World Wide Web (2023) 26:2691–2707
https://doi.org/10.1007/s11280-023-01159-x

# Beyond model splitting: Preventing label inference attacks in vertical federated learning with dispersed training

**Yilei Wang[1,2] · Qingzhe Lv[1] · Huang Zhang[2] · Minghao Zhao[3] · Yuhong Sun[1] · Lingkai Ran[1] · Tao Li[1,4]**

Received: 29 September 2022 / Revised: 21 December 2022 / Accepted: 25 February 2023 /
Published online: 8 May 2023


**Abstract**
Federated learning is an emerging paradigm that enables multiple organizations to jointly train a model without revealing their private data. As an important variant, *vertical* federated learning (VFL) deals with cases in which collaborating organizations own data of the same set of users but with disjoint features. It is generally regarded that VFL is more secure than horizontal federated learning. However, recent research (USENIX Security'22) reveals that it is still possible to conduct label inference attacks in VFL, in which attacker can acquire privately owned labels of other participants; even VFL constructed with model splitting (the kind of VFL architecture with higher security guarantee) cannot escape it. To solve this issue, in this paper, we propose the dispersed training framework. It utilizes secret sharing to break the correlations between the bottom model and the training data. Accordingly, even if the attacker receives the gradients in the training phase, he is incapable to deduce the feature representation of labels from the bottom model. Besides, we design a customized model aggregation method such that the shared model can be privately combined, and the linearity of secret sharing schemes ensures the training accuracy to be preserved. Theoretical and experimental analyses indicate the satisfactory performance and effectiveness of our framework.

**Keywords** Vertical federated learning · Label inference attack · Secret sharing · Dispersed training

## 1 Introduction

Machine learning has gained great success in numerous fields, such as decision-making, risk identification, and disease diagnosis. The widespread adoption of machine learning and its

This article belongs to the Topical Collection: *Special Issue on Privacy and Security in Machine Learning*
Guest Editors: Jin Li, Francesco Palmieri and Changyu Dong.

✉ Tao Li
litao_2019@qfnu.edu.cn

Extended author information available on the last page of the article

effectiveness is mainly attributed to, except for the advances of ML algorithms, the increasing affordability of collecting, storing, and processing large quantities of data. Especially, the feasibility of joint and integrated use of data from multiple sources (e.g., data collected by different institutes or stored in different data centers). However, sharing data may encounter security and privacy issues. With the increase in awareness of data privacy protection, sharing data containing citizens' sensitive information is legislatively forbidden. For example, the regulations like the General Data Protection Regulation (GDPR) [1, 2] in European Union, the Personal Data Protection Act (PDPA) [3] in Singapore, and California Consumer Privacy Act (CCPA) [4] in the US.

Under such circumstances, federated learning (FL) is proposed and gets great popularity. It enables multiple data owners (e.g., clients or data centers) to collaboratively train ML models without revealing their private data. As the basic workflow, participants in FL iteratively (i) conduct local computations on their data to derive certain intermediate results, (ii) conceal the intermediate results with certain cryptographic tools, and (iii) share the protected result with other participants, until the final training result achieved.

According to different kinds of data partitioning, FL can be categorized as *horizontal federated learning* (HFL) and *vertical federated learning* (VFL). HFL deals with the case when the data is horizontally partitioned, i.e., the datasets share the same feature space but differ in the sample space. For example, two hospitals hold medical records of different groups of patients (i.e., the sample space) that describe whether they have a certain disease (i.e., the feature space). Comparatively, VFL deals with the case when the data is vertically partitioned, i.e., datasets share the same sample space but differ in the feature space. A typical example of VFL is that, two hospitals hold medical records of the same group of patients (i.e., sane sample spaces), but each of them describes different aspects of the medical status of a patient, e.g., dataset from hospital A records the COVID swab test of the patients, whereas dataset from hospital A records chest CT scan of them (i.e., different feature spaces).

Due to its relatively symmetric structure, the construction of HFL is simple. It normally is constructed via model averaging – for each iteration the client train a model for several epochs and sends its model to the client, who afterward aggregates the submitted local model and gets the updated global model. But in terms of VFL, especially VFL constructed with model splitting, the training process is relatively complex. In VFL with model splitting, the whole model is divided into a top model (holding at the server) and several bottom models (as specific intermediate states preserved at the clients). The training process is finished with iteratively *bilateral* communication. The clients run their bottom model with their local data and upload the result to the server; the server runs a top model to aggregate the participants' output, computes the gradients of the loss, and sends the gradients back to every participant.

Accordingly, the potential attackers of HFL have chance to peep gradients of all the parameters of the model, which can be used to infer private information. Comparatively, attackers of VFL only control part of the federated model and accordingly can only get gradients of an incomplete model. Thus, it is generally believed that VFL has a higher security guarantee, especially for VFL constructed with model splitting (in which the participants have no access to the last layer of the DNN and thus the label in the server are more secure) [5]. In spite of this, recent research reveals that it is still feasible to conduct inference attacks on VFL [6]. Specifically, Fu et al. [6] find that, the gradients sent from the server essentially can help the client to learn a good feature representation with respect to the labels. This leaked information serves as a pre-trained model for label inference attacks.

