TMI-VFL: Secure Vertical Federated Learning via Threshold Multi-Identity Homomorphic Encryption

Yuqing Song

College of Artificial Intelligence, Tianjin University of Science and Technology, Tianjin 300457, China
ndyq_0708@qq.com

Abstract. Vertical Federated Learning (VFL) enables multiple participants to collaboratively train models using vertically partitioned data. However, during the actual training process, participants must exchange intermediate model representations(e.g., embeddings), which creates a potential attack surface for privacy leakage. Recent studies have shown that the URVFL attack achieves precise and covert data reconstruction by constructing malicious gradients and training a decoder using label information, posing a serious threat to the privacy security of vertical federated learning systems.To address this issue, we propose TMI-VFL, a secure training framework based on Threshold Multi-Identity Fully Homomorphic Encryption.This method establishes a ciphertext computation mechanism that ensures embedding vectors, gradients, and intermediate activation values are all processed in encrypted form, while the threshold decryption scheme prevents any single participant from recovering sensitive information. Experimental results show that under URVFL attacks, the proposed method increases reconstruction error by more than 10-fold, significantly reducing the effectiveness of the attack. Meanwhile, model accuracy decreases by less than 1% and remains close to baseline levels. These results indicate that TMI-VFL achieves an effective trade-off between privacy protection and model utility, providing a practical solution for secure VFL.

Keywords: Vertical Federated Learning,Data Reconstruction Attack,Fully Homomorphic Encryption,Privacy Protection

1 Introduction

In recent years, machine learning technologies—particularly deep neural networks—have achieved significant success in fields such as computer vision, financial risk control [7], and medical analytics. However, high-performance models typically rely on large amounts of high-quality training data, which is often scattered across different organizations and difficult to share directly. In scenarios such as financial risk control or precision marketing , different institutions may each hold distinct feature information about the same user group and wish to improve predictive capabilities through joint modeling. However, due to privacy protection requirements and restrictions imposed by relevant regulations (e.g., GDPR) [12], sharing raw data

across institutions has become impractical [8]. Vertical Federated Learning (VFL) offers a solution to this problem: participants collaboratively train models by exchanging intermediate model representations (e.g., embeddings or gradients) without sharing raw data. However, recent research indicates that these intermediate representations may still reveal participants' private features; attackers could even use them to reconstruct target data, thereby posing potential privacy risks [9].

As shown in Fig. 1, a vertical federated learning system typically consists of one active client and multiple passive clients. The active client is responsible for model training and aggregation, while the passive clients hold different subsets of features for the same user set. In a malicious attack scenario, the active client may exploit the intermediate representations obtained during training to reconstruct the private features of the passive clients.

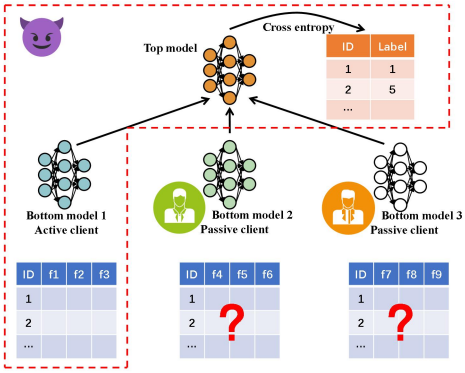

Fig. 1. Diagram of a Malicious Attack

Building on this, researchers have proposed various reconstruction attacks targeting VFL, such as optimization-based gradient inversion attacks and generative model-based feature recovery attacks [10][11].The recently proposed URVFL attack further enhances the stealth and effectiveness of such attacks. By introducing a discriminator with an auxiliary classifier , URVFL generates malicious gradients that are highly similar to normal training gradients in terms of statistical distribution, thereby achieving high-precision feature reconstruction while bypassing existing detection mechanisms [1]. This indicates that existing VFL systems still lack effective defense mechanisms against carefully designed malicious gradient attacks.

Therefore, an ideal secure vertical federated learning system should simultaneously satisfy three key requirements: data privacy, meaning that participants' raw feature data remains confidential throughout the training process; model privacy, meaning that model parameters should not be accessible to other participants; and attack r obustness, meaning the system must be able to withstand data reconstruction attacks, particularly covert attacks such as those in URVFL [2].However, existing methods still struggle to simultaneously meet these requirements. Previous research has primarily explored three directions: 1) One class of methods is based on a federated learning framework, achieving collaborative training through parameter aggregation; however, since the global model is shared among participants, it is difficult to guarantee model privacy [6].2) Another class of methods utilizes secure multi-party computation (MPC) to perform training in the

encrypted state. Although these methods offer strong security, they typically rely on the assumption of no collusion among multiple servers and incur high computational and communication overheads [15].3) Some works utilize homomorphic encryption to perform machine learning computations on ciphertext data; however, existing schemes are mostly designed for outsourced training scenarios involving a single data owner and still face efficiency and scalability challenges in multi-party collaborative environments [16].Research on privacy attacks against VFL is also evolving, ranging from early feature reconstruction methods based on optimization or generative models to the recent ly proposed URVFL attack, which can bypass existing detection mechanismsby constructing malicious gradients with normal statistical characteristics.A comparison of different methods in terms of data privacy, model privacy, and attack resistance is shown in TABLE I. Compared to these works, the TMI-VFL proposed in this paper achieves secure training without relying on the no-collusion assumption by combining multi-identity fully homomorphic encryption with a threshold decryption mechanism, while effectively resisting malicious gradient attacks.

