A Privacy-Preserving Hybrid Federated Learning Framework for Financial Crime Detection

Haobo Zhang
Michigan State University
East Lansing, Michigan, USA
zhan2060@msu.edu

Junyuan Hong
Michigan State University
East Lansing, Michigan, USA
hongju12@msu.edu

Fan Dong
University of Calgary
Calgary, Alberta, Canada
fan.dong@ucalgary.ca

Steve Drew
University of Calgary
Calgary, Alberta, Canada
steve.drew@ucalgary.ca

Liangjie Xue
Coinbase Global, Inc.
USA
liangjie.xue@gmail.com

Jiayu Zhou
Michigan State University
East Lansing, Michigan, USA
jiayuz@msu.edu

ABSTRACT

The recent decade witnessed a surge of increase in financial crimes across the public and private sectors, with an average cost of scams of $102m to financial institutions in 2022. Developing a mechanism for battling financial crimes is an impending task that requires in-depth collaboration from multiple institutions, and yet such collaboration imposed significant technical challenges due to the privacy and security requirements of distributed financial data. For example, consider the modern payment network systems, which can generate millions of transactions per day across a large number of global institutions. Training a detection model of fraudulent transactions requires not only secured transactions but also the private account activities of those involved in each transaction from corresponding bank systems. The distributed nature of both samples and features prevents most existing learning systems from being directly adopted to handle the data mining task. In this paper, we collectively address these challenges by proposing a hybrid federated learning system that offers secure and privacy-aware learning and inference for financial crime detection. We conduct extensive empirical studies to evaluate the proposed framework's feasibility and scalability. The codes will be released in the future.

ACM Reference Format:
Haobo Zhang, Junyuan Hong, Fan Dong, Steve Drew, Liangjie Xue, and Jiayu Zhou. 2023. A Privacy-Preserving Hybrid Federated Learning Framework for Financial Crime Detection. In *KDD FL4Data-Mining '23, August 7, 2023, Long Beach, CA, USA.*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Over the past decade, financial crime has been on the rise, causing significant harm to industries and hindering innovation. Efforts to combat these crimes have led to the establishment of infrastructures such as the U.S. Department of the Treasury's Financial

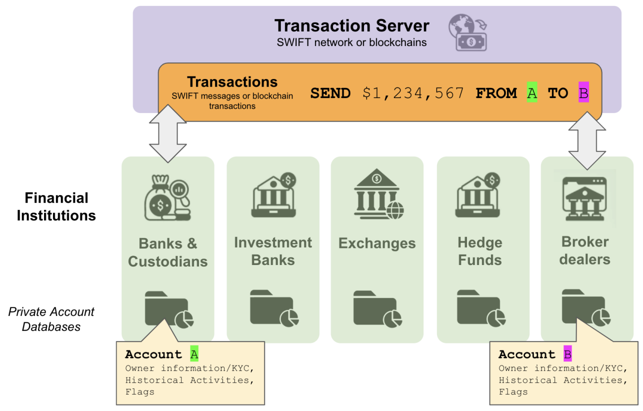

Transaction Server
SWIFT network or blockchains
Transactions
SWIFT messages or blockchain transactions
SEND $1,234,567 FROM A TO B
Financial Institutions
Banks & Custodians
Investment Banks
Exchanges
Hedge Funds
Broker dealers
Private Account Databases
Account A
Owner information/KYC, Historical Activities, Flags
Account B
Owner information/KYC, Historical Activities, Flags

Figure 1: Financial crime detection needs collaborative efforts from multiple financial institutions. A typical transaction moves funds between two accounts in two financial institutions. To determine if a transaction is fraudulent or not, we rely on a collective analysis of the information associated with the transaction and information of the source and target accounts at different institutions. Building models require learning and inference across private and distributed databases in different institutions, which imposes significant challenges and demands a novel learning paradigm.

Crime Enforcement Network (FinCen) [19]. However, criminals have continuously evolved their techniques, making it harder to detect financial crimes using conventional methods.

More recently, financial institutions have sought to use artificial intelligence and machine learning to analyze real-time data to identify financial crimes. Although AI and ML have shown to be very promising [20], the AI system needs to access a variety of data, including samples and features, to fully leverage the powerful modeling capability.

As such, data-driven detection models of financial crimes require the collaboration of multiple financial institutions during both the training and inference stages. For instance, in the case of detecting fraudulent wire transactions in payment network systems, the analysis requires the joint examination of transaction information from the server, account activities of the source account from the corresponding bank, and activities of the target account from another bank.

Both transactions and bank account activities are very sensitive data owned by different financial institutions, and learning a model requires privacy-aware collaborative learning. With the increasing demand and law regulation (e.g., General Data Protection Regulation) for learning from distributed data without directly sharing them, the federated learning paradigm has been developed and attracted great efforts from the data mining and machine learning community. The most common type is horizontal federated learning, in each iteration of which a central server retrieves updated models from participating clients using partial samples and then sends the aggregated model to them for iterative updates. Vertical federated learning, on the other hand, handles the setting where clients have different parts or features of the same set of samples. However, the distributed and private nature of *both samples and features* prevents most existing learning systems from being directly adopted to handle the financial crime prediction task. Besides the learning stage, the inference stage of the model also needs private information from distributed parties.

In this paper, we develop a novel and holistic privacy-preserving approach for financial crime detection from distributed and private financial information. To achieve this goal, the proposed solution leverages a hybrid of vertical and horizontal federated learning of the *transaction client* and *account clients*. From the vertical perspective, the *transaction client* (e.g., the server) will share extracted features with *account clients* (e.g., banks) for collaborative prediction; from the horizontal perspective, the bank units will collaboratively train a unified encoder to extract features from private account information. We combine them by fusing the features from the two involved parties and leveraging new features to train a predictive model in the transaction client. The proposed solution considers comprehensive privacy risks and associated attacks, including model inversion [9], privacy attribute inference [15], membership inference [21], and feature leakage. To protect against these risks, the proposed system combines strategies including noise injection, local feature extraction, and encryption technique.