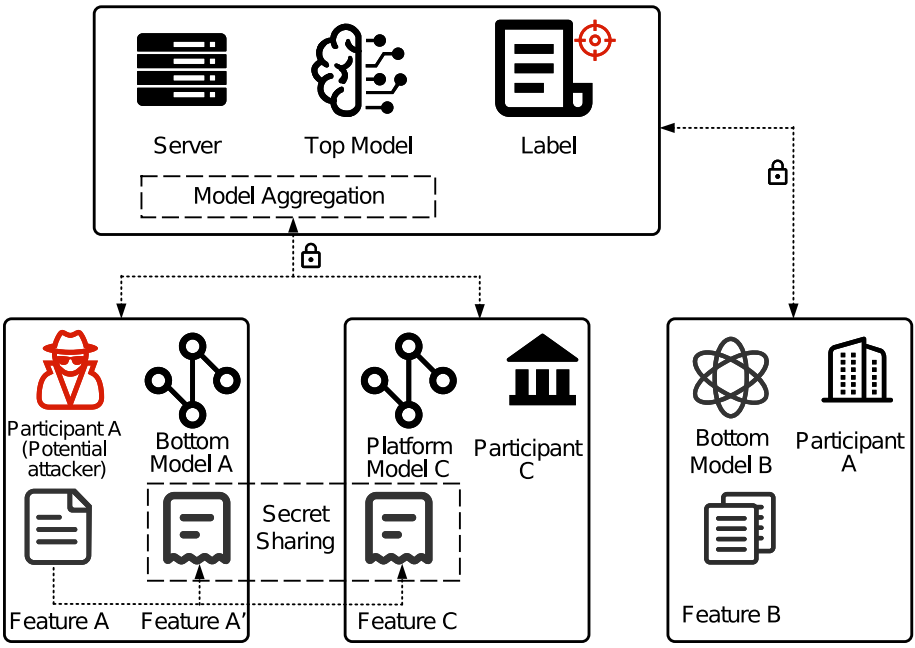


**Figure 1** Architecture and workflow of the dispersed training framework

In this paper, we propose the *dispersed training* framework, to combat such attacks and enable the security of VFL. The basic idea of dispersed training is to utilize secret sharing breaking the correlations between the gradients and the training data. As shown in Figure 1, participant B holds his own data with labels hoping to train a model; he hopes to utilize participant A's data to improve the quality of his model. Participant A is the positional attacker and he is intended to infer participant B's label. In the dispersed training framework, a shadow model (i.e., participant C) is created for participant A, and part of the data of participant B is shared to participant C. In the training phase, the clients (i.e., participants A and C) update their bottom model with their shared data and upload their partial outputs to the server. The server aggregates the clients' partial outputs and trains its top model; their outputs can be efficiently aggregated, due to the linearity of the secret sharing schemes. The server's training output is also segmented into two parts and delivers each segment the A and C, respectively, who later iteratively train and update their bottom model. With such a method, even if the attacker receives the gradients in the training phase, he is incapable to deduce the feature representation of labels from the bottom model.

The rest of the paper is organized as follows. Section 2 presents basic backgrounds of federated Learning and describes how inference attack is conducted on vertical federated Learning. We present our dispersed training framework and its construction on Section 3. The performance evaluation is presented on Section 4. Related work on this topic is presented at Section 5, and we conclude this paper on Section 6.

## 2 Inference attacks in VFL: an introduction and analysis

### 2.1 Federated learning

Compared with the common centralized learning, the federated learning provides a collaborative learning method to perform distributed training models on data, which protects the privacy of data [7]. According to the characteristics of data, the federated learning is mainly divided into three categories: horizontal federated learning, vertical federated learning, and transfer federated learning [8].

Horizontal federated learning is suitable for the situation where the data features of the participants overlap a lot, but the IDs of the samples overlap less. The purpose of horizontal federated learning is to train a more accurate prediction model by combining more samples with the same characteristics. For example, two e-commerce platforms in different regions have purchase records of the same consumption level in their respective regions. The two e-commerce platforms can train a model through horizontal federated learning and can push products to users of this consumption level. Vertical federated learning is suitable for samples with more overlapping IDs and less overlapping data features. The purpose of the vertical federated learning is to train a model by combining more features of the same sample. For example, a bank and a lending company in the same area, the bank has data on a group of people's economics status and whether they owe money, while the lending company has another asset status of the same group of people. Integrating the lending company and the bank into cooperate method can carry out vertical federated learning by using their data, and train an accurate model to predict risk and decide whether to service a loan to someone [8]. Transfer federated learning is more suitable when the data characteristics of the participants and the number of samples overlap less, or the distributions of the participants differs greatly. For example, hospitals and lending companies in different regions want to train a prediction model, but the two parties have little overlap in data features and sample numbers. At this time, transfer federated learning can be used to train the model [9].

### 2.2 Label inference attacks in VFL

The label inference attack of vertical federated learning is proposed by Fu et al. [6], which reveals the privacy relationship between vertical federated learning and labels. Fu et al. [6] designed three attack methods to reason about labels with high privacy, which can enable malicious participants to perform inferring attacks on labels of any participant's data, resulting in serious privacy leakage.

Passive label inference attack is vertical federated learning for model splitting. Although no participant can access the top model, the trained bottom model can be used for inference attacks [10]. Its principle is that the adversary can convert its features into indicative information about labels during the training process, and use that information to make predictions. The adversary uses vertical federated learning to train the bottom model with indicative features, and then adds an initialization layer to the upper layer of the trained bottom model to form a complete model, and uses a small amount of labeled data to perform semi-supervised learning on the attack model. With this model, the label can be directly inferred.

Active label inference attacks use the adversary to accelerate the gradient descent of the bottom model, so that the bottom model is better trained in each iteration, which can provide better training features to the server, making the top model of the server more dependent on the bottom model, and the adversary then fine-tunes this bottom model to form a complete

attack model that can perform inference attacks on labels. Fu et al. [6] designed a malicious local optimization algorithm and added an algorithm to limit the learning rate to the gradient descent algorithm, which can ensure that the learning rate is appropriate, so as to accelerate the gradient descent and make the malicious bottom model better trained.

The direct label inference attack is aimed at vertical federated learning without model splitting. The adversary can receive the gradient leaked from the server. Fu et al. [6] proved by mathematical analysis that the adversary can directly analyze the leaked gradient sign. When the gradient sign is negative, it proves that the guessed label is exactly the actual label. When the gradient sign is positive, the guessed label is incorrect. This can cause serious label leakage.

In summary, all three attacks result in serious privacy breaches. Passive label inference attack is due to the bottom model having better inference ability, and the bottom model is trained by semi-supervised learning to form an attack model to perform label inference attack. Active label inference attack is on top of the base, using malicious SGD to accelerate gradient descent, resulting in a malicious bottom model that can be trained better and has better inference attack capability. The direct label inference attack, on the other hand, is not very related to the first two. The direct label inference attack is directly caused by the leakage of label information through the gradient leaked from the server. Due to previous work [6], common privacy protection methods cannot be used to defend against passive label inference attacks and active label inference attacks, the proposed model in this paper is mainly for the discussion of the first two attack methods.

