# MVFLS: Multi-participant Vertical Federated Learning based on Secret Sharing

**Haoran Shi[1], Yali Jiang[1], Han Yu[3,4], Yonghui Xu[2] *, Lizhen Cui[1,2] †**

[1]School of Software, Shandong University, China
[2]Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR), Shandong University, China
[3]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[4]Joint NTU-WeBank Research Centre on FinTech, NTU, Singapore
202035273@mail.sdu.edu.cn, xu.yonghui@hotmail.com, clz@sdu.edu.cn

## Abstract

Vertical Federated Learning (VFL) is a distributed machine learning method that combines all features from different collaborative train models for Federated Learning (FL) clients with data involving different feature spaces. Currently, VFL mainly uses homomorphic encryption (HE) to preserve privacy. However, HE incurs high computation and communication costs among FL participants, which increase exponentially with the number of participants. Thus, existing VFL methods are unsuitable for multi-party scenarios. Moreover, in VFL scenarios, it is difficult to establish an FL server that can be trusted by all participants. To solve these problems, we propose an efficient Multi-participant Vertical Federated Learning approach based on Secret Sharing (MVFLS). On the one hand, it uses secret sharing instead of homomorphic encryption, which effectively reduces the computational cost. On the other hand, benefiting from secret sharing, it enables VFL to be performed in multi-participant scenarios while eliminating the dependence on having an FL server. This can further reduce the risk of data leakage on the server side and effectively protect the security of participants. Experimental results on real-world and synthetic data sets show that MVFLS can significantly reduce computational cost and improve model accuracy compared with stat-of-the-art federated learning methods.

## Introduction

As a type of federated learning (FL), vertical federated learning (VFL) is principally applicable for data owners who have a slightly varied user space with different sets of features. Fundamentally, VFL is the process of aggregating these different features from multiple companies or organizations. For a specific example (Figure 1), *bank A* wants to establish a model to predict customer credit. *E-commerce B* is in the same city as $A$, so their customer bases are similar. *A* has the customers' financial information and *B* has the online consumption information of the customers. That is, *A* and *B* have different data features of the customers. Then, *B* can make use of its information to help *A* build a VFL model, which can accurately evaluate the customers' credit rating. If $A$ and $B$ jointly establish a VFL model to integrate

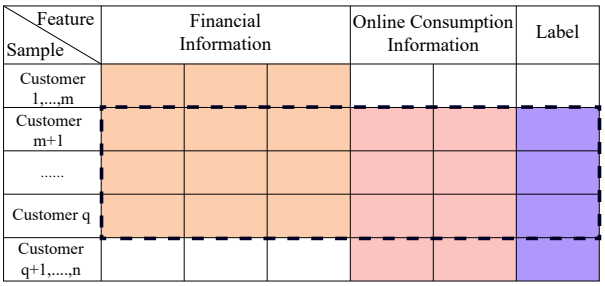



Figure 1: Illustration of Vertical Federated Learning between a *Bank A* and an *E-commerce B*

their feature data of customs without data leakage, so that the credit rating can be predicted more accurately. During this process, the model should protect the privacy of participants' local information. Then, one of the challenges is data security, which means the data of any participant cannot be obtained by others.

Data security is one of the critical properties of vertical federated learning. The majority of VFL models use the Homomorphic Encryption (HE) methods (Paillier 1999) to encrypt transmitted data that contains samples' features. HE performs simple calculations (like plus, multiplication) on the encrypted data of all parties and the decrypted result is consistent with the result of directly calculating the data. As a classical VFL model, Hardy (Hardy et al. 2017) introduces the Paillier homomorphic method (Paillier 1999) in the VFL. In this model, participants can transmit encrypted intermediate results to effectively hide the local data, when they aggregate the features from all parties. However, the weakness of HE is the high computation cost, and it is difficult to apply this security method to multiple parties. As the number of participants in vertical federated learning increases, the interaction becomes more complex and the communication and computation costs can be higher. In addition, homomorphic encryption only can perform a simple calculation on encrypted data and other calculations can only be replaced by approximate formulas, resulting in lower accuracy than training results of traditional machine learning.

Besides the shortcomings of HE, VFL models also have the risk of information leakage caused by the server. The existing VFL models require an authentication server to coor-

*Corresponding author
†Corresponding author

dinate and transmit encrypted information, but there is a risk of information leakage during this process. At the same time, because the model structure is complex, all participants have to interact with the coordinator, which can incur high communication costs.

To solve these problems, we propose an efficient Multi-participant Vertical Federated Learning based on Secret Sharing (MVFLS). First of all, secret sharing is used to replace homomorphic encryption in our improved multi-party VFL, which greatly improves communication and computation efficiency. Secondly, the model we proposed can calculate the accurate gradient and loss function formula without approximation. Thus, compared with the existing VFL, the accuracy of the model can be improved. Thirdly, our method relies on one participant with label information to coordinate the calculations of all parties without a dedicated server, reducing the number of communication rounds and preventing the third party from maliciously sending wrong information.

The rest of this paper is organized as follows. Next section introduces previous related works. After the related work, we describes the details of our model. Then we conduct security analysis of the proposed method. At last, we report the experimental results.

## Related Work

The research work related to this study mainly includes three aspects: federated learning based on homomorphic encryption, vertical federated learning, and multi-party secure computing. In 1978, Rivest, Adleman, and Dertouzos proposed the concept of HE ( homomorphic encryption ) (Rivest et al. 1978). Since then, HE has been a key problem in the field of cryptography research. Homomorphic encryption performs addition and multiplication operations on the ciphertext and the decrypted result is equivalent to the consequence of performing operations on the plaintext. Homomorphic encryption does not reveal the original information during the process of calculation. Existing homomorphic encryption algorithms mainly can be divided into two categories: semi-homomorphic encryption and fully homomorphic encryption. Semi-homomorphic encryption mainly includes multiplicative homomorphic encryption, like RSA algorithm and ElGamal algorithm (ElGamal 1985), and additive homomorphic encryption, like Paillier algorithm (Paillier 1999). Then, the first fully homomorphic encryption scheme is proposed by the Gentry (Gentry 2009), and other full homomorphic encryption algorithms mainly include the BGV scheme (Brakerski, Gentry, and Vaikuntanathan 2014) GSW scheme(Gentry, Sahai, and Waters 2013), and the CKKS scheme (Cheon et al. 2017) that supports approximate calculation of floating-point numbers, and so on. Above several types of homomorphic encryption are introduced into vertical federated learning, which can protect privacy and perform various operations on encrypted data.

