, 2014, pp. 578–602.
[12] M. Abdalla, R. Gay, M. Raykova, H. Wee, Multi-input inner-product functional encryption from pairings, in: EUROCRYPT, Springer, 2017, pp. 601–626.
[13] J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, D. Pointcheval, Decentralized multi-client functional encryption for inner product, in: ASIACRYPT 2018, Springer, 2018, pp. 703–732.
[14] S. Badrinarayanan, V. Goyal, A. Jain, A. Sahai, Verifiable functional encryption, in: ASIACRYPT 2016, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016, pp. 557–587.
[15] D. D. Nguyen, D. H. Phan, D. Pointcheval, Verifiable decentralized multi-client functional encryption for inner product, in: CT-RSA 2023, Springer, 2023, pp. 33–65.
[16] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: Byzantine tolerant gradient descent, in: NeurIPS 2017, Curran Associates, Inc., 2017.
[17] X. Cao, M. Fang, J. Liu, N. Z. Gong, Fltrust: Byzantine-robust federated learning via trust bootstrapping, in: NDSS 2021, 2021. doi:10.14722/ndss.2021.24434.
[18] H. Zeng, J. Li, J. Lou, S. Yuan, C. Wu, W. Zhao, S. Wu, Z. Wang, Bsr-fl: An efficient byzantine-robust privacy-preserving federated learning framework, IEEE Trans. Comput. (2024) 2096–2110.
[19] M. Shayan, C. Fung, C. J. M. Yoon, I. Beschastnikh, Biscotti: A blockchain system for private and secure federated learning, IEEE Trans. Parallel Distrib. Syst. (2021) 1513–1525.
[20] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, Y. Chen, Privacy-preserving collaborative deep learning with unreliable participants, IEEE Trans. Inf. Forens. Secur. (2020) 1486–1500.
[21] Y. Miao, R. Xie, X. Li, Z. Liu, K.-K. R. Choo, R. H. Deng, Efficient and secure federated learning against backdoor attacks, IEEE Trans. Dependable Secure Comput. (2024) 4619–4636.
[22] M. Hao, H. Li, G. Xu, H. Chen, T. Zhang, Efficient, private and robust federated learning, in: ACSAC 2021, ACM, New York, NY, USA, 2021, pp. 45–60. URL: https://doi.org/10.1145/3485832.3488014. doi:10.1145/3485832.3488014.
[23] Z. Lu, S. Lu, X. Tang, J. Wu, Robust and verifiable privacy federated learning, IEEE Trans. Artif. Intell. (2024) 1895–1908.
[24] Y. Chang, K. Zhang, J. Gong, H. Qian, Privacy-preserving federated learning via functional encryption, revisited, IEEE Trans. Inf. Forens. Secur. (2023) 1855–1869.
[25] Q. Wei, G. Rao, X. Wu, Contract-based hierarchical security aggregation scheme for enhancing privacy in federated learning, J. Inf. Secur. Appl. 85 (2024) 103857.
[26] W. Xu, X. Xu, H. Wang, Fed-ha: A privacy-preserving robust federated learning scheme based on homomorphic assessment, J. Inf. Secur. Appl. 93 (2025) 104144.
[27] X. Qian, H. Li, M. Hao, G. Xu, H. Wang, Y. Fang, Decentralized multi-client functional encryption for inner product with applications to federated learning, IEEE Trans. Dependable Secure Comput. (2024) 5781–5796.
[28] M. Fang, X. Cao, J. Jia, N. Gong, Local model poisoning attacks to byzantine-robust federated learning, in: USENIX Sec., 2020, pp. 1605–1622.
[29] G. Castagnos, F. Laguillaumie, Linearly homomorphic encryption from ddh, in: CT-RSA 2015, Springer International Publishing, Cham, 2015, pp. 487–505.
[30] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, I. Tucker, Two-party ecdsa from hash proof systems and efficient instantiations, in: CRYPTO 2019, Springer-Verlag, Berlin, Heidelberg, 2019, pp. 191–221. doi:10.1007/978-3-030-26954-8_7.
[31] A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature problems, in: CRYPTO 1986, Springer-Verlag, Berlin, Heidelberg, 1987, pp. 186–194.

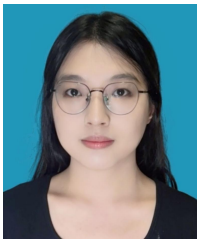
[32] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, Bulletproofs: Short proofs for confidential transactions and more, in: IEEE S&P 2018, 2018, pp. 315–334. doi:10.1109/SP.2018.00020.

[33] N. Cai, J. Han, W. Meng, Vfefl: Privacy-preserving federated learning against malicious clients via verifiable functional encryption, 2025. URL: https://arxiv.org/abs/2506.12846. arXiv:2506.12846.

[34] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in: CCS 1993, Association for Computing Machinery, 1993, p. 62–73. URL: https://doi.org/10.1145/168588.168596. doi:10.1145/168588.168596.

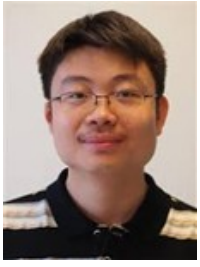
[35] J. Xu, C. Hong, J. Huang, L. Y. Chen, J. Decouchant, Agic: Approximate gradient inversion attack on federated learning, in: SRDS 2022, IEEE, 2022, pp. 12–22.

[36] D. Terr, A modification of shanks' baby-step giant-step algorithm, Math. Comput. (2000) 767–773.



Nina Cai received the Bachelor of Engineering degree from Jinan University, China, in 2019. She is currently pursuing the Master's degree at the School of Cyber Science and Engineering, Southeast University, China. Her research interests include cryptography and privacy-preserving federated learning.

Jinguang Han received the Ph.D. degree from the University of Wollongong, Australia, in 2013. He is a Professor with the School of Cyber Science and Engineering, Southeast University, China. His research focuses on access control, cryptography, cloud computing, and privacy-preserving systems. He served as a Program Committee Co-Chair for ProvSec'2016, FCS'2019, and SPNCE'2020; and as a Program Committee Member for several conferences, including SecureCom'2023, ISC'2022, PST'2021, ESORICS'2020, and ICICS'2019. He is a Senior Member of IEEE.



Weizhi Meng is a Full Professor in the School of Computing and Communications, Lancaster University, United Kingdom. He obtained his Ph.D. degree in Computer Science from the City University of Hong Kong. He was a recipient of the Hong Kong Institution of Engineers (HKIE) Outstanding Paper Award for Young Engineers/Researchers in 2014 and 2017. He also received the IEEE ComSoc Best Young Researcher Award for the Europe, Middle East & Africa Region (EMEA) in 2020. His primary research interests are blockchain technology, cybersecurity, and artificial intelligence in security, including intrusion detection, blockchain applications, smartphone security, biometric authentication, and IoT security. He serves as an Associate Editor or Editorial Board Member for many reputed journals such as *IEEE TDSC* and *IEEE TIFS*, and as a General Chair for various international conferences such as *ACM CCS 2023* and *ESORICS 2022*. He is a Senior Member of IEEE.