## 3 Dispersed training

### 3.1 The training model

Before describing more details with respect to the training model, we present the model in brief. First, a shadow model $c$ is generated, which has the same structure with $a$ but with a benign type. The difference between the malicious and benign types is that when training the model, the benign model utilizes SGD for gradient descent, while the malicious model utilizes its own local optimizer to accelerate gradient descent. Then the malicious model shares its training dataset with the shadow model through secret sharing, and performs local training on the bottom models. Consequently, the output of the bottom model is uploaded to the top model after being merged and aggregated. Finally, the top model assigns the gradient to the bottom models. Note that the shadow model should share the gradient with the malicious model. The above process is repeated until the model converges.

Algorithm 1 is the pseudo-code of the training model. More details are as follows.

1. **Initialization.** Initialize a shadow model $c$, where the structure is similar to $a$ with a benign type. Note that although model $c$ has the same structure, the training process is different from that of model $a$. Then, initialize the parameters of the bottom model and the top model, respectively.
2. **Dataset Splitting.** According to the assumptions in the literature [6], there are two bottom models $a$ and $b$ in the training model. The training sets are $x_a$ and $x_b$, respectively. in which participant $A$ holding model $a$ is an attacker, and the participant $B$ holding model $b$ is an honest participant. Here we only need to split the dataset $x_a$ of model $a$. That is, the bottom model $a$ and shadow model $c$ share the training set data $x_a$. Specifically, $x_a$ is split

**Algorithm 1** The framework for training process.

**Input:** Training data set $x_a$, $x_b$, $y_{top}$ for Bottom model $a, b$, Top model $top$ respectively, learning rate $\eta$.
**Output:** The well trained parameters $\theta_a, \theta_b$ for Bottom model $a, b$.

**(1) Initialize:**
① For malicious model $a$, create model $c$ by copying model $a$ and set it to be a benign party;
② Initialize the parameters $\theta_i$ $(i \in a, b, c)$, $\theta_{top}$ for $a, b, c, top$ respectively.

**(2) Dataset splitting:**
① Choose a random value $\alpha$;
② Assign $\alpha x_a$ and $(1-\alpha)x_a$ to model $a$ and $c$ as their training dataset respectively;

**(3) Bottom model training:**
**while** each epoch does not stop **do**
  **for** each batch $bat_i (i \in a, b, c)$ of sample Ids **do**
    **for** model $a, b, c$ **do**
      $o_i \leftarrow$ **ParticipantForwardProp**$(\theta_i, bat_i)$
    **end for**
  **end for**
**end while**

**(4) The merging and aggregation of the Output:**
① $o_a \leftarrow Mergshare(o_a, o_c)$ ▷ concatenating outputs of $a$ and $c$
② $o_{all} \leftarrow Concat(o_a, o_b)$ ▷ concatenating all bottom model outputs

**(5)Top model training:**
① $o_{final} \leftarrow$ **ServerForwardProp**$(\theta_{top}, o_{all})$
② $L \leftarrow LossFunc(o_{final}, y)$
③ $g_{top} \leftarrow \frac{\partial L}{\partial \theta_{top}}$
④ $\theta_{top} \leftarrow \theta_{top} - \eta \cdot g_{top}$ ▷ updating top model

**(6)Sharing the gradient and update model:**
① **for** model $a, b$ **do**
② $g_a \leftarrow \frac{\partial L}{\partial o_a}, g_b \leftarrow \frac{\partial L}{\partial o_b}$
③ $\{g_a, g_c\} \leftarrow SS(g_a)$ ▷ Secret sharing of gradient
④**ParticipantBackProp**$(\theta_i, g_i, o_i)$:

**ParticipantForwardProp**$(\theta_i, bat_i)$:
**return** bottom model forward outputs $\theta_i(bat_i)$

**ServerForwardProp**$(\theta_{top}, o_{all})$:
**return** top model forward outputs $\theta_{top}(o_{all})$

**ParticipantBackProp**$(\theta_i, g_i, o_i)$:
① $g_i \leftarrow g_i \cdot \frac{\partial o_i}{\partial \theta_i}$
② $\theta_i \leftarrow \theta_i - \eta \cdot g_i$ ▷ updating bottom model

into $\alpha x_a$ and $(1-\alpha)x_a$, where $\alpha$ is a random number, $\alpha x_a$ is participant $A$'s training set, and $(1-\alpha)x_a$ is participant $C$'s training set

3. **Training for Bottom Model.** In each round of local training, models $a$, $b$, and $c$ use their respective training datasets for local training. However, the training data is too large, in the actual training process, the training data is divided into blocks (denoted as $bat_i$, where $i \in a, b, c$). It should be noted that the bottom model is a linear model, and the activation function $Relu()$ is also linear. Therefore, during the entire training process, the parameters of each layer between the bottom model $a$ and $c$ are always linear.
4. **Merging and Aggregation of Outputs.** This stage is divided into two steps:

(a) **The merging of outputs of the model $a$ and $c$.** Here, although both participants $A$ and $C$ participated in the training locally, for the top model, there is still only an interface with model $a$. Therefore, in the dispersed training model, we design a merge function $Mergshare()$. After the model $a$ and $c$ have trained the local model to generate the output $o_a$, $o_c$, the function treats $o_a$, $o_c$ as two shares, and merges the results of model $a$ and $c$ into a new output $o_a$ through $Mergshare()$. The new output $o_a$ is uploaded to the top model through the interface between the top model and model $a$.
Note that models $a$ and $c$ maintain a linear relationship during training, so their merged result is consistent with the upload results after model $a$ is trained alone. That is to say, the new output $o_a$ uploaded to the top model under the name of $A$ is the same as the previous work [6] that $A$ directly uploads the output to the top model. Here we assume that there is an output merging layer among models $a$, $c$ and the top model. Its main function is to combine the outputs of models $a$ and $c$ to the top model. At the same time, the gradients received from the top model need to be secretly shared first, and then downloaded to models $a$ and $c$, respectively.

(b) **The aggregation of outputs $a$ and $b$.** After the merging in step 4(a), the top model obtains $o_a$ and $o_b$ through its interface with models $a$ and $b$, respectively. At this time, according to the method of previous work [6], the function $Concat()$ is used to aggregate $o_a$ and $o_b$ to generate $o_{all}$, which is further trained by the top model.