Further results support the feasibility of our framework to collaboratively train a fraud-detection model while preserving data privacy. As such, it provides a powerful financial crime detection tool to securely and privately conduct collaborative data analysis from multiple financial institutions. Also, the proposed collaborative framework provides a powerful tool for law enforcement to conduct financial crime detection while protecting the privacy of participating financial institutions.

Real-world Impacts. As financial crimes cost hundreds of millions per year, financial fraud detection is a real-world challenge whose solution is urgently needed in financial companies. Further, fraud detection needs data from different financial institutions, which is sensitive and private. Thus, we highlight the scientific contribution of our work as a holistic framework for privacy-preserving collaboratively training of hybrid federated learning, as well as an applicable paradigm to solve financial fraud detection as a critical real-world challenge.

2 BACKGROUND

Financial crime detection. Detecting financial crime has been an ongoing challenge, where the goal is to identify abnormal transactions, including fraud transactions, anomalous payments, and money laundering [16, 17, 20]. As money transfer transactions involve multiple parties, their sophisticated patterns of evolving financial crimes demand joint data analysis from multiple financial institutions. To develop a data-driven financial crime detection system, a machine learning model has to combine the features from the two systems in order to predict crime behaviors precisely. However, data records in the transaction client and banks are located distributedly and cannot be shared due to their sensitive nature. As such, it is challenging to combine the information sources.

Distributed federated learning and its challenges. To enable distributed machine learning that conforms to data privacy, federated learning (FL) does not require data to be directly shared but instead trains a model collaboratively from distributed clients (data sources). The main idea of federated learning is to aggregate knowledge without sharing raw data from multiple clients through locally-trained models [28]. Depending on the status of feature and data splitting, FL can be further categorized into two classes: vertical and horizontal federated learning [27, 30]. Vertical federated learning learns a model that predicts targets based on features concatenated from different clients with synchronized sample indexes. On the other hand, horizontal federated learning maintains the same feature space while collecting non-overlapped data from clients.

Existing FL frameworks cannot be directly adopted in the collaborative learning task of financial crime detection due to their hybrid nature: the vertical relation between the *account clients* and *transaction clients* and the co-existing horizontal relation between different banks. Account clients and transaction clients need to share features for prediction, but the corresponding account client is joined based on the account identification (e.g., an account number or a public key address in blockchains) in transactions. Meanwhile, account clients have to share knowledge such that a more powerful model can be trained.

Privacy risks in federated crime detection and mitigation. Like other FL frameworks, there are non-neglectable privacy risks in hybrid FL on data sharing and model publishing. We consider the following four types of privacy risks in this paper: *Model inversion:* model inversion leaks information reversing the final trained model [9]. By setting up an appropriate loss, optimization-based methods can be used to estimate the original input. *Attribute inference:* the correlations between features can be leaked via attribute inference with the support of some accessible features like some public data [13, 15]. *Membership inference:* while one can infer whether or not a given data point is present in the training set using membership inference [14, 21]. *Feature leakage:* extracted features can also be leaked from untrusted units to third parties, i.e., the feature leakage risk.

There have been three main-stream strategies to defend against privacy risks. *Differential privacy (DP)* adds Gaussian noise or Laplacian noise to the gradients during the training process [1, 23] to defend various attacks. Then the differential privacy property guarantees that it is difficult to distinguish two adjacent datasets. *Homomorphic encryption (HE)* ensures that users can perform computation on encrypted data without first decrypting it [2, 8], with which two nodes can operate the encrypted data without leaking the original data to the third party. *Adversarial privacy-disentanglement (ADV)* aims to adversarially remove the known privacy attribute from the learned representations while preserving the utility attributes [25, 26]. However, DP and ADV lead to the degradation of model performance, while HE suffers from high computation costs and limited computation operation.

3 PROPOSED METHOD

To address the challenges in the previous section, we hereby propose a novel *Hybrid Federated Learning* (HyFL) framework that integrates the vertical and horizontal frameworks smoothly.

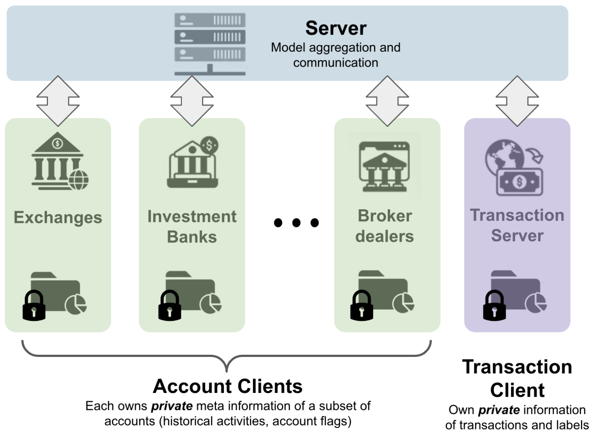

Server
Model aggregation and communication
Exchanges
Investment Banks
Broker dealers
Transaction Server
Account Clients
Each owns *private* meta information of a subset of accounts (historical activities, account flags)
Transaction Client
Own *private* information of transactions and labels

Figure 2: The federated setup and three key participants of the proposed HyFL framework: 1) A *server* responsible for information aggregation; 2) *account clients*: a set of financial institutions owning meta information of accounts; 3) A *transaction client* that owns private transactions and their labels (abnormal or not).