Table 1. Comparison of Current Work with TMI-VFL

Category	Framework	Tech*	DP	MP	PMA	AUD
HBC	[10],[14]	Optimize/Generate	✗	✓	✗	-
Malicious attack	[11],[17]	GAN	✗	✗	✓	✗
Malicious attack	[1]	DAC	✗	✗	✓	✓
Disturbance defense	[18],[19]	Disturbance	✓	✓	Part	Part
Detection defense	[20]	Gradient analysis	-	-	Inspetion	✗
Cryptographic defense	[16]	HE	✓	Part	Semi-honest	✓
This work	**TMI-VFL**	**Multi-Identity FHE+ Threshold**	✓	✓	✓	✓

*HE：Homomorphic encryption，FHE：Fully homomorphic encryption，DAC：Discriminator with auxiliary classifier

DP: Data Privacy; MP: Model Privacy; PMA: Protection against Malicious Attacks; AUD: Anti-URVFL Detection

To address the aforementioned issues, this paper proposes TMI-VFL, a secure vertical federated learning training framework based on Threshold Multi-Identity Fully Homomorphic Encryption. By constructing a ciphertext computation channel during training, this method ensures that the model's intermediate representations and gradients remain in ciphertext form at all times, thereby effectively preventing attackers from exploiting intermediate information to perform feature reconstruction attacks.

(1) We propose TMI-VFL, a secure training framework for vertical federated learning that integrates multi-identity fully homomorphic encryption, threshold decryption, and secret sharing into the model training process, and designs a secure

protocol supporting neural network training. Throughout this process, embeddings, intermediate activation values, and gradients remain encrypted at all times, effectively preventing the leakage of sensitive information and achieving bidirectional privacy protection for both data and models.

(2) We conduct experimental evaluations on the CIFAR-10 dataset. The results demonstrate that while causing only a slight fluctuation in model classification accuracy ($< 1\%$), TMI-VFL increases the attack reconstruction error by more than an order of magnitude, achieving a good balance between privacy protection and model performance.

2 Preliminaries

2.1 Notations

Bold lowercase and uppercase letters denote vectors and matrices, respectively. $\mathbb{Z}_q$ denotes the integer ring modulo q. We use $x \leftarrow \mathcal{D}$ to denote that x is sampled from distribution $\mathcal{D}$. Unless otherwise specified, operations on vectors and matrices are applied element-wise.

2.2 Lattice-based Homomorphic Encryption

LWE problem. The security of lattice-based cryptography relies on the decisional Learning with Errors (LWE) assumption. Given a random matrix $\mathbf{A}$, a secret vector $\boldsymbol{s}$, and an error vector $\boldsymbol{e}$, the distribution$(\mathbf{A}, \mathbf{A}\boldsymbol{s} + \boldsymbol{e})$ is computationally indistinguishable from uniform random samples.

Tool matrix. The tool matrix $\mathbf{G}$ enables efficient homomorphic operations. There exists an efficient decomposition algorithm $\mathbf{G}^{-1}(\cdot)$ such that $\mathbf{A} \cdot \mathbf{G}^{-1}(\cdot)$ reconstructs the input.

Multi-id Identity-based Fully Homomorphic Encryption. We adopt a multi-identity fully homomorphic encryption scheme that supports computation over ciphertexts encrypted under different identities. Each identity ID has a corresponding secret key sk_{ID}, and plaintext m is encrypted as ciphertext c.

To support joint computation, ciphertexts from different identities are expanded into a unified form, allowing evaluation under a combined secret key. The ciphertext satisfies $c =< s, c' > + \boldsymbol{e}$ where $\boldsymbol{e}$ is controllable noise. The scheme supports homomorphic addition and multiplication.

2.3 Assumptions and Threat Model

We consider a vertical federated learning system consisting of a model owner (MO) and multiple data owner (DO). Each DO holds a subset of features, while the MO holds labels and maintains the model.

We assume a malicious but protocol-compliant adversary. Participants follow the protocol but may attempt to infer private information from intermediate results. In particular, the MO may launch reconstruction attacks (e.g., URVFL) to recover private features.The framework aims to achieve:1)Protect DO data privacy.2)Protect MO model parameters.3)Defend against URVFL-style reconstruction attacks.

2.4 Additive Secret Sharing

We use additive secret sharing on the ring $\mathbb{Z}_t(t = 2^\ell)$. For the secret value x $\in$ $\mathbb{Z}_t$, if it is shared between the MO and the DO, the active party holds $< x >_0$, and the passive party holds $< x >_1$, satisfying $x = < x >_0 + < x >_1 \, mod \, t$. We represent x in the two parties' sharing as $< x >$.