5. **Training for Top Model.** The top model is trained with the output $o_{all}$ and to get the output $o_{final}$.
6. **Gradient Download.** After the top model is locally trained, the loss function is first calculated, and then the gradient is passed down to the bottom model. In the previous work [6], the top model will pass gradients down to models $a$ and $b$, respectively. In this paper, the process of model $b$ receiving its gradient is similar to the previous work [6]. However, it's different for downloading the gradient to the model $a$ since model $c$ is involved. In particular, the gradient $g_a$ of the model $a$ is generated first. Then a new $g_a$ and $g_c$ are obtained by the secret sharing function $SS()$, which are passed to the models $a$ and $c$. This function is implemented by the merging layer.
7. Repeat (1)-(6) for the bottom model and the top model until converge.

### 3.2 The attack model

In the attack model, we still use the model completion proposed by Fu et al. [6]. After completing the federated training process, we will get the trained bottom model. This bottom model also has strong capabilities to run label inferring attack. We retrain the bottom model and use a small amount of labeled data as Fu et al. [6]. More specifically, the malicious attacker adds an extra layer to continue semi-supervised learning, and finally trains the bottom model. The newly trained model is used as an attack model to execute label inference attack.

## 4 Performance evaluation

### 4.1 Experiments settings

All experiments are performed on Intel(R) Core i5-12500H @ 2.50GHz, 16GB RAM, NVIDIA GeForce RTX 2050 card (Table 1). The three datasets used in the experiment

**Table 1** Experimental equipment, experimental data on the indicators

| Experimental equipment | Card | Experiment Indicators |
|---|---|---|
| Intel(R) Core i5-12500H | NVIDIA GeForce RTX 2050 | Top-1 Accuracy |

are CIFAR-10, CINIC-10, and BCW, which are also the datasets used by previous work [6]. In this paper, Top-1 Accuracy is selected as the performance indicator for the attack effect of federated original tasks and label inferring. Top-1 accuracy means that the predicted label takes the largest one in the final probability vector as the prediction result. If this prediction result is the actual label, it means the prediction is correct, otherwise, the prediction is wrong. In the process of training the attack model, this paper selects an additional small amount of labeled data for semi-supervised training of the bottom model. In the attack model experiment, CIFAR-10 and CINIC-10 selected 40 labeled samples, and the BCW dataset selected 20 labeled data for semi-supervised training. As mentioned in the previous work [6], the number of labels will affect the effect of the label inference attack. When the number of samples reaches a certain number, the result of the label inference attack will grow slowly. Therefore, the same as in the previous work [6], In this experiment, 40 data and 20 data are also selected.

**CIFAR-10**: CIFAR-10 is a typical classification dataset, which contains 60,000 images, of which 50,000 images are used as a training set and 10,000 images are used as a test set. Each photo is a 32*32 color image, and there are 10 different categories of images in the entire dataset.

**CINIC-10**: CINIC-10 is an extension of CIFAR-10 through subsampled ImageNet images, and like CIFAR-10, it is also divided into 10 categories. To solve the problem of the small number of CIFAR-10, the CINIC-10 dataset appeared. It contains 270,000 images, which is 4.5 times larger than CIFAR-10. The images are evenly divided into three subsets: training set, validation set, and test set. Each subset has 90,000 images, the number of training sets is 1.8 times that of CIFAR-10, and the number of test sets is 9 times that of CIFAR-10. Through this data set, we can test the effect of our designed scheme on large data sets.

**BCW**: The Breast Cancer Wisconsin (BCW) dataset is a breast cancer dataset with a total of 569 pieces of data and 32 feature columns, mainly for nuclear features. The label of the sample is whether the diagnosis is benign or malignant. The data set used in the experiment is the data set used by Fu et al. [6], which randomly selects 426 samples as the training set and the remaining 143 samples as the test set.

In order to compare with the paper proposed by Fu et al. [6], we use the same top and bottom model structure of Fu et al. [6]. For large datasets like CIFAR-10 and CINIC-10, the bottom model is chosen as the residual network and the top model is full connect neural network, for the BCW dataset both the bottom and top models are used as fully connected neural network, and the specific structure is shown in Table 2.

### 4.2 Comparison with original attack

The performance of the model after dispersed training in the face of a passive label inference attack is shown in Figure 2. It can be seen from the Top-1 accuracy of the attack on the three datasets, compared with the label inference attack proposed by Fu et al. [6] the effect has

**Table 2** Neural network structure of the bottom model and top model

| Dataset | Bottom model structure | Top model structure |
|---|---|---|
| CIFAR-10 | ResNet-18 | FCNN-4 |
| CINIC-10 | ResNet-18 | FCNN-4 |
| BCW | FCNN-3 | FCNN-3 |

dropped significantly, especially on the CIFAR-10 dataset by about 70%. After dispersed training, the Top-1 accuracy rates of attacks on CIFAR-10, CINIC-10, and BCW datasets are 9.99%, 10.02%, and 36.36%, respectively. For the CIFAR-10 and CINIC-10 datasets, the Top-1 accuracy rate of the attack is reduced to about 10%. These two datasets only have ten categories, so the Top-1 accuracy rate of the 10% attack is equivalent to the Top-1 accuracy rate for these ten categories to random guess. The results from the CIFAR-10 dataset to the CINIC-10 dataset are all around 10%, indicating that our scheme also has a good defense effect on large datasets. Overall, the dispersed training proposed in this paper can effectively prevent passive label inference attacks.

In addition, this paper also compares active label inference attacks. As can be seen from Figure 3, the model after dispersed training is also effective against active label inference attacks. On the CIFAR-10 dataset, the attack Top-1 accuracy rate dropped from 84.84% to about 10%, and the attack Top-1 accuracy rate for the other two datasets also dropped significantly.

Overall, our scheme can reduce the accuracy of label inference attacks to random guessing, which proves that dispersed training can effectively mitigate label inference attacks.