For simplicity of discussion without the loss of generality, we assume that one **transaction client** (Tx client) has all *private* transaction data, and the *private* account metainformation is stored in a set of **account clients** (Ac clients) separately. During the training phase, the label information (whether or not a transaction is marked as fraud) is stored in the transaction client with an option to be stored in the aggregation server as well. In that case, since the data in account clients share the same dimensions of features, they can be viewed as the clients in a horizontal FL setting. Compared to the transaction client, account clients have different features even for the same transaction. As a result, a vertical FL setting will apply between the account clients and transaction clients. We illustrate the federated setup and key components in Figure 2.

Notations. We now introduce the notations to be used in this paper. Formally, consider a set of $M + 1$ clients $\{1, 2, ..., M + 1\}$ with M account clients and one transaction client. Define a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$. In the vertical FL setting, each client m has a subset of the features from $\mathcal{D}$, i.e., $\mathcal{D}_m = \{\mathbf{x}_{i,m}\}_{i=1}^N$, where $\mathbf{x}_{i,m}$ contains the features of $\mathbf{x}_i$ stored in the client m. The label set $\{y_n\}_{i=1}^N$ can be stored on either a client or the server. In the horizontal FL setting, each client m has a partition of the dataset, i.e., $\mathcal{D}_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_m}$, where N_m is the size of data stored in client m. The label set is stored with the data on each client in horizontal settings.

3.1 Framework Overview

In this section, we give an overview of our proposed framework. First, we define three types of computation nodes, based on which we propose our hybrid federated learning framework. Then we introduce three phases in the training stage, where an auto-encoder is trained on the account data and used to extract the feature embedding. Then a classifier is trained in the transaction client to detect potential financial crime.

Computation Nodes and Learning Framework. In the proposed system, we coordinate two types of computation nodes and a server for hybrid federated learning.

(1) The **transaction client** consists of a set of features describing the transactions for the identification of anomalies in our data, as well as the labels indicating whether a transaction involves a crime or not. With the information, the transaction client is able to train a classifier locally and independently and store the classifier parameters locally.
(2) The **account clients** maintain account activities with their corresponding meta-information (e.g., account flags). The account information provides complementary information for detecting anomalies, as many criminal behaviors are related to historical account activities. We use an autoencoder as a feature extractor of flags in account clients.
(3) A computation **server** aggregates the feature extractors from the account clients and sends the feature embedding from the account clients to the transaction client.

Three Phases in the Training Stage. The proposed learning framework has three phases in the training stage:

(1) (**Feature learning**) First, each account client trains a model to extract the feature embedding of the stored accounts data. Since there is no label stored in an account client, an auto-encoder is trained in a self-supervised fashion. Define the encoder and decoder as $h(\cdot)$ and $h'(\cdot)$. With a global initialization of h_ϕ, h'_ψ, each client will train the models locally by
$$\phi_m, \phi'_m = \arg\min_{\phi,\phi'} \sum_{i=1}^{N_m} \left\| h'_{\phi'}(h_\phi(\mathbf{x}_i)) - \mathbf{x}_i \right\|_2^2. \quad (1)$$
When the number of account clients is very large, federated learning of the model has a large communication overhead. To reduce the overhead, the parameters of auto-encoders are aggregated in the server only once after the training in each account client:
$$\phi = \frac{1}{M} \sum_{m=1}^{M} \phi_m, \ \phi' = \frac{1}{M} \sum_{m=1}^{M} \phi'_m. \quad (2)$$
Then the aggregated auto-encoder will be broadcast back to all the account clients.
(2) (**Extract features**) Second, the account clients use the trained auto-encoder to extract the feature embedding of their local account data as $\mathcal{F}_m = \{h_\phi(\mathbf{x}_i)\}_{i=1}^{N_m}$. Then, the transaction client

will request the corresponding feature embeddings by querying account numbers.

(3) (**Classifier training**) On receiving the embeddings, the transaction client will ensemble the account features with the transaction features as $\{([\mathbf{x}_{i,m_1}, \mathbf{x}_{i,m_2}, \mathbf{x}_{i,M}], y_i)\}$, where $\mathbf{x}_{i,m_1}$, $\mathbf{x}_{i,m_2}$ are the two account embeddings associated with the transactions. To protect data privacy from model inversion and membership inference, Gaussian noise is added to the concatenated features. A high-capacity model such as XGBoost [6] or neural networks (denoted as $f(\cdot)$) is then trained for predicting y_i, i.e., $f([\mathbf{x}_{i,m_1}, \mathbf{x}_{i,m_2}, \mathbf{x}_{i,M}])$.

Although our framework follows the client-server paradigm, where a dedicated server communicates with all clients, our framework can also work in a peer-to-peer (P2P) paradigm without a server: feature embeddings can be directly sent from the account clients to the transaction client.

3.2 Threat Model and Privacy Risk Sources

The successful design of privacy-awareness machine learning systems depends on a realistic threat model and analysis of possible risk sources. Based on our proposed HyFL framework, we elaborate on the threat models. We first introduce four types of possible attacks. In the threat model, the privacy risk arises from the communications between any two nodes which could be of either the same or different types. We also discuss the privacy risks at the inference stage in addition to the communication during training.

Privacy Risk Sources. The privacy risks of the proposed framework are four-fold. First, *gradient inversion* targets recovering input data from a trained model. Sensitive information, such as data points, may be recovered from model parameters or gradients via this attack. Another attack on the final trained model is the *membership inference*, which infers a specific data point in the training set by evaluating the difference between the distribution of training data and test data. By stealing the latent correlations among features, *attribute inference* can infer private information. There are also risks from unreliable units in the framework, where the *feature leakage* can happen to extracted features. We give more formal definition of these privacy risk sources in appendix A.

