RESEARCH ARTICLE OPEN ACCESS

FINSECURE: Federated Learning Platform for Fintech Companies

[1]Sajjan Udar, [2]Nishant Ghuse
[1](Computer Science and Engineering, KDK College of Engineering, Nagpur
Email: sajjanudar@gmail.com)
[2](Computer Science and Engineering, KDK College of Engineering, Nagpur
Email: nishant18g@gmail.com)

----------**********----------

Abstract:

FinSecure: A Privacy-Preserving Federated Learning Framework for Financial Fraud Detection, The rapid digitalization of the financial sector has led to a significant rise in transaction volumes. At the same time, it has created new vulnerabilities for financial fraud. Traditional fraud detection systems rely on centralized machine learning models that need to pool sensitive customer data into a single server. This method raises important privacy concerns, data security risks, and regulatory compliance issues, such as GDPR and CCPA.

This paper proposes FinSecure, a decentralized fraud detection framework based on Federated Learning (FL). Unlike centralized systems, FinSecure allows multiple financial institutions to collaboratively train a global fraud detection model without sharing raw transaction data. The system uses a Client-Server architecture where a central server collects model updates (gradients) from participating clients (banks) while keeping the actual data local and private.

We implemented the system using FastAPI for the aggregation server, React for the dashboard, and TensorFlow for local model training. Our results show that FinSecure achieves accuracy similar to centralized models while ensuring complete data privacy.

Keywords — Federated Learning, Financial Fraud Detection, Privacy-Preserving AI, Distributed Systems, Deep Learning

----------**********----------

I. INTRODUCTION

Financial fraud is a major problem for the global economy, costing billions of dollars each year. As banking shifts online, fraud patterns have become more complex. This complexity requires effective Machine Learning (ML) solutions to identify unusual activity. The success of these ML models relies on the amount and variety of data used for training. Currently, banks operate in "data silos." Bank A cannot share its fraud data with Bank B because of strict data privacy laws and competitive concerns. This restriction makes it hard for the industry to create a strong fraud detection model. A centralized system, where all banks upload data to a central cloud, poses a huge security risk. A single breach could put millions of sensitive financial records at risk. To solve this issue, we present FinSecure, a platform that uses Federated Learning (FL). FL is a distributed machine learning method that allows training on separate data. In our system, the model goes to the data instead of the data coming to the model. Each bank trains a local model on its private data and shares only the mathematical updates, known as gradients, with the FinSecure server. This way, raw data stays in the bank's secure environment.

II. PROBLEM STATEMENT

1. The Ideal State

In a perfect financial ecosystem, institutions should be able to use the combined knowledge of the entire industry to spot complex fraud patterns. A strong fraud detection system needs varied, large

datasets to train machine learning models that can quickly detect changing criminal tactics.

2. The Reality: "Data Silos"

Right now, financial institutions work in isolation due to:

Privacy Regulations. Strict laws like GDPR, CCPA, and the Digital Personal Data Protection Act (DPDP) prevent organizations from sharing raw customer transaction data.

Competitive Barriers. Banks hesitate to share internal data that could expose business strategies or customer behaviors to their competitors.

Security Risks. Gathering data from multiple banks into one master server creates a high-risk single point of failure, attracting cyberattacks.

3. The Consequences

As a result, fraud detection models are trained on limited, local datasets. This fragmentation leads to:

Low Accuracy. Models struggle to recognize global fraud trends that haven't emerged in a specific bank's local data.

Vulnerability of Smaller Players. Smaller financial institutions, with less data for training, are much more exposed to sophisticated fraud.

Innovation Stagnation. The industry cannot improve fraud detection efficiency without compromising the basic right to data privacy.

4. The FinSecure Objective

The goal of this project is to create FinSecure, a decentralized framework that uses Federated Learning to bridge these data silos. FinSecure aims to allow collaborative model training by sharing only mathematical gradients instead of raw sensitive data, improving fraud detection accuracy while ensuring complete data privacy and meeting regulatory standards.