There are many encrypted methods in VFL, i.e., HE. because privacy leakage exists in the process of vertical federated learning. Specifically, participants without label information cannot build their machine learning models, which prevents them from uploading models or gradient parameters. As a result, they have to transmit intermediate results containing local data, which brings the risk of privacy leakage. Therefore, some privacy protection methods have been needed to ensure data security in vertical federated learning. (Yu, Vaidya, and Jiang 2006) proposed a privacy-preserving SVM (Support Vector Machine) classification method based on vertically partitioned data. (Hardy et al. 2017) focuses on vertical federated learning for linear and logistic regression models based on additive homomorphism and applies more machine learning models to the VFL framework. Then, (Gilad-Bachrach et al. 2016) adopted fully homomorphic encryption to protect privacy in neural network training. (Qiang et al. 2019) reduces the number of interactions between parties in each round, thus simplifying the training steps. The communication and calculation costs of this homomorphic encryption (HE) method are high, and polynomial approximation is required to evaluate the nonlinear function. These methods may lead to low accuracy in multi-party scenarios, so they are not suitable for multiple parties. Thus, (Xu, Yuan, and Xintao 2019) proposes differential privacy (DP) as a security guarantee, which can extend vertical federated learning to multi-party participation. However, DP still requires data to be transmitted to other places, and there is a certain probability of privacy leakage. At the same time, it is necessary to weigh the relationship between accuracy and privacy.

In this paper, secure multi-party computing (SMPC) is introduced into vertical federated learning. This theory mainly focuses on collaborative computing between participants and the protection of private information. SMPC protocol allows multiple parties to perform collaborative calculations, output the calculation results, and ensure that no one party can get any other information except the encrypted results. Secure multi-party computation originated from the millionaire problem raised in the 1980s. Then, SMPC works are divided into several categories: some of the previous works are based on garbled circuit (Yao 1986), which are calculated by simulating complex circuits, but those are suitable for two parties. Part of the research concentrates on secret sharing (Shamir 1979), which mainly appropriately splits the secret, and the secret message can be recovered only if the split shares are combined. Secret sharing technology is a commonly used technology in the field of secure multi-party computing. (Keith et al. 2017) designed a novel secret sharing protocol to safely aggregate data held by a large number of users in Horizontal Federated learning. This protocol can be extended to multi-party scenarios and can help the VFL framework become computationally efficient.

## The Proposed Approach

In order to solve the problems existing in the existing vertical federated learning methods, such as the time-consuming encryption process, and propose a vertical federated learning method suitable for multi-party participation scenarios, this section introduces the proposed Multi-participant Vertical Federated Learning based on Secret Share in detail. First, we introduced the definition of related symbols and the problem description. Then we introduce a secret sharing strategy that saves encryption time to replace the traditional homomorphic encryption method of vertical federated learning.

## Definition and Problem Statement

Denoted by $D = \{(x_i, y_i | i = 1, ..., N)\}$ $N$ data samples, where $x_i \in R^{d \times 1}$ is distributed among $M$ participants $P\{p_i | i = 1, ..., M\}$. In our vertical federated learning setting, participants $p_M \in P$ holds the label information. So its dataset is denoted as $D_{p_M} = \{x^{p_M}; y\}$, and the data set of participants $p_a$ without label information is represented as $D_{p_a} = \{x^{p_a}\}(p_a \in P)$. $f(\cdot)$ denotes the loss function in different machine learning methods and $\lambda$ is the hyper-parameter. Our goal is to use the data of all parties to establish a joint machine learning model without revealing privacy and it is formulated as

$$\underset{\theta_{p_1}, ... \theta_{p_M}}{argmin} \mathcal{L} = \frac{1}{N} \sum_{p_a \in P} f(\theta_{p_a}; D_{p_a}) + \frac{\lambda}{2} \sum_{p_a \in P} \|\theta_{p_a}\|^2 \quad (1)$$

$\theta_{p_a}$ is the training parameters of $k^{th}$ participants. $f(\cdot)$in Linear and Logistic regression respectively are:

$$f(\theta_{p_a}; D_{p_a}) = \| \sum_{p_a \in P} \theta_{p_a} x^{p_a} - y \|^2$$

$$f(\theta_{p_a}; D_{p_a}) = log(1 + e^{-y \sum \theta_{p_a} x^{p_a}}) - y \sum_{p_a \in P} \theta_{p_a} x^{p_a} \quad (2)$$

To simplify the equation, we set $u^{p_a} = \theta_{p_a} x^{p_a}$. The gradients of Linear and Logistic regression respectively are :

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta_{p_a}} &= 2x^{p_a} \| \sum_{p_a \in P} u^{p_a} - y \| + \lambda \sum_{p_a \in P} \|\theta_{p_a}\| \\ \frac{\partial \mathcal{L}}{\partial \theta_{p_a}} &= x^{p_a} (y - \frac{e^{\sum u^{p_a}}}{1 + e^{\sum u^{p_a}}}) \end{aligned} \quad (3)$$

Set $\eta$ is learning rate, each one updates their parameters:

$$\theta'_{p_a} = \theta_{p_a} - \eta \frac{\partial \mathcal{L}}{\partial \theta_{p_a}} \quad (4)$$

Through the equation 3 and 4, each participant needs to obtain $Y = \sum u^{p_a} - y$ to update its own parameters in the linear regression model. Similarly, in the logistic regression, each participant has to obtain $G = y - \frac{e^{\sum u^{p_a}}}{1 + e^{\sum u^{p_a}}}$ to calculate new parameters. In order to obtain the aggregation data Y or G without data leakage, while ensuring the efficiency of the algorithm, we need to explore more secure data aggregation strategies. We have adopted some secure aggregation mechanisms and conducted some explorations.