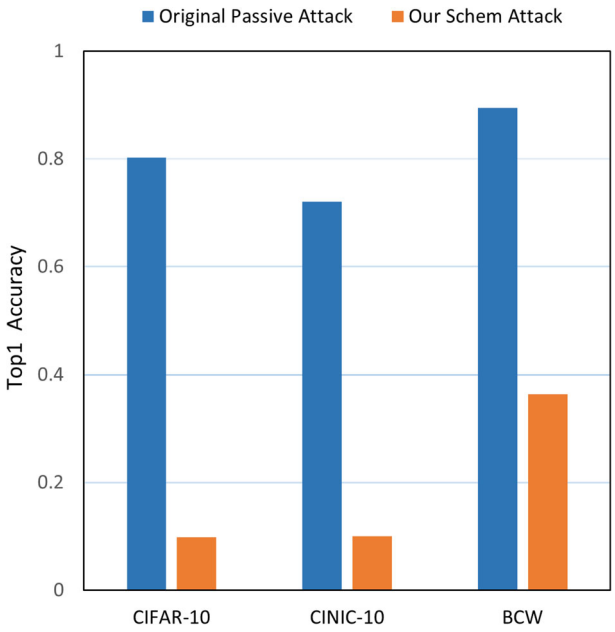


**Figure 2** Performance comparison between the original passive attack and our attack

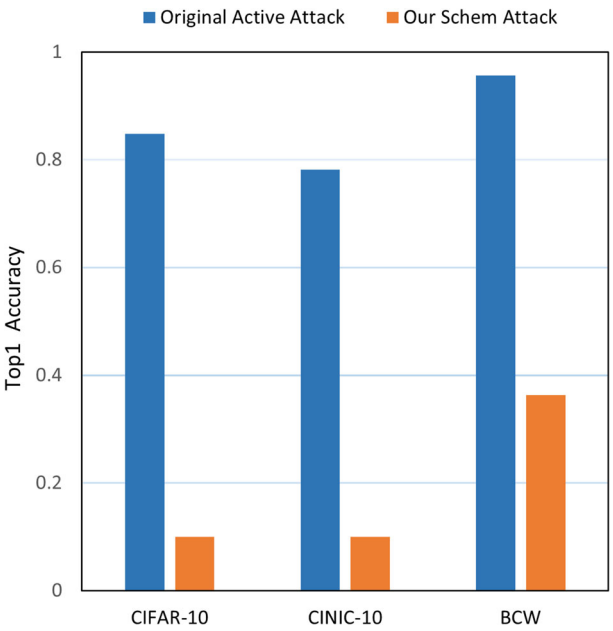


**Figure 3** Performance comparison between the original active attack and our attack

### 4.3 Comparison with original federated tasks

Table 3 shows the accuracy of the original federated tasks for each dataset after dispersed training. Compared with the original federated learning, the federated accuracy decreased after dispersed training. It drops by 4%, 15%, and 18% on the BCW, CIFAR-10, and CINIC-10 datasets, respectively. Among them, the accuracy of the CIFAR-10 and CINIC-10 datasets dropped significantly, which was caused by the large datasets. Although the label inference attack can be effectively prevented after dispersed training, the federated accuracy has also decreased to a certain extent, especially on large datasets. Therefore, the implement of dispersed training requires a trade-off between defending against label inference attacks and federated training accuracy.

### 4.4 Comparison with gradient compression

Compared to common machine learning privacy-preserving methods, our scheme can mainly make the performance of active label inference attacks and passive label inference

**Table 3** The performance of the original federated tasks and our federated tasks after dispersed training

| Dataset | Metric | Original federated tasks | Our federated tasks |
|---|---|---|---|
| CIFAR-10 | Top-1 Acc | 0.8280 | 0.6784 |
| CINIC-10 | Top-1 Acc | 0.7132 | 0.5306 |
| BCW | Top-1 Acc | 0.8671 | 0.8252 |

**Table 4** Defense performance of gradient compression and our scheme against the active label inference attack

| Dataset | Original active attack | Gradient compression | Our scheme attack |
|---|---|---|---|
| CIFAR-10 | 0.8484 | 0.64 [6] | 0.099 |
| CINIC-10 | 0.7818 | 0.5884 | 0.1002 |

attacks reduced to random guesses and can make the performance of the original federated task not degraded too much. For example, in the method of adding noise to the gradient, the performance of the original federated learning task decreases from 0.8 to about 0.1, which can lead to the failure of the federated task. Let's take gradient compression, a privacy-preserving method, as an example. As shown in Table 4, the active label inference attack decreases from 0.8484 to 0.64 after gradient compression at a compression rate of 0.9, while our scheme can be reduced to about 0.10 for random guesses. The original federated performance is reduced, both down to roughly 0.7 or so.

### 4.5 Output distribution of malicious model

We use t-SNE [11] to map the output of bottom model A into 2D space. As shown in Figure 4, the classification of each color is less obvious. Because of our dispersed training, the bottom model A cannot learn about the relationship between labels and features and has a poor ability to perform label inference attacks. The attack model formed after model completion cannot perform label inference attacks.

## 5 Related work

At present, data security is a thorny issue, such as the malicious recovery of image data, the anonymity of network data transmission, privacy protection of distributed systems, data