2.5 Vertical Federated Learning and Neural Network Training

Vertical Federated Learning. Consider an active party ID_0 and n passive parties $ID_1, \cdots, ID_n$. Each party holds different characteristics of the same sample, and the active party also holds labels. Each party has its own bottom model $f_n(n = 0, \cdots, n)$, the active party additionally possesses the top model f_{top}. During training, the passive party computes the local embedding $h_n = f_n(x_n)$ and sends it to the active party, the active party computes its own embedding $h_0 = f_0(x_0)$, concatenates them to obtain $\mathrm{h} = [\mathrm{h}_0, \mathrm{h}_1, \cdots, \mathrm{h}_\mathrm{N}]$, and obtains the prediction and calculates the loss through the top model. Subsequently, the active party returns the gradient $\nabla \mathbf{h}_\mathrm{n}$ to each passive party for updating their bottom models.

Neural network layers. Linear layers can be represented as $\mathbf{y} = \mathbf{W} \circ \mathbf{x} + \mathbf{b}$, where is the linear operator. Nonlinear layers (such as ReLU, pooling) operate element-wise. Backpropagation calculates the parameter gradients ∇_W and ∇_b using the chain rule. In vertical federated learning, these gradients need to be securely distributed to all parties.

3 The details of the proposed TMI-VFL

In this section, we introduce the proposed TMI-VFL secure training framework. As shown in Fig. 2, this framework constructs a ciphertext computation channel based on threshold multi-identity fully homomorphic encryption during the vertical federated learning training process, ensuring that the model ' s intermediate representations and gradients remain in the ciphertext domain at all times, thereby preventing the leakage of sensitive information.The system consists of one MO and multiple DO. Each DO holds a distinct subset of features from the same dataset and participates in joint training without sharing raw data. The MO is responsible for maintaining the model architecture and parameters. Through a secure computation protocol, all participants can perform feature combination, model forward

computation, and gradient updates in the ciphertext domain, thereby achieving multi-party collaborative model training while ensuring data privacy.

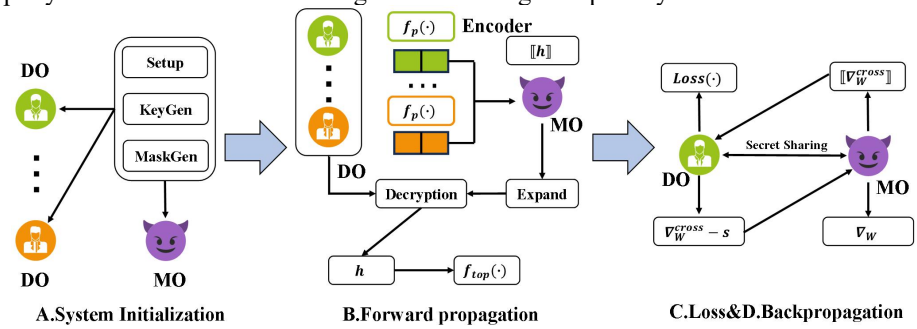

Fig. 2.TMI-VFL Defense Flowchart

A. System Initialization

In the secure training framework proposed in this paper, participants collaboratively train a neural network model without sharing raw data. Each DO holds a different subset of features from the same batch of samples, while the MO is responsible for maintaining the model structure and trainable parameters.

Let the set of participants be $\mathrm{I} = \{\mathrm{ID}_0, \mathrm{ID}_1, \cdots, \mathrm{ID}_\mathrm{n}\}$, where ID_0 denotes the MO, and$ID_i(1 \le i \le n)$ denotes thei th DO.All participants share the same set of sample indices, but features are partitioned among participants by column. During training, the neural network consists of a series of layer functions$\mathrm{f}_1, \mathrm{f}_2 \cdots, \mathrm{f}_\ell$, whereℓ denotes the number of network layers. For an input batch X, let $\mathrm{X}_0 = \mathrm{X}$.Then the forward computation of thei th layer can be expressed as $\mathrm{X}_\mathrm{i} = \mathrm{f}_\mathrm{i}(\mathrm{X}_{\mathrm{i}-1})$, $\mathrm{i} = 1, \cdots, \ell$, with the final output being $\mathrm{Y} = \mathrm{X}_\ell$. To prevent leakage of intermediate representations, all intermediate variables are protected using secret sharing throughout the training process.

(1) Initialization

Setup$(1^\lambda, 1^l, 1^d)$ [3] let λ be the security parameter, l the maximum depth of the computational circuit, and d the maximum depth of the layers. By selecting $\chi = \chi(\lambda, \mathrm{L})$, we obtain a bounded error distribution B_χ. Choose n, m, and q as parameters for the standard error learning problem:

$$m = \omega(nlbq) = [(k+1)n + 1]\ell_q, \ell_q = \lceil lbq \rceil$$

Using the GenBasis($1^n, 1^m, q$) [3] algorithm, we obtain a random uniform matrix $\boldsymbol{A_0} \in \mathbb{Z}_q^{m \times n}$ and a set of short bases $\boldsymbol{S_0} \in \mathbb{Z}^{m \times m}$ for the corresponding lattice. For each $(1, \mathrm{b}) \in [\mathrm{k}] \times \{0,1\}$, we obtain a uniformly random matrix $\boldsymbol{A_{i,b}} \in \mathbb{Z}_q^{n \times m}$ and randomly select $\boldsymbol{z} \leftarrow \mathbb{Z}_q^m$ output system parameters $pp = (n, m, q, \chi, \boldsymbol{B_\chi})$, $mpk = (\boldsymbol{A_0}, \{\boldsymbol{A_{i,b}}\}, \boldsymbol{z}), msk = \boldsymbol{S_0}$. All algorithms use pp as the input by default.