## Secure Aggregation based on Secret Sharing

To ensure that private information is not leaked in data aggregation, this section introduces a novel secure aggregation method, in which secret sharing means that a secret *S* is split into several parts, and a part of shares can be reconstructed into *S*. We rely on a secure aggregation method (Keith et al. 2017), which extends the traditional secret sharing method. This section will describe the details of this aggregation method.

Suppose that the server aggregates inputs from $n$ participants $P$, and each participant $p_a \in P$ holds a private local data $u_a$ and transmits the input value $y_a$ to the server. The goal of this method is to calculate the sum of $u_a$ without disclosing local information $u_a$.

If $u_a$ is masked in a specific way and participants transmit masked values to server, $u_a$ can be calculated safely. To be concrete, we assume that each pair of users $(p_a, p_b)$ agrees on one random number $S_{p_a,p_b}$ or $S_{p_b,p_a}$ ($S_{p_a,p_b} = S_{p_b,p_a}$), as a mask to conceal the actual value $u_a$and $u_b$. When a is less than b, $p_a$ computes input $y_a = u_a + S_{p_a,p_b}$ before sending $y_a$ to the server and $p_b$ subtracts the masked value $S_{p_b,p_a}$ from $u_b$. Therefore, if two input values are added, this mask will be canceled and local data will not be exposed. In this way, server can compute the aggregation value Y through computing the sum of all participants' inputs $y_a$, which is equal to the sum of $u_a$ (equation 5).

$$\begin{aligned} y_a &= u_a + \sum_{a<b; p_b \in P} S_{a,b} - \sum_{a>b; p_b \in P} S_{a,b} \\ Y &= \sum_{p_a \in P} y_a = \sum_{p_a \in P} u_a \end{aligned} \quad (5)$$

To generate a common number$S_{p_a,p_b}/p_b, p_a$ between two participants, we adopt Diffie-Hellman key agreement (Diffie and Hellman 1976). Suppose that server defines a group $\mathcal{G}$ of prime order $q$, along with a generator $g$. The next process is how to agree on value $S_{p_a,p_b}$ between $p_a$ and $p_b$. They respectively pick random numbers $sk_a (sk_a < q)$ and $sk_b (sk_b < q)$ as private keys. After that, they separately compute public keys $pk_a = g^{sk_a}$ and $pk_b = g^{sk_b}$. Then, they exchange their public keys. $p_a$ can compute agreement results $S_{p_a,p_b} = pk_b^{sk_a} \ mod \ q$ through others' public key and own private key. In a similar fashion, $p_b$ can calculate $S_{p_b,p_a} = pk_a^{sk_b} \ mod \ q$. equation 2 proves $S_{p_a,p_b} = S_{p_b,p_a}$

$$\begin{aligned} S_{p_a,p_b} &= pk_b^{sk_a} \ mod \ q = (g^{sk_b})^{sk_a} \ mod \ q \\ &= (g^{sk_a})^{sk_b} \ mod \ q = pk_a^{sk_b} \ mod \ q \\ &= S_{p_b,p_a} \end{aligned} \quad (6)$$

A pair of participants can safely negotiate a random number. Furthermore, to reduce communication overhead, we introduce Pseudorandom Generator (PRG). PRG is an algorithm to generate a "random" string. To be more concrete, after receiving a fixed-length seed, a PRG generator outputs a "random" number of length [0,*l*) (*l* can be defined). When any algorithms cannot distinguish this number bought by PRG and the real random number of length [0,*l*), we can judge that the PRG model can ensure the seed security. In our method, $S_{p_a,p_b}$ represents the seed, and this masked value is denoted as $PRG(S_{p_a,p_b})$. If the generator receives the same seed, their outputs are same. $PRG(S_{p_a,p_b})$ replaces $S_{p_a,p_b}$ as the negotiated random value of both parties.

Although the proposed secure aggregation strategy can ensure the privacy of data and the effectiveness of VFL relying on a central server, there is still a problem about how to ensure the privacy of data and the efficiency of encrypted aggregation methods in the serverless VFL method. As an encryption method, secret sharing can be applied to vertical

federated learning, which can help the server calculate the data of all parties.

## Serverless Vertical Federated Learning

To explore the serverless VFL method, this section takes the vertical federated learning of three participants as an example. In this case, there are three participants $p_a, p_b, p_c$ involved in this VFL model training and $p_c$ contains the label information. The goal of our method is that none of the three parties upload local data, but participants can establish a joint model. Thus, the calculation of $Y$ or $G$ is the key to success. The process of calculation is that $p_a$ and $p_b$ adopt secret sharing method to generate their own pair of key $< sk_a, pk_a >$ and they exchange their public key $pk_a, pk_b$ . Then, they can negotiate a random value $PRG(S_{p_a,p_b})$ through Diffie-Hellman key agreement. $p_a$ figures out $u_{p_a}$, then sends masked value $y_{p_a} = u_{p_a} + PRG(S_{p_a,p_b})$ to $p_c$; $p_b$ computes $u_{p_b}$ and transmits encrypted value $y_{p_b} = u_{p_b} - PRG(S_{p_b,p_a})$. After receiving $y_{p_a}$ and $y_{p_b}$, $p_c$ computes the sum of them and its own data $u_{p_C}$. If it is in a linear regression model, $p_c$ calculates the aggregate $Y$ following the equation 7. Otherwise, $p_c$ calculates the aggregation result $G$ for logistic regression in a similar fashion. $p_c$ sends $Y$ or $G$ to others;

$$\begin{aligned} Y &= \sum_{p_a \in p_a, p_b} y_{pk} + u_{p_c} - y \\ &= (u_{p_a} + PRG(S_{p_a,p_b})) + (u_{p_b} - PRG(S_{p_b,p_a})) \\ &+ u_{p_c} - y \\ &= u_{p_a} + u_{p_b} + u_{p_c} - y \end{aligned} \quad (7)$$

All parties update and calculate new parameters $\theta'_{pk}$ according to the equation 4.