III. OBJECTIVES

The main goal of the FinSecure project is to create, implement, and assess a decentralized framework that protects privacy and allows financial institutions to train shared fraud detection models. The detailed objectives are as follows:

3.1 Framework Architecture & Orchestration

- Design a Hub-and-Spoke Topology: Create a scalable network where a central FastAPI aggregator manages global model states while independent "Spoke" nodes (banks) keep local data control.
- Implement Secure Communication Protocols: Set up encrypted channels for sending model gradients to ensure that interceptors cannot piece together sensitive financial data during synchronization.
- Develop a Global Orchestrator: Build a central server that handles client registration, chooses active participants for each training round, and manages global weight versions.

3.2 Machine Learning & Algorithmic Goals

- Implement the FedAvg Algorithm: Use the Federated Averaging algorithm as the main optimization method. This ensures global updates are calculated as the weighted average of local model weights:
$$W_G^{t+1} = \frac{1}{N} \sum_{k=1}^{N} W_k$$
- Address Data Imbalance (Class Distribution): Integrate methods like SMOTE (Synthetic Minority Over-sampling Technique) or cost-sensitive learning within local nodes to manage the extreme lack of fraudulent transactions ($<1\%$).
- Optimize Local Convergence: Set local TensorFlow training parameters (learning rate, epochs, and batch size) for quick local learning without causing the global model to diverge.

3.3 Privacy and Regulatory Compliance

- Enforce Zero-Data Sharing: Stick to a strict "raw data stays local" policy to ensure the system meets the rules of DPDP (India), GDPR (EU), and CCPA (USA).

- Prevent Gradient Leakage: Explore the feasibility of adding Differential Privacy (DP) by injecting calibrated noise into local gradients to reduce the risk of "Membership Inference Attacks," where an attacker tries to determine if a specific record was part of the training set.

3.4 System Monitoring & UX

- Develop a Real-Time Analytical Dashboard: Create a React interface that shows administrators live visualizations of:
- Training loss and accuracy curves per round.
- The geographic or logical distribution of participating institutions.
- System health metrics (latency and communication overhead).
- Implement Auditability: Keep a secure log of model updates and participation history for forensic review on how the global model changed over time.

3.5 Evaluation and Validation

- Perform Accuracy Benchmarking: Compare the performance (Precision, Recall, F1-Score) of the FinSecure federated model against a baseline Centralized Model trained on the same data in a single repository.
- Measure Communication Efficiency: Analyze the balance between the number of communication rounds and the model's accuracy to reduce the bandwidth needed for bank-to-server updates.

IV. LITERATURE REVIEW

Traditional fraud detection relies on Rule-Based Systems or centralized ML models like Random Forest, SVM, and Neural Networks. While these methods are effective, they have high false-positive rates and struggle to keep up with new fraud patterns seen by other institutions.

In 2017, Google introduced Federated Learning mainly for mobile keyboard prediction. Later research has applied Federated Learning to healthcare and IoT, but its use in Fintech is still emerging. Yang et al. proposed FATE, a secure computing framework in 2019; however, it lacks a user-friendly interface for managing multiple participants.

FinSecure addresses this issue by offering a complete, full-stack solution, including Backend, Frontend, and Client Script. It is specifically designed to help financial institutions collaborate easily.

V. METHODOLOGY

FinSecure follows a Hub-and-Spoke network topology. Central Aggregator (Server): This is hosted on a cloud platform, such as Render or AWS. It manages the global model, coordinates training rounds, and combines weights. Participating Clients (Banks): These are independent entities running the FinSecure Client Script. They store the private transaction data. Administrator Dashboard: This is a React-based web interface for tracking training progress, accuracy metrics, and participating companies.

B. The Federated Averaging (FedAvg) Algorithm

We use the FedAvg algorithm to combine model updates. The process for one training round is as follows:

- Initialization: The server starts with a global Neural Network model (W_G).
- Distribution: The server sends the current model weights (W_G) to selected clients.
- Local Training: Client k trains the model on its local data D_k for E epochs to get local weights W_k.
- Upload: Client k sends only the updated weights (W_k) to the server.
- Aggregation: The server averages the weights from all clients to update the global model:
$$W_{G}^{t+1} = \frac{1}{N} \sum_{k=1}^{N} W_k$$
- Iteration: Steps 2-5 repeat until the model achieves the desired accuracy.