Threat Model. A majority of FL systems focus on the *honest but curious* nodes, i.e., one type of node will try to recover the data from the other two types of nodes, but it will not modify the model or features [12]. In this work, we consider a comprehensive threat model by further assuming that the nodes of different types are *potentially* honest but curious. Benefit from our framework, there is no direct communication between the transaction client and the server, so we can only investigate potential privacy risk sources from communications between two combinations of nodes:

(1) During the communication between the aggregation server and account clients, model updates from account clients are sent to the server, which leads to direct gradient inversion risk as well as the membership inference. Also, optimization-based methods can be utilized to estimate private attributes with the support of some possible public data, which is the risk of attribute inference.

(2) During the communication between the account clients and the transaction client, features from the account clients will be sent to the server, which raises the problem of feature leakage. Since the transaction has no access to the account clients' model, there is no risk of either model inference or attribute inference. However, with the features from the account clients, membership can be inferred by the transaction client.

(3) The final risk lies in the transaction client. In the inference stage, attackers can utilize the predictive model in the transaction client to complete membership inference or attribute inference. We introduce the details of privacy risk in the inference stage in appendix A.

Solutions. Targeting the four privacy risk sources, we propose to adopt various defense mechanisms in our framework to protect privacy. (1) To defend the model inversion and membership inference attack, we add Gaussian noise to the data in the transaction clients before training the classifier, which confuses the inversion or inference attackers from the real features. (2) To defend the attribute inference attack, we keep the encoder model only to account clients and away from the transaction client such that the transaction client cannot recover the sensitive attribute. (3) We also leverage encryption during the communication so that the server cannot recover the original features from model parameters.

4 SYSTEM IMPLEMENTATION

In this section, we describe the technical details of our novel Hybrid Federated Learning (HyFL) framework. We introduce the training and inference stages, approaches to preserve privacy during the two stages, and techniques to increase the scalability of our framework and make it feasible. We first introduce a vanilla framework as the backbone of our framework. Then we enhance the vanilla framework with privacy-preserving approaches.

Algorithm 1 Vanilla HyFL Framework without Privacy Protection.

```
Input: Transaction features without account meta information in the transaction (Tx) client
Input: Accounts associated with account meta information in account (Ac) clients
  Ac clients locally train $\phi_m, \phi'_m$ with Eq. (1)
  Server aggregates $\phi_m, \phi'_m$ with Eq. (2) and returns the extractors to Ac clients
  while Training or Inference do
      Tx client sends accounts to account clients
      Ac clients (1) query and extract features according to the accounts, (2) send extracted features to transaction client
      Tx client receives the extracted features and makes a prediction.
```

At the beginning of our framework, the account clients train their feature extractors locally. Then the server collects and aggregates these trained extractors and sends the aggregated extractor back to the account clients. Since a portion of the features and the labels are stored in the transaction client, both the training and inference stages start from the transaction client. We can either directly use the original features or use a feature extractor to extract features, depending on the size of the features used. If the feature size is small, then we can work with the original feature space. Otherwise, if high-dimensional data such as text and images are used, then we recommend using a feature extractor for dimension reduction.

Next, the original or extracted features will be sent to the server. The transaction client will use account identifiers to query for the corresponding features stored in the account clients. When an account client receives a query from the transaction client for a specific account, it searches for the corresponding account features (e.g., flags) and uses a feature extraction model to generate the intermediate features. Then the extracted features will be sent back to the transaction client to predict the final label. Due to the lack of labels in account clients, we propose to use an unsupervised model as our feature extractor, and the autoencoder [3, 24] is an ideal instantiation. Our final implementation uses XGBoost [6] as the classifier in the transaction client for its strong performance in classification tasks. This concludes the backbone of our framework, which we call the vanilla HyFL framework. The framework of training and inference stages is summarized in Algorithm 1 in the appendix. Note that in the vanilla HyFL framework, the server only has the model of feature extractors, but no features from the account clients. On the other hand, the transaction client only has the extracted features but not the encoders, so it cannot recover the data from the features only.

Privacy Preserving Enhancement. To tackle the three privacy risk sources, we enhance our framework through the lens of several privacy-preserving methods, including differential privacy mechanisms, feature sharing, and homomorphic encryption. The risk sources are first analyzed for each communication, then targeted solutions are identified correspondingly.

(1) The first risk source is the model sharing from account clients to the server, which induces the risk of inference attacks and inversion attacks. To address this issue, we adopt encryption to encrypt the model parameters before sending them to the server. Without the private key, the server cannot decrypt the data but can still aggregate the encrypted parameters.
(2) Another risk source is the feature sharing from the account clients to the transaction client. To alleviate the risk of attribute inference attacks and feature leakage, instead of directly sending the original account meta information themselves, a feature extractor is adopted by the account clients. Without access to the model in the account clients, the transaction client cannot recover the original account data with only the extracted features.
(3) The final risk source is that in the inference stage, the predictive model in the transaction client may be used to infer the membership or private attribute. To mitigate this problem, we add privacy-protecting Gaussian noise to the data in the transaction client before training the classifier. Note that our data is in tabular form. In that case, unlike image data, some little perturbation can lead to fidelity. Normalization to the standard normal distribution is utilized as well before the noise injection, which changes the distribution of original data, making it even more difficult to recover the data for the attackers.

Strengthening our vanilla HyFL framework with the above privacy-preserving enhancement and feasible scalability improvement, we illustrate the final HyFL framework in Algorithm 2. The key difference versus Algorithm 1 is the encryption and noise mechanism for privacy protection.