In this case, three parties cooperate to establish a machine learning model under a vertical federated learning setting. Moreover, the gradients calculated by each party are the same as the loss they would be received without privacy protection, so the model is lossless. Then, we have explored how to expand the scope of use of this model.

## Multi-participant Vertical Federated Learning

To extend the VFL model to multiple parties scenarios and protect the private information in the multi-party model, we adopt secret sharing as a privacy protection method. Suppose that there are M participants $p_a$ in the multi-party model training, the party $p_M$ holds the label attribute. Calculating the sum of $u_{p_a}$ is the key to this module.

Each pair of participants ($p_a$, $p_b$) without label is asked to negotiate one masking number. Each party without label computes their own encrypted value $y_{p_a}$ (equation 8) and sends it to $p_M$.

$$y_{p_a} = u_{p_a} + \sum_{p_b \in P; p_b \neq p_M} (\sum_{a<b} PRG(S_{p_a,p_b}) - \sum_{a>b} PRG(S_{p_b,p_a})) \quad (8)$$

$p_M$ aggregates $y_{p_a}$ through the equation 9, which is equivalent to the sum of $u_{p_a}$. In linear regression, $p_M$ outputs $Y$ according to equation 5. In terms of logistic regression, $p_M$ calculates and outputs $G$ in the same way.

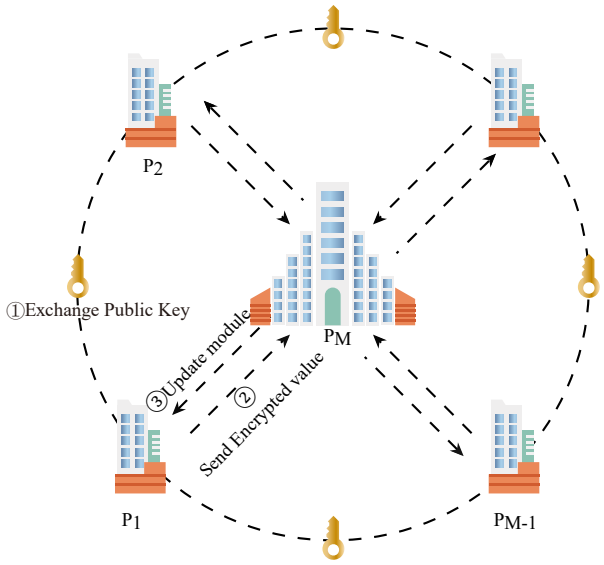



Figure 2: Multi-participant Vertical Federated Learning

$$Y = \sum_{p_a \in P; p_a \neq p_M} y_{p_a} + u_{p_M} - y = \sum_{p_a \in P} u_{p_a} - y \quad (9)$$

This method can achieve vertical federated learning training. However, the efficiency of VFL models depends on the communication and computation cost of privacy methods. This method requires participants without label information to interact with the remaining $M-2$ participants and they need to store others' public key values, so this kind of key agreement protocol will inevitably waste a lot of storage space and time.

Therefore, this paper proposes two improvements to this model. To begin with, participants only negotiate with neighborhood nodes, this is, $p_1$ agrees on parameters with $p_2$ and $p_{M-1}$, and $p_{M-1}$ adopt $PRG_{S_{p_1,p_{M-1}}}$ and $PRG(S_{p_{M-2},p_{M-1}})$ as masked value to hide $u_{p_{M-1}}$.The other $p_a$ only interacts with $p_{a-1}$ and $p_{a+1}$. Every participant without label transmits $y_{p_a}$ (equation 10) to $p_M$. In this way, each participant only needs to calculate one or two encryption values. Thus, this method can deeply improve efficiency, and as the number of participants increases, the advantages of the model become more obvious.

$$y_{p_a} = \begin{cases} u_{p_1} + PRG(S_{p_1,p_2}) - PRG(S_{p_1,p_{M-1}}) \\ \qquad\qquad\qquad\qquad\qquad p_a = p_1 \\ u_{p_{M-1}} - PRG(S_{p_{M-1},p_{M-2}}) + PRG(S_{p_{M-1},p_1}) \\ \qquad\qquad\qquad\qquad\qquad p_a = p_{M-1} \\ u_{p_a} - PRG(S_{p_a,p_{a-1}}) + PRG(S_{p_a,p_{a+1}}) \\ \qquad\qquad\qquad\qquad\qquad Otherwise \end{cases} \quad (10)$$

Besides, to reduce the communication cost, this model can reduce the number of samples in each round of training. At the beginning of each round, a proportion of random samples are selected, and the participants only calculate the data of the selected sample. From a general perspective, the

Algorithm 1: MVFLS

**Input:** Parameter $g$, Batch Size $b$
**Output:** Weight Parameters $\theta_{p_1}, ..., \theta_{p_M}$

1: Initialize $\theta_{p_1}, ..., \theta_{p_M}$
2: **for** each iteration i = 1,2... **do**
3: ▷ run on $p_1, ..., p_M$
4: Generate $< sk_{p_a}, pk_{p_a} >$
5: Exchange their public key $pk_{p_1}, ..., pk_{p_{M-1}}$
6: Select a mini-batch sample $s \leftarrow b$
7: **if** a = 1 **then**
8: Compute $y_{p_a}(u_{p_a}, pk_{p_{M-1}}, pk_{p_2}; s)$
9: **else if** a = M-1 **then**
10: Compute $y_{p_a}(u_{p_a}, pk_{p_1}, pk_{p_{M-2}}; s)$
11: **else**
12: Compute $y_{p_a}(u_{p_a}, pk_{p_{a-1}}, pk_{p_{a+1}}; s)$
13: **end if**
14: Sent $y_{p_a}$ to $p_M$
▷ run on $p_M$
15: **for** each $p_a \in P \cup p_a \neq p_M$ in parallel **do**
16: $Y \leftarrow Y + u_{p_M}$
17: **end for**
18: **if** Linear regression **then**
19: $Y \leftarrow Y - y$
20: **else**
21: $Y \leftarrow y - \frac{e^Y}{1+e^Y}$
22: **end if**
23: Send Y to the others
▷ run on $p_1, ..., p_M$
24: $\theta_{pk} \leftarrow \theta_{pk} - \eta \nabla L(Y)$
25: **end for**

model only uses part of the sample data for training in each round.