(2) Assign identity tags

Assign identity ID_0 to each MO, assign identity ID_i to each DO, and define the identity set $I = \{ID_0, ID_1, \cdots, ID_n\}$.

(3) Generate a complete private key for each identity

For each ID ∈ I, use the KeyGen(msk,ID) [20] algorithm , where

$$\mathbf{A_{ID}} = \mathbf{A_0}||\mathbf{A_{1,id_1}}||\mathbf{A_{2,id_2}}\cdots||\mathbf{A_{k,id_k}} \in \mathbb{Z}_q^{m\times(k+1)n}, \mathbf{A_{i,id_i}} = \mathbf{A_{i,b}} \in \mathbb{Z}_q^{n\times m}$$

the ExtBasis($\mathbf{S_0}, \mathbf{A_{ID}}, \boldsymbol{z}$) [21] algorithm is used, with the output $\boldsymbol{t_{ID}} \in \{0,1\}^{(k+1)n}$ satisfying $\mathbf{A_{ID}} \cdot \mathbf{t_{ID}} = z \bmod q$. Let the private key be $\boldsymbol{sk_{ID}} = \boldsymbol{s} = (1, -\boldsymbol{t_{ID}^T}) \in \mathbb{Z}_q^{(k+1)n+1}$, then $\mathbf{s} \cdot (\boldsymbol{A_{ID}^{'}})^{\mathrm{T}} = 0 \bmod q \in \mathbb{Z}_q^m$ holds.

(4) Distribute key material and generate a random mask

Here pp, mpk, I, is the public data, and each participant obtains the private key shares $sk_{ID_i}\ for\ ID \in I$ for all identities. The MO collaborates with all DO to pre-generate a batch of random masks, jointly generating the random vector $\boldsymbol{r_i}$. Each DO holds a secret share of $\boldsymbol{r_i}$. These masks will be used for embedding masks and gradient masks during training.

B. Forward Propagation

The forward propagation phase preserves the inherent parallelism of VFL. Passive clients return ciphertext results immediately after completing local embedding computations, without needing to interactively wait for the MO. Therefore, the forward computations of all DO proceed in parallel. Each DO encrypts its own embedding using a public key; the MO performs homomorphic computation; and then the MO collaborates with the DO to perform threshold decryption.

(1) DO performs local embedding computation and encrypts the shared data using the public key pk.

Each passive party DO computes the local embedding $h_p = f_p(x_p)$ based on its feature x_p. To prevent the plaintext embedding from being leaked, DO ecretly shares with $h_p = < h_p >_{MO} + < h_p >_{DO}$, where $< h_p >_{MO} = \boldsymbol{r_i}$ is a random mask, and $< h_p >_{DO} = h_p - \boldsymbol{r_i}$. DO retains $< h_p >_{DO}$, encrypts it with the public key pk, and sends it to the MO.

$$[\![< h_p >_{DO}]\!] = (\boldsymbol{A_{ID}^{'}})^{\mathrm{T}} \cdot \mathrm{R} + \mu \mathrm{G}_{(k+1)n+1} + \mathrm{E} \quad (1)$$

R is a randomly selected matrix $\mathrm{R} \in \{0,1\}^{m\times m}$,and let μ be the plaintext embedding bits to be encrypted,$\mathrm{E} \leftarrow \chi^{[(k+1)n+1]\times m}$

(2) MO computes the embedding locally and aggregates it.

MO applies $h_a = f_a(x_a)$ to its own feature x_a and encrypts its embedding using its public key to obtain $[\![h_a]\!]$. MO collects ciphertext shares sent by different DO and aggregates the ciphertext based on linear homomorphism to obtain $[\![h_p]\!] = [\![< h_p >_{DO}]\!] + < h_p >_{MO}$. Subsequently, MO concatenates its own embeddings to construct the complete input $[\![h]\!] = [[\![h_a]\!], [\![h_p]\!]_1, \cdots, [\![h_p]\!]_n]$. The DO embedding remain in ciphertext form on the MO side, and the plaintext is not exposed.

(3) MO performs multi-identity ciphertext expansion

Given the identity set and the initial ciphertext as inputs, the extended ciphertext $\widehat{[\![h]\!]} \in \mathbb{Z}_q^{k[(d+1)n+1]\times km}$ is obtained. The specific process is as follows:

① For any $j \in [k] \setminus i$, run the $\boldsymbol{X_j} \leftarrow$Extened(μ, mpk, ID, $ID^{'}$) algorithm[3]:

Extened(μ, mpk, ID, $ID^{'}$)

Let $\mathrm{X} = \mathrm{Lcomb}(\mathrm{U}, \mathbf{t_{ID'}^T} \cdot (\mathbf{A_{ID}^T} - \mathbf{A_{ID'}^T}))$,thrn:

$$\boldsymbol{s}^{'}\boldsymbol{C} = \boldsymbol{s}^{'}[(\boldsymbol{A}_{\boldsymbol{ID}}^{'})^{T} \cdot R + \mu \mathrm{G} + \mathrm{E}] = (\boldsymbol{t}_{\boldsymbol{ID}^{'}}^{\boldsymbol{T}}\boldsymbol{A}_{\boldsymbol{ID}^{'}}^{\boldsymbol{T}} - \boldsymbol{t}_{\boldsymbol{ID}^{'}}^{\boldsymbol{T}}\boldsymbol{A}_{\boldsymbol{ID}}^{\boldsymbol{T}})R + \mu \boldsymbol{s}^{''}G + \boldsymbol{s}^{'}E \quad (2)$$