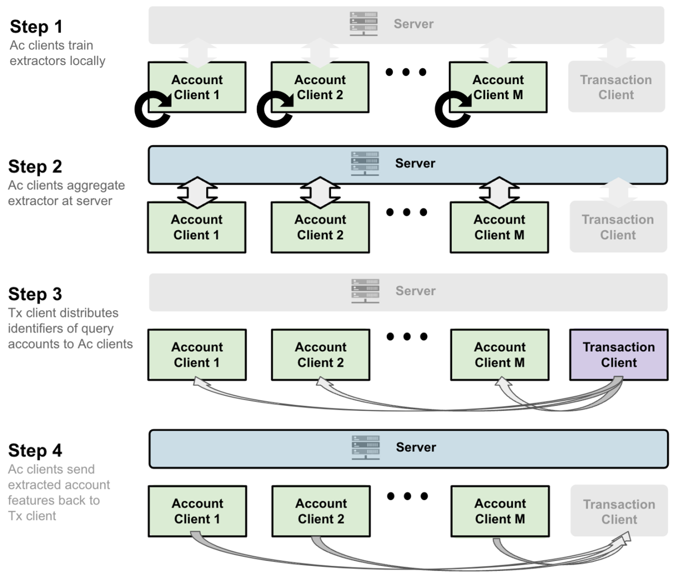

Step 1
Ac clients train extractors locally
Server
Account Client 1
Account Client 2
Account Client M
Transaction Client
Step 2
Ac clients aggregate extractor at server
Step 3
Tx client distributes identifiers of query accounts to Ac clients
Step 4
Ac clients send extracted account features back to Tx client

Figure 3: Communication protocol and information flow between the clients and the server in our proposed HyFL. (1) The account clients train the feature extractors locally. (2) The account clients send the extractors to the models and receive the aggregated model from the server. (3) The transaction client requests account meta information from the account clients by sending account identifiers. (4) The account clients send the features back to the transaction client.

Algorithm 2 The proposed HyFL framework.

```
Input: Transaction features without account meta information in the transaction (Tx) client
Input: Account meta information in account (Ac) clients
  Ac clients (1) locally train $\phi_m, \phi'_m$ with Eq. 1 and send to the server
  Server aggregates $\phi_m, \phi'_m$ with Eq. 2 and returns to the Ac clients
  while Training do
    Tx client sends account identifiers to the Ac clients
    Ac clients query and encrypt features according to the accounts and send them to the Tx client
    Tx client (1) receives and decrypts the features, (2) adds Gaussian noise into the aggregated features, (3) trains a classifier $f$ with aggregated features
  while Inference do
    Tx client sends account identifiers to Ac clients
    Ac clients query and encrypt features according to the accounts identifiers and send to the Tx client
    Tx client (1) receives and decrypts the features; (2) predicts the label $\hat{y}$ with $f$.
    Tx client Outputs the prediction label $\hat{y}$
```

5 EXPERIMENTAL RESULTS

5.1 Experiment Setup

We use the synthetic dataset provided by SWIFT. Specifically, the size of the training set is approximately three million, with the proportion of positive and negative samples being 1 : 100. The size of the test set is one-quarter of the training set. The data we use is in the form of tables with a limited number of features but of large size, which is suitable for the model training and queries across different nodes in our framework. Since we have billions of samples but a limited number of features, it yields a high risk of overfitting if we use a neural network as the classifier, which is backed up by our conducted experiments. Yet this kind of dataset provides a unique opportunity for the model such as XGBoost.

5.2 Comparison with Baselines

In this section, we compare our framework with the baselines, to evaluate its effectiveness. We set 0.01 as a reasonable noise variance according to the Fig. 6. Since we have both vertical and horizontal settings in our framework, current frameworks cannot be utilized to evaluate our proposal. Thus, we consider the following two settings as our baselines: (1) Centralized setting where all the account data and transaction data are stored in a single node.(2) Vanilla HyFL as in Alg 1 where the setting is the same as our proposed framework but without privacy enhancement.

To conduct a comprehensive evaluation, we compare our framework with four typical models for binary classification tasks: XGBoost present for tree models, Support Vector Machine (SVM) [5] present for linear models, Logistic Regression (LR) [18] present for simple non-linear models, and multi-layer perceptron (MLP) [10] present for deep non-linear models.

The results for three settings are shown in Table 1. The performance of SVM is the worst across the three settings, which implies that the linear classification model is not suitable for such a task with tabular data, because such data is not linearly separable. Similarly, although LR has some non-linear properties from the sigmoid function, it still cannot achieve satisfactory performance due to the linear combination of features before the sigmoid function. On the other hand, MLP shows great improvement with the support of strong power to extract useful features from raw data. Surprisingly, XGBoost achieves the best performance across all three settings. One possible reason is that as the number of estimators in XGBoost increases, the complex relationship between features can be extracted. In that case, XGBoost can also construct a strong non-linear map from features to the predicted label just like MLP.

From the perspective of different frameworks, we can observe that the vanilla setting has a similar performance to the centralized setting as shown in Table 1. Also, we find HyFL with privacy enhancement does not harm the performance too much and it still has a reliable utility with a high AUCPR.

5.3 Sensitivity Study

5.3.1 Impact of Account Client Number. In this section, we present the impact of the number of account clients on the model performance. We only change the account client number and maintain the other settings in this section. Based on the account client number,

	XGBoost	SVM	LR	MLP
	Centralized setting			
Precision	0.97	0.50	0.97	0.99
Recall	0.79	0.46	0.61	0.67
F1	0.86	0.39	0.68	0.76
AUCPR	0.7037	0.0011	0.2976	0.5608
	Vanilla HyFL			
Precision	0.98	0.50	0.97	0.98
Recall	0.79	0.37	0.61	0.71
F1	0.86	0.13	0.68	0.79
AUCPR	0.7075	0.0009	0.2977	0.5392
	HyFL			
Precision	0.98	0.50	0.97	0.95
Recall	0.76	0.37	0.61	0.70
F1	0.83	0.13	0.68	0.78
AUCPR	0.6839	0.0009	0.2975	0.5438

Table 1: Performance of different classifiers under three different settings.