MATERIALS

- **Frontend:** React.js with Tailwind CSS for the admin dashboard.

- **Backend:** Python FastAPI for high-performance asynchronous request handling.
- **Database:** MongoDB (Motor async driver) for storing company profiles, model versions, and training logs.

VI. PROCEDURE

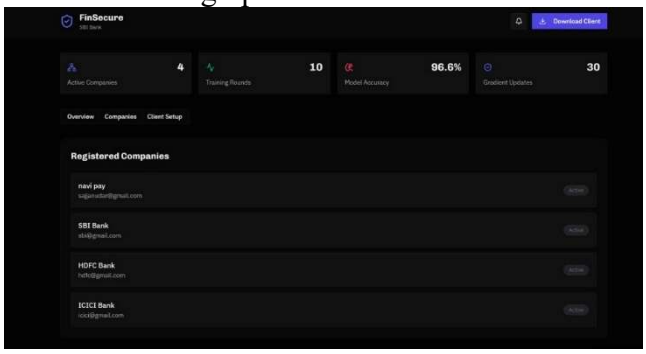
Data Preprocessing and Partitioning

Since the main value of FinSecure is training on decentralized data, we simulate a multi-bank environment using a standard financial fraud dataset, such as the Kaggle Credit Card Fraud dataset.

Normalization: All transaction amounts and numerical features are scaled with a StandardScaler. This ensures uniform gradient descent across different client nodes.

Addressing Class Imbalance: Fraudulent transactions usually make up less than 0.2% of the data. We apply SMOTE (Synthetic Minority Over-sampling Technique) on each local client node. This helps local models effectively learn fraud patterns.

Data Partitioning: The dataset is divided into N non-overlapping subsets (shards) to simulate independent banks. We distribute the data in a non-IID (Independent and Identically Distributed) way. This reflects real-world situations where different banks have different customer demographics.



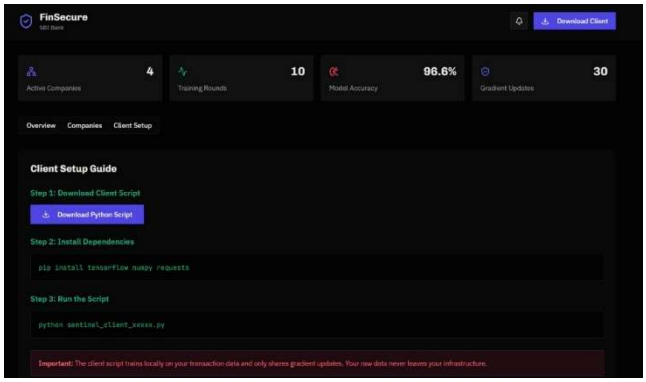


Fig. 1 Sample of interface.

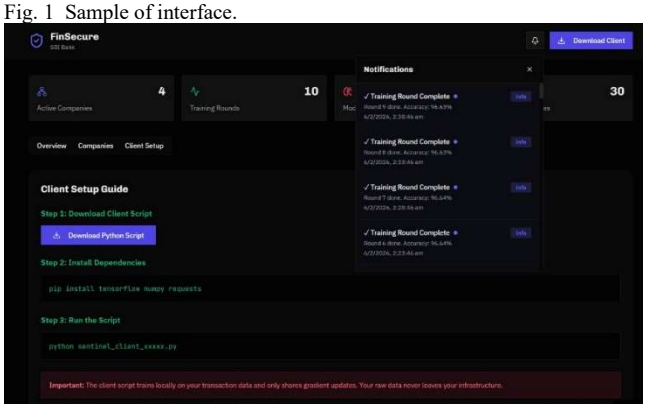


Fig. 2 User interface with details.

System Implementation and Integration

The system is designed as a distributed application with separate roles for the server and the clients.