From the expression in Lcomb , we obtain:

$$\mathbf{s}\mathrm{X} = \boldsymbol{t}_{\boldsymbol{ID}^{'}}^{\boldsymbol{T}}(\boldsymbol{A}_{\boldsymbol{ID}}^{\boldsymbol{T}} - \boldsymbol{A}_{\boldsymbol{ID}^{'}}^{\boldsymbol{T}})R + e_x \quad (3)$$

Adding the two equations above yields:

$$\mathbf{s}\mathrm{X} + \boldsymbol{s}^{'}\boldsymbol{C} = \mu \boldsymbol{s}^{'}G + \boldsymbol{s}^{'}E + e_x \quad (4)$$

② The extended ciphertext can be decomposed into k^2 parts, each of which is a matrix of the form $[\![\widehat{h}]\!][a,b] \in \mathbb{Z}_{\mathrm{q}}^{[(\mathrm{d}+1)\mathrm{n}+1]\times \mathrm{m}}$, defined as follows:

$$[\![\widehat{h}]\!][a,b] = \begin{cases} C, & a = b; \\ \boldsymbol{X_j}, & a = i \neq j, b = j; \\ 0^{[(\mathrm{d}+1)\mathrm{n}+1]\times \mathrm{m}}, & \text{else.} \end{cases} \quad (5)$$

For the i user with identity ID_i and initial ciphertext $[\![h]\!] \in \mathbb{Z}_{\mathrm{q}}^{[(\mathrm{d}+1)\mathrm{n}+1]\times \mathrm{m}}$, the expanded ciphertext is:

$$[\![\widehat{h}]\!]_i = \begin{bmatrix} C & 0 & \cdots & 0 & \cdots & 0 \\ 0 & C & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ X_1 & X_2 & \cdots & C & \cdots & X_k \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & C \end{bmatrix} \quad (6)$$

where X_j is located at row i, column $j \in [k]$.

Let $\hat{s} = (s_{ID_1}, s_{ID_2}, \cdots s_{ID_k})$ be calculated as follows:

$$\hat{s}[\![\widehat{h}]\!] = (\mu s_1 G + \hat{e}_1, \mu s_2 G + \hat{e}_2, \cdots, \mu s_k G + \hat{e}_k = \mu \hat{s}\widehat{G}_k + (\hat{e}_1, \hat{e}_2, \cdots \hat{e}_k) \quad (7)$$

This expression has the same form as $\mathrm{s} \cdot \mathrm{G} = \mu \mathrm{s}\mathrm{G}_{(\mathrm{k}+1)\mathrm{n}+1} + \mathrm{s}\mathrm{E}$ and satisfies the correctness of encryption and decryption.

(4) Threshold Decryption

The MO sends $[\![h]\!]$ to all DO. Each DO partially decrypts $[\![h]\!]$ using its own private key share $p_i = Dec([\![h]\!], sk_i)$, defining the vector $\widehat{w} = (0, \cdots, 0, \left\lceil \frac{q}{2} \right\rceil, 0, \cdots 0)^T \in \mathbb{Z}_q^{[(k+1)n+1]}$. For the i user $\left\lceil \frac{q}{2} \right\rceil$ located in row $(i-1)[d+1)n+1]+1$, where $[\![\widehat{h}]\!]$ is the initial ciphertext, we have $\mu^{'} = sk_i\widehat{C}\widehat{G}^{-1}(\widehat{w}) = \mu \cdot \left\lceil \frac{q}{2} \right\rceil + e^{'}$. The partially decrypted results are then sent to the MO, which aggregates all DO decryption results to reconstruct the aggregated plaintext result h.

(5) The MO performs the forward pass of the linear layer in the ciphertext domain.

MO calculates the linear output of the first layer of the top model based on the ciphertext embedding: $u = W_{top} \cdot h + b, W_{top}$, where b represent the weights and gradients of the top model, respectively. To prevent gradient leakage during the backward pass, MO generates a random mask vector s and constructs $< u >_{DO} = W_{top} \cdot h - s$. MO retains $< u >_{DO}$ and does not send data to the DO during the forward pass; instead, it sends $< u >_{DO}$ to the DO for decryption during serialization, and additionally $< u >_{MO} = s + b$.

At this point, the embeddings of all DO and MO, as well as the gradients, have not been leaked in plaintext during the forward pass.

C. Loss Calculation

The loss calculation phase involves labels and gradients. To prevent the MO or DO from unilaterally recovering sensitive information, this phase is executed sequentially via MPC. That is, in each training step, only the MO collaborates with a single DO; the forward propagation phase DO will be abbreviated as DO in subsequent steps.

(1) Data Sharing

In this phase, the MO collaborates with only one DO per training step. First, the data not yet shared during the forward propagation phase is transmitted to the DO ($< u >_{DO}$).