	1	10	50	100	200
Precision	0.97	0.97	0.97	0.97	0.97
Recall	0.79	0.79	0.71	0.71	0.71
F1	0.86	0.86	0.79	0.79	0.79
AUCPR	0.7037	0.7037	0.5344	0.5344	0.5344

Table 2: Impact of account client number on the performance.

	0.5	0.1	0.01	0.002
Precision	0.97	0.94	0.97	0.59
Recall	0.79	0.75	0.71	0.51
F1	0.86	0.82	0.79	0.51
AUCPR	0.7037	0.6068	0.5344	0.0534

Table 3: Impact on the model performance of the data size.

the account data is split randomly and used to train an auto-encoder locally in each account client. Then the server aggregates all the auto-encoders from the account clients as the global auto-encoder. The results are summarized in Table 2. As the number of account clients increases, the model performance tends to be stable, which implies that our framework is robust even with a large number of account clients. Yet the model can achieve a better performance when the account client number is small, we can still obtain utility as the account client number is large.

5.3.2 Impact of Data Size. In this section, we study the impact of the size of the training set. We randomly sample the training data from the original training set, based on the sampling ratio, where we maintain the relative proportion between positive and negative samples. From Table 3 we can see the model performance is destroyed by the decreasing data size. Note the number of positive samples is quite limited in our case. Hence, as we decrease the data size, the number of positive samples is getting smaller, which makes it even more difficult to learn from the positive samples.

6 CONCLUSION

In this paper, we tackled the challenges of the collaboration of multiple financial institutions for financial crime detection. We showed that existing FL paradigms could not be directly applied to solve this detection problem because of the way private data and features are distributed. To address the challenges, we proposed a hybrid FL framework that allows sharing features and training the model in a privacy-preserving way. Empirical results showed the robustness and effectiveness of the framework under different scenarios. We believe the proposed solution can greatly transform the landscape of the financial system by better safeguarding them with a powerful financial crime detection tool.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. 2018. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)* 51, 4 (2018), 1–35.

[3] Dor Bank, Noam Koenigstein, and Raja Giryes. 2020. Autoencoders. *arXiv preprint arXiv:2003.05991* (2020).

[4] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. 2022. Flower: A friendly federated learning framework. (2022).

[5] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*. 144–152.

[6] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[7] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948* (2020).

[8] Caroline Fontaine and Fabien Galand. 2007. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security* 2007 (2007), 1–10.

[9] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 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.

[10] Matt W Gardner and SR Dorling. 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment* 32, 14-15 (1998), 2627–2636.

[11] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems* 33 (2020), 16937–16947.

[12] Oded Goldreich. 2004. Foundations of cryptography. II: Basic applications. 2 (05 2004). https://doi.org/10.1017/CBO9780511721656

[13] Neil Zhenqiang Gong and Bin Liu. 2016. You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors. In *25th USENIX Security Symposium (USENIX Security 16)*. 979–995.

[14] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2021. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)* (2021).

[15] Jinyuan Jia and Neil Zhenqiang Gong. 2018. {AttriGuard}: A practical defense against attribute inference attacks via adversarial machine learning. In *27th USENIX Security Symposium (USENIX Security 18)*. 513–529.

[16] Eren Kurshan and Hongda Shen. 2020. Graph computing for financial crime and fraud detection: Trends, challenges and outlook. *International Journal of Semantic Computing* 14, 04 (2020), 565–589.

[17] Eren Kurshan, Hongda Shen, and Haojie Yu. 2020. Financial crime & fraud detection using graph computing: Application considerations & outlook. In *2020 Second International Conference on Transdisciplinary AI (TransAI)*. IEEE, 125–130.

[18] Michael P LaValley. 2008. Logistic regression. *Circulation* 117, 18 (2008), 2395–2399.

[19] Financial Crimes Enforcement Network. 2023. What We Do. https://www.fincen.gov/what-we-do. https://www.fincen.gov/what-we-do

[20] Zeinab Rouhollahi. 2021. Towards artificial intelligence enabled financial crime detection. *arXiv preprint arXiv:2105.10866* (2021).

[21] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.

[22] Canh T Dinh, Nguyen Tran, and Josh Nguyen. 2020. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems* 33 (2020), 21394–21405.

[23] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*. 61–66.

[24] Wei Wang, Yan Huang, Yizhou Wang, and Liang Wang. 2014. Generalized autoencoder: A neural network framework for dimensionality reduction. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 490–497.

[25] Zhenyu Wu, Haotao Wang, Zhaowen Wang, Hailin Jin, and Zhangyang Wang. 2020. Privacy-preserving deep action recognition: An adversarial learning framework and a new dataset. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[26] Zhenyu Wu, Zhangyang Wang, Zhaowen Wang, and Hailin Jin. 2018. Towards privacy-preserving visual recognition via adversarial training: A pilot study. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 606–624.

[27] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

[28] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. 2019. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13, 3 (2019), 1–207.

[29] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*. IEEE, 268–282.

[30] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. 2021. A survey on federated learning. *Knowledge-Based Systems* 216 (2021), 106775. https://doi.org/10.1016/j.knosys.2021.106775

[31] Ligeng Zhu, Zhijian Liu, , and Song Han. 2019. Deep Leakage from Gradients. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*.