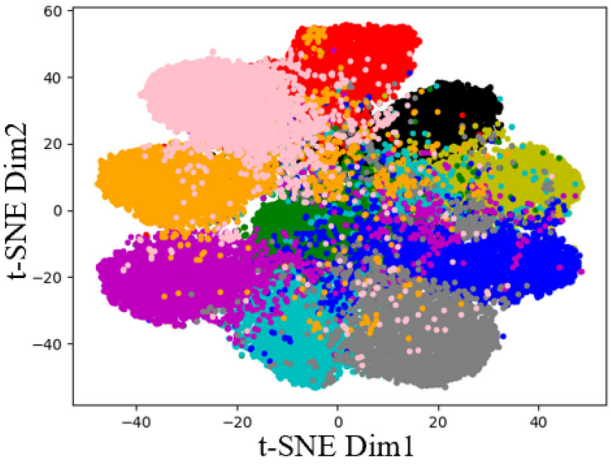


**Figure 4** The outputs of attack model *a*

privacy protection in big data environment [12], etc. Various solutions are proposed to solve these kinds of problems [13–19]. Zhang et al. [20] proposed an interesting approach for optimizing the multicast traffic based on the advantages of the software-defined networking. Among them, machine learning, especially federated learning, is a hot topic. Federated learning (FL) was first proposed by google [21–23], aiming at building machine learning models with distributed entities (e.g. devices or datasets), where the private information of the entities should be protected [13]. In general, FL has three categories, Horizontal Federated Learning [23–25], Vertical Federated Learning [26–30] and Federated Transfer Learning [31]. Federated Reinforcement Learning [32] has also recently emerged. Security issues [33–39], especially in vertical federated learning, has got arouse wide concern [40, 41]. Wei et al. [42] investigate the issues of security and privacy in VFL. Fu et al. [6] discuss the problem of label leakage instead of membership inference or sample property [43–46]. Rassouli et al. [47] prove it's possible for the adversary to reconstruct the passive party's feature under the black box. To protect privacy in VFL, Zhu et al. [48] propose a secure framework PIVODL and Han et al. [49] propose FedValue by using Shapley-CMI and guaranteeing the data privacy toward the view of game theory.

In general, differential privacy (DP) [50–53] and homomorphic encryption (HE) [54–57] are used to protect the privacy of the data in VFL. Geyer et al. [58] guarantee privacy of the users in the training process. Yuan et al. [59, 60] utilize HE to train data in the cloud. While DP and HE do not work here. First, if we only add some ransom salts to the training data, the training and the attack process are almost identical to the previous work [6]. Therefore, we add a convergence level, where the output and shared gradient converge. As for HE, it also fails in our setting. It's due to the fact that the bottom model can still be trained well even if the messages between the bottom and top model are encrypted with HE. Consequently, the attacks can be implemented by adding one level. Recently, secure multi-party computation (SMC) [61–64] is implemented to solve the privacy issues in VFL. For example, SecureML, an SMC framework is used to scalable preserve the privacy in machine learning. SMC can preserve privacy of sensitive data [65]. Mohassel et al. [66] propose a 3PC model by utilizing secret sharing with non-colluding servers. Personalized federated learning [67–69] is used to address data heterogeneity in federated learning. Secret sharing [70] preserves the information with respect to the intersection elements.

## 6 Conclusion

The previous work revealed that the privacy security of VFL also has great risks. Malicious participants in VFL can launch inference attacks on the labels of other participants, resulting in serious privacy leakage. To solve this problem, we propose a dispersed training framework, which introduces a new bottom model, which can share part of the gradient during the training of the malicious bottom model through secret sharing, so that the malicious bottom model cannot better obtain the relationship between labels and features, thereby preventing label inference attacks. Experiments show that dispersed training can effectively prevent label inference attacks. However, the accuracy of the original federal task is also affected to a certain extent, and it can only trade-off between raw federated task accuracy and attack accuracy, which provides a good direction for our future research. In the future, we can take into account the original federal performance and reduce the research on the direction of attack accuracy.

**Acknowledgements** This study is supported by the Foundation of National Natural Science Foundation of China (Grant No.: 62072273, 72111530206, 61962009, 61873117, 61832012, 61771231, 61771289); The Major Basic Research Project of Natural Science Foundation of Shandong Province of China (ZR2019ZD10); Natural Science Foundation of Shandong Province (ZR2019MF062); Shandong University Science and Technology Program Project (J18A326); Guangxi Key Laboratory of Cryptography and Information Security (No: GCIS202112); The Major Basic Research Project of Natural Science Foundation of Shandong Province of China (ZR2018ZC0438); Major Scientific and Technological Special Project of Guizhou Province (20183001), Foundation of Guizhou Provincial Key Laboratory of Public Big Data (No. 2019BD-KFJJ009), Talent project of Guizhou Big Data Academy. Guizhou Provincial Key Laboratory of Public Big Data. ([2018]01).

**Author Contributions** All authors contributed to the study conception and design. Yilei Wang put forward the main idea, Yilei Wang, Minghao Zhao and Qingzhe Lv wrote the main manuscript text, Qingzhe Lv, Huang Zhang and Yuhong Sun wrote the main experimental code, Tao Li revised the manuscript text, Lingkai Ran searched for the required literature. All authors reviewed the manuscript.

**Funding** This study is supported by the Foundation of National Natural Science Foundation of China (Grant No.: 62072273, 72111530206, 61962009, 61873117, 61832012, 61771231, 61771289); The Major Basic Research Project of Natural Science Foundation of Shandong Province of China (ZR2019ZD10); Natural Science Foundation of Shandong Province (ZR2019MF062); Shandong University Science and Technology Program Project (J18A326); Guangxi Key Laboratory of Cryptography and Information Security (No: GCIS202112); The Major Basic Research Project of Natural Science Foundation of Shandong Province of China (ZR2018ZC0438); Major Scientific and Technological Special Project of Guizhou Province (20183001), Foundation of Guizhou Provincial Key Laboratory of Public Big Data (No. 2019BD-KFJJ009), Talent project of Guizhou Big Data Academy. Guizhou Provincial Key Laboratory of Public Big Data. ([2018]01).

**Data Availability** The data used to support the findings of this study are available from the second author upon request.

## Declarations

**Ethical Approval and Consent to Participate** The authors guarantee that this manuscript is an original work. This manuscript has not been published or presented elsewhere in part or in entirety and is not under consideration by another journal. We have read and understood your journal's policies, and we believe that neither the manuscript nor the study violates any of these. All authors have seen and approved the final version of the submitted manuscript.