(2) Reconstructing the Output Results

The MO sends the final propagation output $< u >_{MO}$ to the DO. The DO reconstructs the output u of the linear layer from the forward pass.

$$u =< u >_{DO} + < u >_{MO} = W_{top} \cdot h - s + s + b = W_{top} \cdot h + b \tag{8}$$

(3) MPC-based gradient recovery and loss calculation

The DO calculates the loss function $L(u, y)$ based on u and the label y, and obtains the output gradient ∇u as shown in Eq.(9).

$$\mathrm{L} = \frac{1}{|\mathcal{B}|} \sum \mathrm{l(u, y)} \tag{9}$$

Where $\mathcal{B}$ is the batch of data selected for the training phase.

D. Backpropagation

Backpropagation is performed serially by the MO in collaboration with each DO. This design prevents contamination between DO while ensuring that each DO gradients are calculated solely based on its own data.

(1) DO share output gradients

To facilitate subsequent gradient calculations, DO secretly share their output gradients $\nabla \mathrm{u} = < \nabla \mathrm{u} >_{MO} + < \nabla \mathrm{u} >_{DO}$ .The DO also sends the homomorphically encrypted output gradient ($< \nabla \mathrm{u} >_{DO}$) to the MO:

(2) MO Shares the Full Embedding

To prevent the leakage of DO $\nabla \mathrm{u}$, the MO must share its own h here $h = < h >_{MO} + < h >_{DO}$

(3) MO and DO Collaboratively Perform MPC Multiplication

MO uses its own $< \nabla \mathrm{u} >_{MO}$ and the $[\![< \nabla \mathrm{u} >_{DO}]\!]$ sent by DO to compute the cross-product $[\![\nabla_W^{cross}]\!]$ via homomorphic multiplication:

$$[\![\nabla_W^{cross}]\!] = < \nabla \mathrm{u} >_{MO} \odot < h >_{DO} + [\![< \nabla \mathrm{u} >_{DO}]\!] \odot < h >_{MO} \tag{10}$$

(4) MO protects the gradient with a random mask

MO randomly selects a vector s, constructs the masked gradient $[\![\nabla_W^{cross}]\!] - s \rightarrow [\![\nabla_W^{cross} - s]\!]$, and sends this ciphertext to DO.

(5) The DO decrypts the masked cross-gradient and reassembles the gradient

DO decrypts to obtain $\nabla_W^{cross} - s$, combines it with its own $< \nabla \mathrm{u} >_{DO}$ and $< h >_{DO}$ to obtain $\widetilde{\nabla w} = \nabla_W^{cross} - s + < h >_{DO} \odot < \nabla \mathrm{u} >_{DO}$. To prevent the

embedding from being inferred from the gradient, DO adds a small noise vector e to $\widetilde{\nabla w}$, and then sends $\widetilde{\nabla w}$ to MO.

(6) MO computes ∇w, and DO and MO update their respective models

$$\nabla \mathrm{w} = \widetilde{\nabla w} + s + < h >_{MO} \odot < \nabla \mathrm{u} >_{MO} = \nabla u \odot h \tag{11}$$

4 Experiments

This section presents an experimental evaluation of the proposed TMI-VFL framework. We first describe the experimental setup, including the dataset, model architecture, comparison methods, and evaluation metrics. Subsequently, we report and analyze the defense performance, conduct ablation experiments on key parameters, and finally discuss the advantages and limitations of the proposed approach.

4.1 Experimental Setup

Dataset and Task: We conduct experiments on the CIFAR-10 dataset, which contains 60,000 color images of size 32×32 across 10 classes, with 50,000 for training and 10,000 for testing.To simulate a VFL scenario, each image is split into two parts along the channel dimension. The MO holds one part along with the labels, while DO holds the remaining part.Following the URVFL setting, 10% of the training data is randomly selected as the attacker's auxiliary dataset, with no overlap with the training set.

Model Architecture: Both active and passive parties adopt the same backbone architecture. For image data, we use a simplified ResNet-34 (ClientNet_small), consisting of three residual blocks followed by global average pooling, producing a 64-dimensional feature vector. The server model (SimpleServer) is a two-layer fully connected network that concatenates features from both parties for classification.The attacker's encoder, decoder, and shadow model follow the URVFL setup, based on the first three layers of ResNet-34, generating feature maps of size 16×16×64.For tabular datasets, all components are implemented using multi-layer perceptrons (MLPs).

Comparison Methods: We evaluate defense performance against feature reconstruction attacks using the following baselines:Vanilla VFL,Standard training without any privacy protection, serving as a baseline to evaluate vulnerability to reconstruction attacks.Perturbation-based methods: Gaussian noise with different magnitudes ($\varepsilon = 0.1, 1.0, 2.0$) is added to the embeddings of the passive party to reduce feature recoverability.

Other methods in TABLE I (e.g., attack or detection-based approaches) are excluded, as they do not provide direct protection during training. In addition, many cryptographic approaches assume semi-honest settings or focus on efficiency, which differs from our focus on defending against malicious reconstruction attacks.In contrast, the proposed TMI-VFL protects intermediate representations and gradients

via cryptographic mechanisms, enabling effective defense against URVFL-style attacks.