A PRIVACY RISK SOURCES

Privacy Risk Sources. We give a formal definition to each of these four attacks as follows:

- *Gradient inversion* is defined as the data reconstruction on the server-end with the support of the model parameters and gradients from the clients [11, 31]. Formally, given a model parameterized by θ, the attacker makes use of the ground-truth gradients g^* of mini-batch from a client to recover the original data in the client by matching the gradient on parameterized data x in terms of the cosine similarity:

$$\max_{x} \frac{\nabla^T \mathcal{L}_\theta(x^*;\theta)\cdot\nabla\mathcal{L}_\theta(x;\theta)}{\|\nabla\mathcal{L}_\theta(x^*;\theta)\|\cdot\|\nabla\mathcal{L}_\theta(x;\theta)\|},$$

  where x^* is the ground-truth data batch and $\mathcal{L}$ is the loss function. The cosine similarity between the ground truth and estimated gradients is the objective of model inversion. Since this objective function is differentiable, the Adam optimizer, which is a strong and commonly-used optimizer in deep learning, is typically utilized to maximize the recovered data until convergence. In federated learning, the gradients are typically replaced by the model update.
- *Membership inference* is first proposed in [21], which uses a binary classifier to predict whether a given data point is in the training set of the target model. Specifically, the attacker has its own dataset, which may not contain labels. The attacker will first collect the predicted labels or logits of part of our own dataset using the target model from the API and then train our own model with the labeled dataset. Note that in FL, model parameters are usually released to the public and are considered to be a white-box setting. Then we have a model to mimic the target model, a labeled dataset to train the model, and an unlabeled dataset not adopted for training. The attacker can then use these two datasets to train a binary classifier that predicts the attendance of the data point in the original training set.
- *Attribute inference* is the attack where the attacker uses a model to infer missing information of a data point from its incomplete information [29]. For instance, with tabular financial data, the adversary may have only part of the elements of each item. The attribute inference is to optimize the missing part towards the minimum loss.
- *Feature leakage* is another potential privacy risk in FL that some unreliable modes may leak the features of data to other third parties. Despite the fact that feature leakage does not lead to direct data leakage, malicious third parties can use such features to adopt and support other kinds of privacy attacks like membership inference and attribute inference.

Privacy Risks in Both Stages. In horizontal federated learning, each client can make predictions locally once trained well in the training stage [7, 22] so the privacy risk decreases during the inference stage. In our setting, however, the account meta-information, such as historical account activities and flags stored in the account clients, needs to be accessed in both training and inference stages, which means the transaction client cannot access such information directly. Moreover, this indicates that even in the inference stage, all the nodes are involved in the prediction, which is different from the horizontal framework. Even worse, we consider that attackers can steal data in the inference stage by utilizing the model inference or membership inference attack on the trained classifier. As such, the challenge arises that we need to protect privacy in both stages. We show that the privacy risks of each communication discussed above are similar for both stages so that we can ensure privacy in both stages by targeting the three privacy risk sources.

	RandomUnder	RandomOver	SMOTE	Reweight
Precision	0.55	0.51	0.51	0.51
Recall	0.80	0.83	0.81	0.88
F1	0.58	0.49	0.51	0.52
AUCPR	0.4712	0.4717	0.4572	0.67

Table 4: Evaluation of imbalance-target methods.

B COMMUNICATION FLOW

Summary of Communication Flows. Considering the aforementioned improvements and privacy protection, we built an implementation of the proposed HyFL based on the Flower [4] project. We summarize the complete communication flows of our implementation in the training and inference stages in Fig. 4 and Fig. 5, respectively. An encryption key exchange is involved during the initialization of the training stage, which will also be used in the testing stage.

C MORE EXPERIMENTS RESULTS

C.1 Ablation Study

C.1.1 Impact of Sampling. In this section, we analyze the impact of different sampling methods used to solve data imbalance. The proportion of the negative and positive data in our training set is about 1 : 783.73. The typical method to solve such data imbalance is to resample the training set so that the number of positive and negative samples is balanced. Another typical method is to reweight the losses of positive and negative samples. We evaluate four methods to evaluate the impact of imbalance-target methods: (1) RandomUnder, which randomly under-sample the negative samples. (2) RandomOver, which randomly over-sample the positive samples (3) SMOTE, which uses KNN to generate synthetic data to augment positive samples. (4) Reweight, which assigns a higher weight to the loss of the positive samples. Surprisingly, we find the model performance is hampered by all the resampling methods. Due to the great data imbalance, such resampling methods will drop a large number of negative samples, which leads to a catastrophe in model performance. Instead, reweighting only suffers from a small loss of model performance since it does not drop any negative samples but forces the model to focus more on the positive samples.

C.1.2 Impact of Communication Interval. In this section, we analyze the impact of the number of communication rounds between the bank clients and the server, as well as that of the communication interval. We set the total number of epochs as 50, where is the multiplication of the communication interval (I) and the communication round number (R). Specifically, we consider four settings with different combinations of I and R. We find if the number of the account clients is small, I and R only have little influence on the performance of the final model. Hence, in this section, we use 100 account clients to evaluate the impact of I and R, presented in Table 5. It is shown that more frequent communication leads to better