The whole training process is shown in Figure 2. At first, participants without labels generate encryption pairs( Please see line 4 in Algorithm 1). Then, they broadcast their public keys and calculate their sample set ( Please see line 5 and 6 in Algorithm 1). After participant computes and submit their intermediate data $y_{p_a}$, $p_M$ calculates the secure aggregation value $Y$ or $G$ and sends this results back ( Please see line from 7 to 23 in Algorithm 1). Then, each one updates its weight parameter ( Please refer to line 24 in Algorithm 1).

## Security Analysis

In this section, we prove the security of our model meets in semi-honest settings and our privacy method meets Indistinguishability under chosen-plaintext attack (IND-CPA) security. In other words, we should ensure that any participants cannot infer others' parameters or information.

**Assumption 1** *Semi-honest Security: Participants follow the instructions and requirements of the agreement, but retain the data of interaction with other parties, and try to exploit these data to infer other people's private information.*

The attacker is likely to infer others' local data $x_{p_a}$ through a linear function $u_a = \theta_{p_a} x_{p_a}$. Therefore, to prevent data leakage, each participant adds two masks to hide

Algorithm 2: Encrypted scheme $\Pi$

1: The parameter-generation algorithm $\mathbf{Param}$ takes as input the security parameters $1^n$ and run $\mathcal{G}(1^n)$ to obtain $(G, q, g, Hash)$. $\mathbb{G}$ is a cyclic group of order $q$ with generator $g$ and $Hash$ is a function:$\{0,1\}^* \rightarrow \{0,1\}^n$.
2: The key-generation algorithm $\mathbf{Gen}$ takes as input $(G, q, g)$. Then choose a uniform $x \in Zp$ and compute $h := g^x$. Each party's public key is $< G, q, g, h_i >$ and the private key is $< G, q, g, x_i >$
3: The agreement algorithm $\mathbf{Agree}$ between participants $p_a$ and $p_b$. $p_a$ takes as input $(x_a, h_b)$. Then, compute $S_{a,b} := Hash((h_b)^{x^a} mod\ q)$

their data, and transmits encrypted values. then $p_M$ can aggregate the others' encrypted values $y_{p_a}$, which is equal to the sum of their real data $x_{p_a}$.

Thus, mask values are the key to protecting data privacy. If the mask value can be guessed by $p_M$, the information of participants can be easily calculated. If and only if mask values $PRG(S_{p_a,p_b})$ "look" like random numbers, data privacy leakage can be successfully avoided. In practical applications, *hash()* replaces *PRG()* to implement the encryption mechanism.

**Theorem 1** *This encryption method meets Indistinguishability under chosen-plaintext attack (IND-CPA) security.*

**Proof 1** *We construct encrypted scheme* $\Pi$ *(Param, Gen, Agree) (Algorithm2) and truly random encryption* $\widetilde{\Pi}(\widetilde{Param}, \widetilde{Gen}, \widetilde{Agree})$. *There is a negligible function negl() for PPT adversities:*

$$|Pr\left[Pseu^{cpa}_{\mathcal{A},\Pi} = 1\right] - Pr\left[Pseu^{cpa}_{\mathcal{A},\widetilde{\Pi}} = 1\right]| \leq negl(n) \tag{11}$$

*We create a distinguisher D, which can differentiate pseudorandom numbers from a uniformly random string:*

- *First, in the process of encryption-oracle queries oracle, D runs $\mathcal{A}$, $\mathcal{A}$ queries its encryption oracle() on message m to obtain responds (R+m) and q(n) is upper bound on the round of queries.*
- *Then, the challenge is that when $\mathcal{A}$ outputs a pair of message $m_0, m_1$, D chooses a random $b \in \{0,1\}$ and outputs (R+$m_b$) to $\mathcal{A}$. Note that $\mathcal{A}$ has access to queries oracle continually.*
- *Eventually, $\mathcal{A}$ outputs a bit b': D output 1 if b = b', 0 otherwise.*

*There are the three parts as follows: the first part proofs that*

$$|Pr_{k\leftarrow\{0,1\}^n}\left[D^{F_k(\cdot)} = 1\right] - Pr_{f\leftarrow Func_n}\left[D^{f(\cdot)} = 1\right]| \leq negl(n) \tag{12}$$

*There are two possibilities:*
*(1) If D's oracle function $F_k(\cdot)$ generates pseudorandom number, the view of $\mathcal{A}$ describes that:*

$$Pr\left[Pseu^{cpa}_{\mathcal{A},\Pi} = 1\right] = Pr_{k\leftarrow\{0,1\}^n}\left[D^{F_k(\cdot)} = 1\right] \tag{13}$$

*(2) If $D'$s oracle function f() creates random number ,the view of $\mathcal{A}$ can note that:*

$$Pr\left[Pseu^{cpa}_{\mathcal{A},\widetilde{\Pi}}=1\right]=Pr_{f\leftarrow Func_n}\left[D^{f(\cdot)}=1\right] \quad (14)$$

*Through the function (6), we can draw a conclusion that:*

$$|Pr_{k\leftarrow\{0,1\}^n}\left[D^{F_k(\cdot)}=1\right]-Pr_{f\leftarrow Func_n}\left[D^{f(\cdot)}=1\right]| \leq negl(n) \quad (15)$$