Evaluation Metrics: 1) Reconstruction Error: Using the attacker's (URVFL) decoder to reconstruct the test set, we calculate the Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure(SSIM) between the reconstructed image and the original image. Under effective defense, the MSE should increase significantly, while the PSNR and SSIM should decrease significantly, indicating that the attacker cannot recover valid visual information.2) Classification accuracy: The model's classification accuracy on the test set, which measures performance on the primary task.

Implementation Details: All experiments were conducted on Ubuntu 20.04 using the PyTorch framework. The homomorphic encryption component implements the CKKS scheme based on TenSEAL, with parameters set to polynomial order 8192, modular chains [60,40,40,60], and scaling factors .Training employed the Adam optimizer with a learning rate of 1e-4 and a batch size of 128. Training for the no-defense and masked-defense scenarios consistes of 25 epochs, while TMI-VFL training also ran for 25 epochs. The pre-training seed for the attack model was fixed at 141.

4.2 Comparison of Defense Effectiveness

TABLE II presents the comparison results of the attacker's reconstruction capability and the model's classification performance under different defense mechanisms. Without defense, the URVFL attack can effectively recover the passive party's features, with an MSE of 0.0349, a PSNR of 20.60 dB, and an SSIM of 0.561, indicating that the attacker can reconstruct the original input quite well.

After applying masking defense, the attacker's reconstruction capability decreases significantly as the noise amplitude increases. When $\varepsilon = 1.0$, MSE rises to 0.3026, PSNR drops to 11.22 dB, and SSIM falls to 0.0469;when $\varepsilon = 2.0$, MSE reaches 0.3335, and PSNR and SSIM drop to 10.80 dB and 0.0119, respectively. However, at the same time, the model's classification accuracy decreases from 56.0% to 43.41%, indicating that noise perturbations affect model performance.

Table 2. Comparison of Reconstruction Metrics under Attack and Defense

Defense mechanisms	Noise level ε	MSE	PSNR (dB)	SSIM	Classification accuracy (%)
No defense (baseline)	–	0.0349	20.60	0.561	56.0
Mask Defense	0.1	0.0321	20.97	0.577	49.25
Mask Defense	1.0	0.3026	11.22	0.0469	46.50
Mask Defense	2.0	0.3335	10.80	0.0119	43.41
TMI-VFL	**–**	**≥0.3892**	**≤11.00**	**≤0.05**	**55.56**

In contrast, the TMI-VFL proposed in this paper still achieves a classification accuracy of 55.56% under ciphertext-domain training conditions, which is essentially consistent with the unprotected baseline (56.0%).Results from multiple experiments show that under TMI-VFL conditions, the attacker's reconstruction error is no less than 0.3892, PSNR is no higher than 11.00 dB, and SSIM is no higher than 0.05. This indicates that the attacker is unable to recover valid information, thereby effectively blocking URVFL attacks.

4.3 Analysis of the Training Process

To further analyze the impact of the proposed defense mechanism during training, Fig. 3 presents both the reconstruction error and the classification accuracy over training epochs.

As shown in Fig. 3(a), under the unprotected baseline, the attacker reconstruction error gradually decreases and stabilizes, indicating that the attack model progressively learns effective feature recovery. In contrast, under the proposed defense, the reconstruction error increases throughout training and converges to a significantly higher level, demonstrating that the attacker is unable to reconstruct the original input features.

Fig. 3(b) illustrates the corresponding classification accuracy (normalized to the no-defense baseline). It can be observed that although minor fluctuations occur during training, the model accuracy steadily improves and eventually converges close to the baseline performance. This indicates that the proposed method introduces only negligible performance degradation while effectively enhancing privacy protection.

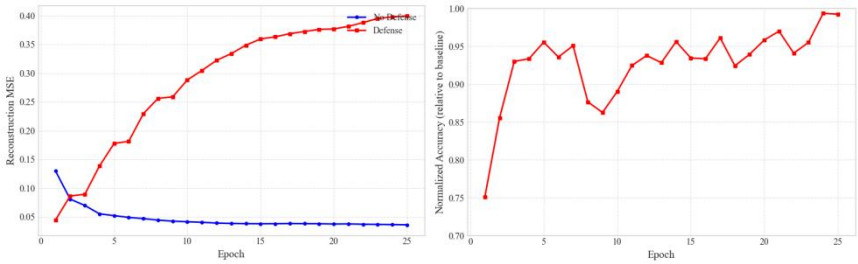

Fig. 3. Training dynamics under defense: (a) Reconstruction error over training epochs; (b) Classification accuracy over training epochs.

4.4 Ablation Studies

Impact of Noise Magnitude: As shown in TABLE II, when $\varepsilon = 0.1$, the reconstruction error is even slightly lower than that of the unprotected model, suggesting that minor perturbations may actually aid the attack due to regularization effects; when $\varepsilon \geqslant 1.0$, the MSE increases significantly, indicating that the defense mechanism is effective. However, an excessively high ε leads to unstable model training, as evidenced by the fluctuations in SSIM.

Encryption Overhead: We measured the average training time per epoch for TMI-VFL and the unprotected model. On an NVIDIA RTX 3050 GPU, unprotected training takes approximately 15 seconds per epoch, while TMI-VFL took about 50 seconds per epoch due to encryption and decryption operations. Although the computational burden increases, this trade-off provides cryptographically strong privacy guarantees, which is acceptable for security-sensitive scenarios.