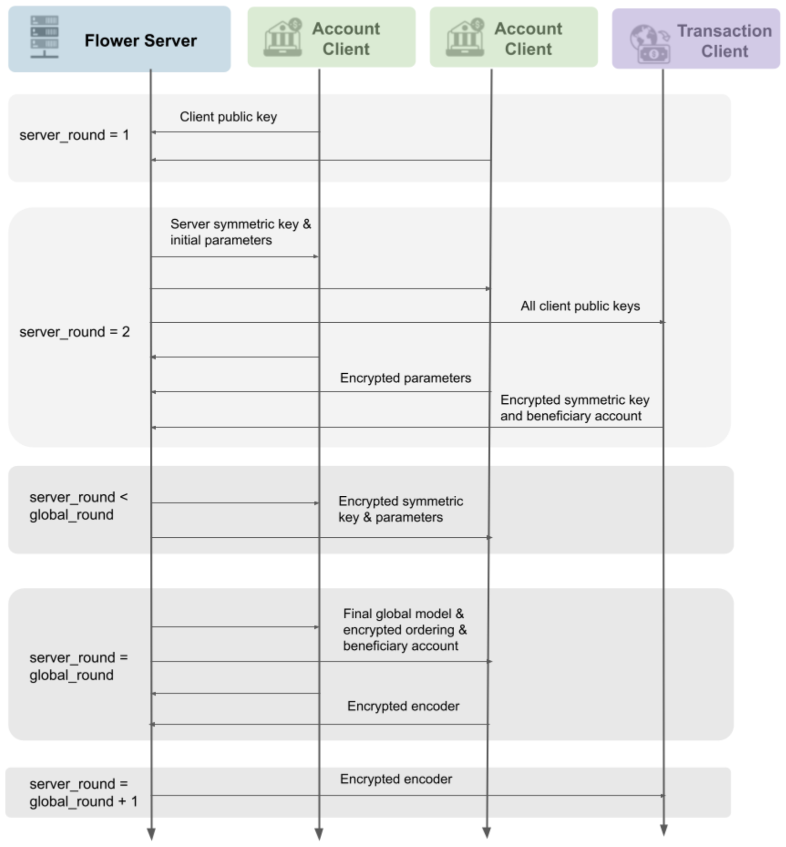

Flower Server
Account Client
Account Client
Transaction Client
server_round = 1
Client public key
server_round = 2
Server symmetric key & initial parameters
All client public keys
Encrypted parameters
Encrypted symmetric key and beneficiary account
server_round < global_round
Encrypted symmetric key & parameters
server_round = global_round
Final global model & encrypted ordering & beneficiary account
Encrypted encoder
server_round = global_round + 1
Encrypted encoder

Figure 4: Overview of the communication flow of the proposed HyFL framework in the training stage.

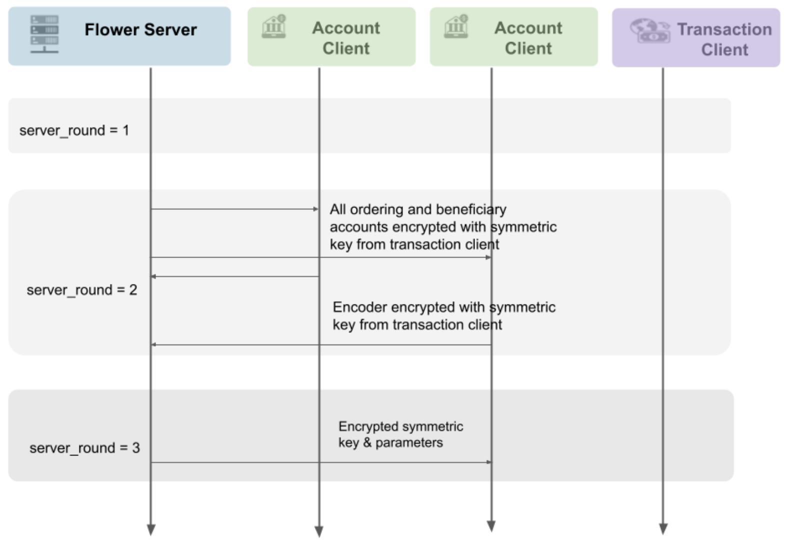

Flower Server
Account Client
Account Client
Transaction Client
server_round = 1
server_round = 2
All ordering and beneficiary accounts encrypted with symmetric key from transaction client
Encoder encrypted with symmetric key from transaction client
server_round = 3
Encrypted symmetric key & parameters

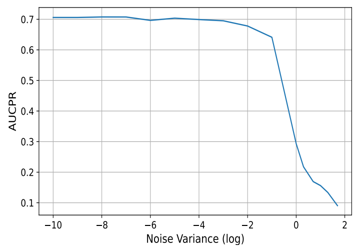
Figure 5: Overview of the communication flow of the proposed HyFL framework in the inference stage.

	I1-R50	I5-R10	I10-R5	I50-R1
Precision	0.97	0.97	0.97	0.97
Recall	0.79	0.79	0.71	0.71
F1	0.86	0.86	0.79	0.79
AUCPR	0.7037	0.7037	0.5344	0.5344

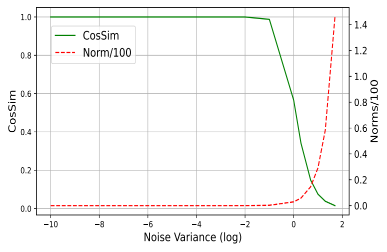
Table 5: Performance evaluation of the number of communication rounds.

model performance. However, due to the overhead of frequent communication, a trade-off is necessary between the communication cost and the model performance.

C.1.3 Impact of Noise. We give an analysis of the model performance w.r.t. different noise variances, shown in Fig. 6. We utilize Gaussian noise with the mean as zero and analyze the model performance with different variances. To study the impact of noise variance, we consider three factors: AUCPR, average norm, and average cosine similarity between original data and noise-injected data. From Fig 6a, we can see that noise with a small variance until 10^{-3} can enhance the model generalization, and hence the AUCPR can be improved. The norm and cosine similarity are close to zero and one, respectively, which indicates that there is little influence on original data with such small noise. However, as the variance increases beyond 10^{-2}, both the cosine similarity and AUCPR decrease significantly, and the norm increases greatly. Thus, the results present a trade-off between the privacy budget and the model utility. It is shown that with the variance as 10^{-3}, the AUCPR on test data peaks, which means a good generalization performance in the inference stage. In this case, we can have a better trade-off between privacy and performance.



(a) AUCPR vs. Variance



(b) Norm / CosSim vs. Variance

Figure 6: The impact of noise variance added into the training data on three factors: AUCPR, Norm, and CosSim. AUCPR is to evaluate model performance on classification. The Norm is calculated as the average norm of noise added to each data point. The CosSim is the average cosine similarity of noisy samples and original samples.