**Consent for Publication** All authors have checked the manuscript and have agreed to the submission

**Human and Animal Ethics** The authors declare that this study does not involve human participants or animals.

**Competing Interests** The authors have no competing interests to declare that are relevant to the content of this manuscript

## References

1. Voigt, P., Von dem Bussche, A.: The EU general data protection regulation (GDPR. A Practical Guide, 1st Ed., Cham: Springer International Publishing **10**(3152676), 10–5555 (2017)
2. Hoofnagle, C.J., van der Sloot, B., Borgesius, F.Z.: The european union general data protection regulation: what it is and what it means. Inform. Commun. Technol. Law **28**(1), 65–98 (2019)
3. Chik, W.B.: The singapore personal data protection act and an assessment of future trends in data privacy reform. Comput. Law Secur. Rev. **29**(5), 554–575 (2013)
4. Shatz, S., Chylik, S.E.: The california consumer privacy act of 2018: A sea change in the protection of california consumers. The Business Lawyer **75** (2020)
5. Hu, H., Salcic, Z., Sun, L., Dobbie, G., Yu, P.S., Zhang, X.: Membership inference attacks on machine learning: A survey. ACM Comput. Surv. (CSUR) **54**(11s), 1–37 (2022)

6. Fu, C., Zhang, X., Ji, S., Chen, J., Wu, J., Guo, S., Zhou, J., Liu, A.X. Wang, T.: Label inference attacks against vertical federated learning. In: 31st USENIX Security Symposium (USENIX Security 22), Boston, MA (2022)
7. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282, PMLR (2017)
8. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol. (TIST) **10**(2), 1–19 (2019)
9. Liu, Y., Kang, Y., Xing, C., Chen, T., Yang, Q.: A secure federated transfer learning framework. IEEE Intell. Syst. **35**(4), 70–82 (2020)
10. Vepakomma, P., Gupta, O., Swedish, T., Raskar, R.: Split learning for health: Distributed deep learning without sharing raw patient data, arXiv:1812.00564 (2018)
11. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. J. Mach. Learn. Res. **9**(11) (2008)
12. Yuan, F., Chen, S., Liang, K., Xu, L.: Research on the coordination mechanism of traditional Chinese medicine medical record data standardization and characteristic protection under big data environment, vol. 1 of *1*. No.517 Shungong Road, Shizhong District, Jinan, Shandong Province, China: Shandong:Shandong People's Publishing House, 1 ed., (2021)
13. Chen, C., Huang, T.: Camdar-adv: generating adversarial patches on 3d object. Int. J. Intell. Syst. **36**(3), 1441–1453 (2021)
14. Jiang, N., Jie, W., Li, J., Liu, X., Jin, D.: Gatrust: A multi-aspect graph attention network model for trust assessment in osns. IEEE Transactions on Knowledge and Data Engineering (2022)
15. Yan, H., Chen, M., Hu, L., Jia, C.: Secure video retrieval using image query on an untrusted cloud. Appl. Soft Comput. **97**, 106782 (2020)
16. Ai, S., Hong, S., Zheng, X., Wang, Y., Liu, X.: Csrt rumor spreading model based on complex network. Int. J. Intell. Syst. **36**(5), 1903–1913 (2021)
17. Li, T., Wang, Z., Chen, Y., Li, C., Jia, Y., Yang, Y.: Is semi-selfish mining available without being detected? Int. J. Intell. Syst. (2021). https://doi.org/10.1002/int.22656
18. Li, T., Wang, Z., Chen, Y., Li, C., Jia, Y., Yang, Y.: Is semi-selfish mining available without being detected?. International Journal of Intelligent Systems (2021)
19. Zhang, X., Wang, T.: Elastic and reliable bandwidth reservation based on distributed traffic monitoring and control. IEEE Transactions on Parallel and Distributed Systems (2022)
20. Zhang, X., Wang, Y., Geng, G., Yu, J., Delay-optimized multicast tree packing in software-defined networks. IEEE Transactions on Services Computing (2021)
21. Konečnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. arXiv:1610.02527 (2016)
22. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv:1610.05492 (2016)
23. McMahan, H.B., Moore, E., Ramage, D., y Arcas, B. A.: Federated learning of deep networks using model averaging. vol. 2, arXiv:1602.05629 (2016)
24. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1175–1191 (2017)
25. Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1310–1321 (2015)
26. Du, W., Atallah, M.J.: Privacy-preserving cooperative statistical analysis. In: Seventeenth Annual Computer Security Applications Conference, pp. 102–110. IEEE (2001)
27. Du, W., Han, Y.S., Chen, S.: Privacy-preserving multivariate statistical analysis: Linear regression and classification. In: Proceedings of the 2004 SIAM International Conference on Data Mining, pp. 222–233. SIAM (2004)
28. Sanil, A.P., Karr, A.F., Lin, X., Reiter, J.P.: Privacy preserving regression modelling via distributed computation. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 677–682 (2004)
29. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 639–644 (2002)
30. Wan, L., Ng, W.K., Han, S., Lee, V.C.: Privacy-preservation for gradient descent methods. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 775–783 (2007)
31. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowledge Data Eng **22**(10), 1345–1359 (2009)