5 Conclusion

This paper investigates the issue of data reconstruction attacks in vertical federated learning, focusing on the privacy risks posed by URVFL attacks in exploiting intermediate representations to recover passive party features. To address this issue, we propose TMI-VFL, a secure training framework based on threshold multi-identity fully homomorphic encryption.This framework establishes a ciphertext computation channel during training, ensuring that embeddings, gradients, and intermediate activation values remain in ciphertext form at all times, thereby fundamentally preventing attackers from reconstructing features using intermediate information.

Experimental results show that, in the absence of defenses, URVFL attacks can recover passive party data with low reconstruction error. While perturbation-based defenses can reduce attack success rates to some extent, they significantly degrade model performance. In contrast, TMI-VFL significantly increases attack reconstruction error while maintaining model classification accuracy close to the baseline, effectively reducing the attacker's ability to recover original features. The results demonstrate that the proposed method achieves a good balance between privacy protection and model utility.

Future work will focus on developing more efficient homomorphic computation optimization methods and exploring the application of this framework in larger-scale federated learning scenarios.

Acknowledgements. This work is supported by the National Science Foundation of China under Grants U22B2027, 62503362.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Yao D, Li S, Gong X, et al. URVFL: Undetectable data reconstruction attack on vertical federated learning[J]. arXiv preprint arXiv:2404.19582, 2024.

[2] Liu X, Liu Z, Li Q, et al. Pencil: Private and extensible collaborative learning without the non-colluding assumption[J]. arXiv preprint arXiv:2403.11166, 2024.

[3] Tu G S, Yang X Y, Zhou T P. Efficient identitybased multi-identity fully homomorphic encryption scheme[J]. Journal of Computer Applications, 2019, 39(3): 750-755.
[4] Qian B, Xie Y, Li Y, et al. Tree-based models for vertical federated learning: A survey[J]. ACM Computing Surveys, 2025, 57(9): 1-30.
[5] Anees A, Field M, Holloway L. Development of federated learning neural networks with combined horizontal and vertical data partitioning[J]. Applied Soft Computing, 2026: 114734.
[6] Hu X B, Wang Y P, Zhou N R. AISS-VFL: Efficient Adaptive Importance-Aware Sample Selection for Vertical Federated Learning over Distributed Labels[J]. Expert Systems with Applications, 2026: 132011.
[7] Chen T, Jin X, Sun Y, et al. Vafl: a method of vertical asynchronous federated learning[J]. arXiv preprint arXiv:2007.06081, 2020.
[8] Zhang C, Xie Y, Bai H, et al. A survey on federated learning[J]. Knowledge-Based Systems, 2021, 216: 106775.
[9] Tanuwidjaja H C, Choi R, Baek S, et al. Privacy-preserving deep learning on machine learning as a service—a comprehensive survey[J]. Ieee Access, 2020, 8: 167425-167447.
[10] Jiang X, Zhou X, Grossklags J. Comprehensive analysis of privacy leakage in vertical federated learning during prediction[J]. Proceedings on privacy enhancing technologies, 2022.
[11] Vu M N, Tre'R J, Alharbi R, et al. Active data reconstruction attacks in vertical federated learning[C]//2023 IEEE International Conference on Big Data (BigData). IEEE, 2023: 1374-1379.
[12] "Complete guide to GDPR compliance,"Feb 2020, accessed: Nov.2023. [Online]. Available: https://gdpr.eu/
[13] Abuadbba S, Kim K, Kim M, et al. Can we use split learning on 1d cnn models for privacy preserving training?[C]//Proceedings of the 15th ACM Asia conference on computer and communications security. 2020: 305-318.
[14] Luo X, Wu Y, Xiao X, et al. Feature inference attack on model predictions in vertical federated learning[C]//2021 IEEE 37th international conference on data engineering (ICDE). IEEE, 2021: 181-192.
[15] Tian H, Zeng C, Ren Z, et al. Sphinx: Enabling privacy-preserving online learning over the cloud[C]//2022 IEEE Symposium on Security and Privacy (SP). IEEE, 2022: 2487-2501.
[16] Ng L K L, Chow S S M. SoK: Cryptographic neural-network computation[C]//2023 IEEE Symposium on Security and Privacy (SP). IEEE, 2023: 497-514.
[17] Pasquini D, Ateniese G, Bernaschi M. Unleashing the tiger: Inference attacks on split learning[C]//Proceedings of the 2021 ACM SIGSAC conference on computer and communications security. 2021: 2113-2129.
[18] Mireshghallah F, Taram M, Jalali A, et al. Not all features are equal: Discovering essential features for preserving prediction privacy[C]//Proceedings of the Web Conference 2021. 2021: 669-680.
[19] Vepakomma P, Singh A, Gupta O, et al. NoPeek: Information leakage reduction to share activations in distributed deep learning[C]//2020 International Conference on Data Mining Workshops (ICDMW). IEEE, 2020: 933-942.

[20] Erdogan E, Küpçü A, Cicek A E. Splitguard: Detecting and mitigating training-hijacking attacks in split learning[C]//Proceedings of the 21st Workshop on Privacy in the Electronic Society. 2022: 125-137.