*The second part is proof* $Pr\left[Pseu^{cpa}_{\mathcal{A},\widetilde{\Pi}}=1\right]=\frac{1}{2}+negl'(n)$

- *The situation 1: The value is never used through encryption oracle queries and the probability of D outputting 1 is approximately half(some details in one-time pad case).*
- *The situation 2: The value is applies in the queries:* $q(n)/2^n$.

$$\begin{aligned}
&Pr\left[Pseu^{cpa}_{\mathcal{A},\widetilde{\Pi}}=1\right]\\
&\quad=Pr\left[Pseu^{cpa}_{\mathcal{A},\widetilde{\Pi}}=1\wedge Situation1\right]\\
&\quad+Pr\left[Pseu^{cpa}_{\mathcal{A},\widetilde{\Pi}}=1\wedge Situation2\right]\\
&\quad\leq Pr\left[Pseu^{cpa}_{\mathcal{A},\widetilde{\Pi}}=1\right]\\
&\quad+Pr\left[Pseu^{cpa}_{\mathcal{A},\widetilde{\Pi}}=1\wedge Situation2\right]\\
&\quad=\frac{1}{2}+\frac{q(n)}{2^n}\\
&\quad=\frac{1}{2}+negl'(n)
\end{aligned} \quad (16)$$

*Concrete security shows that:*

$$Pr\left[Pseu^{cpa}_{\mathcal{A},\Pi}=1\right]\leq\frac{1}{2}+negl()$$

In this case, it can protect the participant information in the semi-honest setting and this encryption method meet IND-CPA secure. We not only analyze the security of our model theoretically, but also conduct experiments to test the performance of the model.

## Experimental Results

To verify the effectiveness of the proposed MVFLS, we conduct experiments on both Linear and Logistical regression problems on six real world datasets. The details of these datasets are described in the following section.

### Datasets

We evaluate the performance of our vertical federated learning over six datasets. Table 1 describes the information about these datasets. For linear regression, we first evaluate on Boston and California Housing datasets (Dua and Graff 2017). Boston Housing has 379 records and 14 features and California has 15480 samples and 9 features. We also evaluate on Facebook Metrics dataset (Moro, Rita, and Vala 2016) that includes 500 records and 19 features. For logistic regression, we evaluate on Iris (Dua and Graff 2017), Breast cancer and Default of Credit card clients (Yeh and Lien 2009). Breast cancer has 150 samples, and 4 features and Breast cancer has 426 records and 31 features, and Default of Credit card clients includes 30000 samples and 24 features.

Table 1: Properties of the Datasets

| Dataset | #Instances | #Attributes |
|---|---|---|
| Boston Housing | 379 | 14 |
| Facebook Metrics | 500 | 19 |
| California Housing | 15480 | 9 |
| Iris | 150 | 4 |
| Breast cancer | 426 | 31 |
| Default of Credit card clients | 30000 | 24 |

### Comparison Baselines

We compare our method with state-of-the-art approaches:

**VFLYang (Qiang et al. 2019)**: This method is applied in the VFL for a linear regression model, using Paillier homomorphic encryption for privacy protection. We set only two participants in this model.

**VFLHardy (Hardy et al. 2017)**: It is applied in the VFL for a logistic regression model, adopting Paillier homomorphic encryption for privacy protection, and the Taylor approximation formula is substituted for gradient and Loss function. There are only two participants in this module.

**VFLXu (Xu, Yuan, and Xintao 2019)**: This applies different privacy to protect the privacy of the participants. Furthermore, the participants add some noisy to hide the local data during the process of the transmission.

Our model is MVFLS-$q$, where $q$ means the number of participants (If not specified, MVFLS has three participants). In the VFLXu model, $\epsilon$ is the privacy parameter. And we change the sample batch size, a% means every participant only transmits and computes a% data in each round. We replicate the experiment 20 times and report the averaging results.

### Evaluation Metrics

We compare the performance of different VFL models using the following metrics:

**Time**: The total running time of the VFL model is used to evaluate the efficiency of each method.

**MSE**: MSE (Mean Squared Error) is used to measure the distances between reconstructed data and original data.

**Accuracy**: The accuracy indicates the proportion of the number of samples correctly classified by the model.

**AUC**: The AUC( Area Under Curve ) on the testing set is used for performance evaluation, which can distinguish the performance of the classifier.

### Result

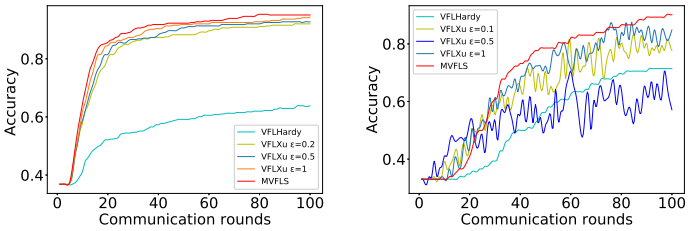
This section mainly describes the performance and the evaluation of MVFLS. Firstly, we evaluate the utility of linear

Table 2: Comparison of time cost on three benchmark datasets for Linear Regression.

| | VFLYang | **MVFLS-3** | MVFLS-4 | MVFLS-5 |
|---|---|---|---|---|
| Boston Housing | 20162.06s ± 578.13s | **29.46s ± 3.16s** | 47.82s ± 1.12s | 59.01s ±12.70s |
| California Housing | 93491.53s ± 889.43s | **37.85s ± 1.20s** | 47.50s ± 0.53s | 57.92s ± 2.14s |
| Facebook Metric | 43825.58s ± 431.02s | **45.50s ± 2.72s** | 50.23s ± 3.54s | 60.43s ± 0.46s |