32. Tianqing, Z., Zhou, W., Ye, D., Cheng, Z., Li, J.: Resource allocation in iot edge computing via concurrent federated reinforcement learning. IEEE Internet of Things Journal **9**(2), 1414–1426 (2021)
33. Hu, L., Yan, H., Li, L., Pan, Z., Liu, X., Zhang, Z.: Mhat: an efficient model-heterogenous aggregation training scheme for federated learning. Inform. Sci. **560**, 493–503 (2021)
34. Mo, K., Huang, T., Xiang, X.: Querying little is enough: Model inversion attack via latent information. In: International Conference on Machine Learning for Cyber Security, pp. 583–591. Springer (2020)
35. Ren, H., Huang, T., Yan, H.: Adversarial examples: attacks and defenses in the physical world. International Journal of Machine Learning and Cybernetics **12**(11), 3325–3336 (2021)
36. Kuang, X., Zhang, M., Li, H., Zhao, G., Cao, H., Wu, Z., Wang, X.: Deepwaf: detecting web attacks based on cnn and lstm models. In: International Symposium on Cyberspace Safety and Security, pp. 121–136. Springer (2019)
37. Yan, H., Hu, L., Xiang, X., Liu, Z., Yuan, X.: Ppcl: Privacy-preserving collaborative learning for mitigating indirect information leakage. Inform. Sci. **548**, 423–437 (2021)
38. Li, J., Hu, X., Xiong, P., Zhou, W., et al.: The dynamic privacy-preserving mechanisms for online dynamic social networks. IEEE Transactions on Knowledge and Data Engineering (2020)
39. Lu, Z., Liang, H., Zhao, M., Lv, Q., Liang, T., Wang, Y.: Label-only membership inference attacks on machine unlearning without dependence of posteriors. Int. J. Intell. Syst. **37**(11), 9242–9441 (2022)
40. Melis, L., Song, C., De Cristofaro, E., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 691–706. IEEE (2019)
41. Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning. In: Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), pp. 1–15 (2018)
42. Wei, K., Li, J., Ma, C., Ding, M., Wei, S., Wu, F., Chen, G., Ranbaduge, T.: Vertical federated learning: Challenges, methodologies and experiments. arXiv:2202.04309 (2022)
43. Backes, M., Berrang, P., Humbert, M., Manoharan, P.: Membership privacy in microrna-based studies. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 319–330, 2016
44. Chen, D., Yu, N., Zhang, Y., Fritz, M.: Gan-leaks: A taxonomy of membership inference attacks against gans. arXiv:1909.03935 (2019)
45. Pyrgelis, A., Troncoso, C., De Cristofaro, E.: Knock knock, who's there? membership inference on aggregate location data, arXiv:1708.06145 (2017)
46. Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., Backes, M.: Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. arXiv:1806.01246 (2018)
47. Rassouli, B., Varasteh, M., Gunduz, D.: Privacy against inference attacks in vertical federated learning. arXiv:2207.11788 (2022)
48. Zhu, H., Wang, R., Jin, Y., Liang, K.: Pivodl: Privacy-preserving vertical federated learning over distributed labels. IEEE Transactions on Artificial Intelligence (2021)
49. Han, X., Wang, L., Wu, J.: Data valuation for vertical federated learning: An information-theoretic approach. arXiv:2112.08364 (2021)
50. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 308–318 (2016)
51. Chaudhuri, K., Monteleoni, C.: Privacy-preserving logistic regression. Advances in Neural Information Processing Systems, **21** (2008)
52. Dwork, C.: Differential privacy: A survey of results. In: International Conference on Theory and Applications of Models of Computation, pp. 1–19. Springer (2008)
53. Song, S., Chaudhuri, K., Sarwate, A.D.: Stochastic gradient descent with differentially private updates. In: 2013 IEEE Global Conference on Signal and Information Processing, pp. 245–248. IEEE (2013)
54. Giacomelli, I., Jha, S. Joye, M., Page, C.D., Yoon, K.: Privacy-preserving ridge regression with only linearly-homomorphic encryption. Cryptology ePrint Archive (2017)
55. Hall, R., Fienberg, S.E., Nardi, Y.: Secure multiple linear regression based on homomorphic encryption. Journal of Official Statistics **27**(4), 669–691 (2011)
56. Nikolaenko, V., Weinsberg, U., Ioannidis, S., Joye, M., Boneh, D., Taft, N.: Privacy-preserving ridge regression on hundreds of millions of records. In: 2013 IEEE Symposium on Security and Privacy, pp. 334–348. IEEE (2013)
57. Rivest, R.L., Adleman, L., Dertouzos, M.L., et al.: On data banks and privacy homomorphisms. Foundations of Secure Computation **4**(11), 169–180 (1978)
58. Geyer, R.C., Klein, T., Nabi, M.: Differentially private federated learning: A client level perspective. arXiv:1712.07557 (2017)
59. Yuan, J., Yu, S.: Privacy preserving back-propagation neural network learning made practical with cloud computing. IEEE Transactions on Parallel & Distributed Systems **25**(01), 212–221 (2014)

60. Zhang, Q., Yang, L.T., Chen, Z.: Privacy preserving deep computation model on cloud for big data feature learning. IEEE Trans. Comput. **65**(5), 1351–1362 (2015)
61. Araki, T., Furukawa, J., Lindell, Y., Nof, A., Ohara, K.: High-throughput semi-honest secure three-party computation with an honest majority. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 805–817 (2016)
62. Furukawa, J., Lindell, Y., Nof, A., Weinstein, O.: High-throughput secure three-party computation for malicious adversaries and an honest majority. Cryptology ePrint Archive (2016)
63. Mohassel, P., Rosulek, M., Zhang, Y.: Fast and secure three-party computation: The garbled circuit approach. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 591–602 (2015)
64. Zhao, C., Zhao, S., Zhao, M., Chen, Z., Gao, C.-Z., Li, H., Tan, Y.-a.: Secure multi-party computation: theory, practice and applications. Inform. Sci. **476**, 357–372 (2019)
65. Kilbertus, N., Gascón, A., Kusner, M., Veale, M., Gummadi, K., Weller, A.: Blind justice: Fairness with encrypted sensitive attributes. In: International Conference on Machine Learning, pp. 2630–2639, PMLR (2018)
66. Mohassel, P., Rindal, P.: Aby3: A mixed protocol framework for machine learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 35–52 (2018)
67. Arivazhagan, M.G., Aggarwal, V., Singh, A.K., Choudhary, S.: Federated learning with personalization layers. arXiv:1912.00818 (2019)
68. Jebreel, N.M., Domingo-Ferrer, J., Blanco-Justicia, A., Sanchez, D.: Enhanced security and privacy via fragmented federated learning. arXiv:2207.05978 (2022)
69. Li, Q., He, B., Song, D.: Model-contrastive federated learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10713–10722 (2021)
70. Le, P.H., Ranellucci, S., Gordon, S.D.: Two-party private set intersection with an untrusted third party. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 2403–2420 (2019)

## Authors and Affiliations

**Yilei Wang[1,2] · Qingzhe Lv[1] · Huang Zhang[2] · Minghao Zhao[3] · Yuhong Sun[1] · Lingkai Ran[1] · Tao Li[1,4]**

Yilei Wang
wang_yilei2019@qfnu.edu.cn

Qingzhe Lv
lvqingzhe2021@163.com

Huang Zhang
zhanghuang_gz@163.com

Minghao Zhao
zhaominghao.thu@gmail.com

Yuhong Sun
sun_yuh@163.com

Lingkai Ran
rlk0729@163.com

1 School of Computer Science, Qufu Normal University, 276800 Rizhao, Shandong, China

2 Institute of Artificial Intelligence and Blockchain, Guangzhou University, 510700 Guangzhou, Guangdong, China

3 School of Data Science and Engeneering, East China Normal University, 200062 Shanghai, China

4 State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, 550025 Guiyang, Guizhou, China