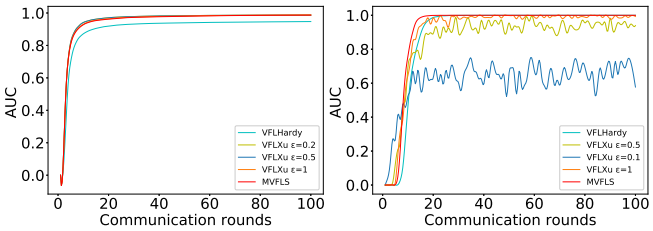
Table 3: Comparison of mean-square error on three benchmark datasets for Linear Regression.

| | VFLYang | VFLXu($\epsilon$=0.1) | VFLXu($\epsilon$=0.5) | VFLXu($\epsilon$=1) | **MVFLS-3** |
|---|---|---|---|---|---|
| Boston Housing | 10.089 ± 0.116 | 54.580 ± 6.354 | 26.814 ± 5.348 | 16.672 ± 0.203 | **10.090 ± 0.431** |
| California Housing | 0.250 ± 0.136 | 0.249 ± 0.271 | 0.248 ± 0.114 | 0.246 ± 0.102 | **0.236 ± 0.228** |
| Facebook Metric | 0.627 ± 0.354 | 9.039 ± 1.175 | 1.565 ± 0.392 | 1.306 ± 0.011 | **0.629 ± 0.048** |




(a) Cancer (b) Iris

Figure 3: Classification accuracy of logistic regression on Cancer and Iris dataset




(a) Cancer (b) Iris

Figure 4: AUC of logistic regression on Cancer and Iris dataset

regression by training time and MSE (Mean Squared Error) in Table 2 and 3. Secondly, we compare our model with classical models for Logistic regression in terms of training time (Table 4), accuracy, and AUC (Area Under Curve) (Figure 3(a) to 4(b)). Finally, this experiment testes the model with different samples in every round. The result is shown in from Figure 5(a) to 5(c).

**Experimental Results on Linear Regression Task** For Linear regression, we record the training time on four methods in Table 2. MVFLS-3 always achieves the fastest training rate compared to others. MVFLS-4 and MVFLS-5 take twice or third times as MVFLS-3 to train. With the participants involved, the running time is inevitably improved. All of our three methods are more efficient than the traditional method. For comparison, the time cost of MVFLS-3 is 101–2470× faster than traditional VFL. Especially when it runs on larger data set California Housing, Our module achieves significantly faster than the best performing baseline by 2470 times.

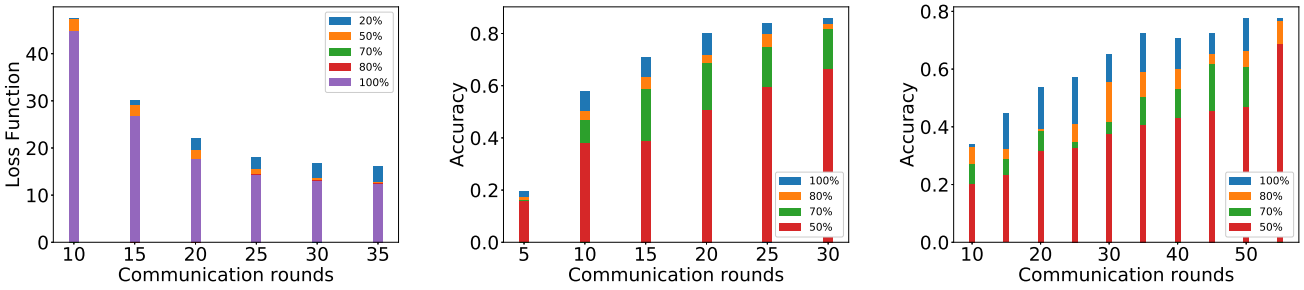
Table 3 compares the MSE among different methods. In terms of MSE, our model is almost the same as VFLYang, which means our model maintains the accuracy of the model. The MSE of VFLXu varies with the change of the privacy parameter $\epsilon$. As $\epsilon$ decreases, the added noise value increases, resulting in lower MSE. Overall, MVFLS has a higher accuracy rate than VFLXu. It can be seen from the two tables that the model we designed has a good performance in both computational efficiency and MSE.

**Experimental Results on Logistic Regression Task** Table 4 records the running time of MVFLS and VFLHardy for logistic regression. The results are similar to that of time cost for linear regression. The time cost of the VFLHardy model is much higher than that of the other models. For comparison, the time cost of VFLHardy is 391–3059× as MVFLS-3. For MVFLS with different participants, the training time of the model will be longer with participants' increase, but they are far less than VFLHardy. It can be concluded that compared with VFLHardy, MVFLS model can improve computation efficiency and show better model performance.

In the figures 3(a) and 3(b), experiments compare our model with VFLHardy and VFLXu through the accuracy metric on Iris and breast cancer datasets. On the cancer data set (as shown in Figure 3(a)), when the accuracy of our model finally reached 95% and this result was slightly higher than that of VFLXu, the accuracy of VFLHardy was only 63%. Therefore, MVFLS has certain advantages in the cancer data set. In the figure 3(b) , the accuracy of this model reached about 90%, while VFLHardy can only reach about 70% on Iris data set. And, when privacy parameter $\epsilon$ is equal to 1, the accuracy of VFLXu finally reaches 86%. As $\epsilon$ decreases, the accuracy rate decreases and fluctuates sharply.

Table 4: Comparison of time cost on three banchmark datasets for Logistic Regression

| | VFLHardy | **MVFLS-3** | MVFLS-4 | MVFLS-5 |
|---|---|---|---|---|
| Iris | 3912.35s ± 282.04s | **10.89s ± 2.05s** | 43.78s ± 7.97s | |
| Breast Cancer | 22414.47s ± 869.70s | **20.19s ± 1.50s** | 45.54s ± 2.01s | 58.87s ± 1.19s |
| Default of Credit card clients | 183554.60s ± 551.72s | **59.99s ± 2.80s** | 104.73s ± 13.76s | 258.84s ± 72.1s |




(a) Boston Housing (b) Cancer (c) Iris

Figure 5: Sensitivity study results on batch size

In terms of Figure 4(a) and 4(b), we compare the performance of different models according to the AUC metric. On the breast cancer data set, the growth trend of the AUC in our model is similar to that of VFLXu, both of which are higher than that of VFLHardy. On the Iris data set, AUC in our model has the fastest and most stable growth. The AUC trend of VFLHardy is slightly slower than our model, and that of VFLXu with $\epsilon = 1$ is similar to MVFLS model. When the $\epsilon$ becomes smaller, AUC goes up more slowly with more sharply fluctuation.

**Sensitivity Study on batch size** To further reduce the communication cost, this experiment reduces the size of transmitted data. Generally speaking, participants in each round randomly select a certain percentage of samples to participate in every round during vertical federal learning training, and we test our model with $20\%$ $50\%$ $70\%$ $80\%$ of random samples in one round. The experimental results are shown from Figure 5(a) to Figure 5(c).

On the Boston Housing dataset, each participant transmits information about $20/50/70/80/$ percentage of samples in one round. It can be seen from the figure 5(a) that when only $20\%$ of the samples are trained in one round, the loss function value is slightly larger than the full data set training and the results of others are similar to the full data set sample training results. The experiment 5(b) compares the changes in the different sizes of transmitted random samples on the Breast cancer data set. When $80\%$ of the data is involved in the training, the result is similar to the full data set training. In Figure 5(c), the result of $80\%$ of the samples in one round training is closest to the result of the lossless data set. The model trained with $20\%$ of the samples has the lowest accuracy and the worst performance. As the size of training samples decreases, the accuracy grows slowly and model performance will continue to decline.

If each participant appropriately reduces the amount of transmitted sample data, it can deeply reduce communication costs.

## Conclusion

This paper proposed a Multi-participant Vertical Federated Learning based on Secret Sharing model (MVFLS). This method uses secret sharing to solve the problem of privacy protection in VFL settings, which has lower communication and higher calculation efficiency of the privacy protection compared to existing vertical federated learning methods based on HE. In addition, to simplify the model and reduce the amount of communication, we introduced a method without a third-party server, which further improves the performance of the model under VFL. Experiments on Linear and logistical regression problems demonstrate the superiority of MVFLS over other state-of-the-art methods. Though MVFLS has shown promising results for vertical federated learning problems, our privacy method has lower security than HE. In our future work, we will focus on this direction.

## References

Brakerski, Z.; Gentry, C.; and Vaikuntanathan, V. 2014. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3): 1–36.

Cheon, J. H.; Kim, A.; Kim, M.; and Song, Y. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, 409–437. Springer.

Damgård, I.; Pastro, V.; Smart, N.; and Zakarias, S. 2012.

*Multiparty Computation from Somewhat Homomorphic Encryption*. Springer-Verlag New York, Inc.

Diffie, W.; and Hellman, M. 1976. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6): 644–654.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

ElGamal, T. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4): 469–472.

Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 1322–1333.

Gentry, C. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 169–178.

Gentry, C.; Sahai, A.; and Waters, B. 2013. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In *Annual Cryptology Conference*.

Geyer, R. C.; Klein, T.; and Nabi, M. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*.

Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; and Wernsing, J. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, 201–210. PMLR.

Hardy, S.; Henecka, W.; Ivey-Law, H.; Nock, R.; Patrini, G.; Smith, G.; and Thorne, B. 2017. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:arXiv:1711.10677*.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2019. Advances and open problems in federated learning. *CoRR, arXiv:1912.04977*.

Keith, B.; Vladimir, I.; Ben, K.; Antonio, M.; Hugh, M., Brendan; Sarvar, P.; Daniel, R.; Aaron, S.; and Karn, S. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*.

Kilbertus, N. 2018. Blind Justice: Fairness with Encrypted Sensitive Attributes.

Kim, M.; Song, Y.; Shuang, W.; Xia, Y.; and Jiang, X. 2017. Secure Logistic Regression Based on Homomorphic Encryption: Design and Evaluation. *JMIR Medical Informatics*, 6(2).

Konečnỳ, J.; McMahan, H. B.; Ramage, D.; and Richtárik, P. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

Melis, L.; Song, C.; De Cristofaro, E.; and Shmatikov, V. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, 691–706. IEEE.

Mohassel, P.; and Zhang, Y. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *Security amp; Privacy*, 19–38.

Moro, S.; Rita, P.; and Vala, B. 2016. Predicting social media performance metrics and evaluation of the impact on brand building: A data mining approach. *Journal of Business Research*, 69(9): 3341–3351.

Niemi; Valtteri; and Renvall, A. 1998. Secure multiparty computations without computers. *Theoretical Computer Science*, 35(4): 173–183.

Nock, R.; Hardy, S.; Henecka, W.; Ivey-Law, H.; Patrini, G.; Smith, G.; and Thorne, B. 2018. Entity resolution and federated learning get a federated resolution. *arXiv preprint arXiv:1803.04035*.

Paillier, P. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Stern, J., ed., *Advances in Cryptology — EUROCRYPT '99*, 223–238. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-48910-8.

Qiang, Y.; Yang, L.; Chen, T.; and Tong, Y. 2019. Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology*.

Rabin, M. O. 2005. How To Exchange Secrets with Oblivious Transfer. *IACR Cryptol. ePrint Arch.*, 2005(187).

Rivest, R. L.; Adleman, L.; Dertouzos, M. L.; et al. 1978. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11): 169–180.

Shamir, A. 1979. How to share a secret. *Communications of the ACM*, 22(11): 612–613.

Xu, D.; Yuan, S.; and Xintao, W. 2019. Achieving Differential Privacy in Vertically Partitioned Multiparty Learning. *arXiv preprint arXiv:1911.04587*.

Yao, A. C.-C. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, 162–167. IEEE.

Yeh, I.-C.; and Lien, C.-h. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2): 2473–2480.

Yu, H.; Vaidya, J.; and Jiang, X. 2006. Privacy-preserving svm classification on vertically partitioned data. In *Pacific-asia conference on knowledge discovery and data mining*, 647–656. Springer.