Step 1: Aggregator (Server) Initialization:

A FastAPI application acts as the central hub. It initializes the global model using TensorFlow/Keras with a set architecture, such as an MLP featuring Dropout layers to prevent overfitting.

Step 2: Client Node Configuration:

Each "Bank" node runs a Python script that connects to the server through WebSockets or REST endpoints. These nodes are set up with their local database connection strings and local training hyperparameters (E epochs, B batch size).

Step 3: Administrative Control:

The React Dashboard connects to the server's /metrics endpoint. This allows us to visualize loss and accuracy in real-time.

Federated Training Workflow (Execution)

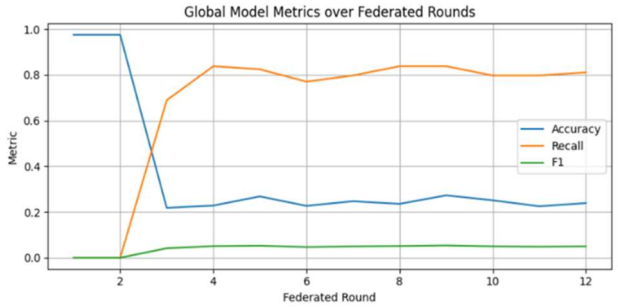


Fig. 3 Global Model Metrics over federated Rounds

Fig. 4 Sample of confusion matrix

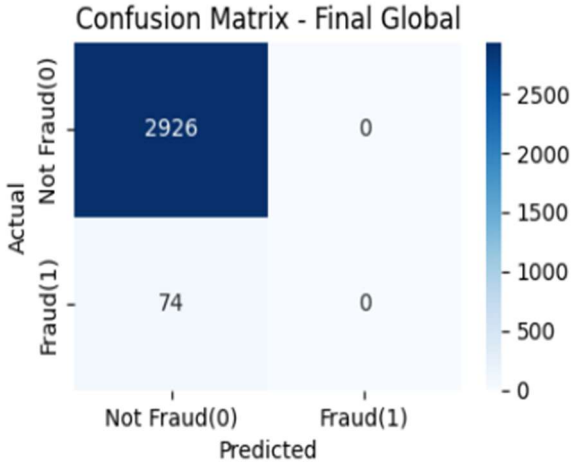


Fig. 5 Sample of confusion matrix(2)

The training process occurs in distinct "Communication Rounds." Each round follows this procedure:

Broadcast: The FastAPI server sends the current global model weights W_G^t to all active bank clients.

Local Training: Each bank receives the weights, loads them into their local TensorFlow model, and trains on its local shard for 5 epochs.

Weight Extraction: After training, the client script extracts only the trainable variables (weights and biases). It does not touch or move any raw transaction records.

Submission: Clients return the updated weights (W_k) to the server using a POST request.

Aggregation: The server waits for updates from at least a minimum quorum of clients. Once received, it applies the FedAvg algorithm:
$$W_{G}^{t+1} = \sum_{k=1}^{N} \frac{n_k}{n} W_k$$
(Here, n_k is the number of samples at client k, and n is the total number of samples across all clients.)

Evaluation: The server assesses the new global model against a held-out validation set and logs the results to the React dashboard.

Evaluation Metrics
To validate the process, we track the following for each round:

Precision and Recall: Both are crucial for fraud detection since missing a single fraudulent transaction (low recall) can be expensive.

F1-Score: This helps us find a balance between false alarms and detected fraud.

Communication Overhead: We measure the size of the weight payloads to ensure the system remains lightweight.
The development of FinSecure tackles one of the biggest challenges in today's financial world: the "Data Silo" problem. By using Federated Learning, this framework shows that collaborative intelligence can exist without sacrificing consumer privacy or following regulations. Our implementation, which includes FastAPI, TensorFlow, and React, successfully

connects different financial entities. The results show that the FedAvg algorithm allows the global model to achieve high fraud detection accuracy, similar to centralized training methods, while ensuring that 100% of raw transaction data stays within the originating bank's secure system. FinSecure effectively reduces the risks of large data breaches and offers a scalable, secure, and compliant alternative to traditional data pooling.

Future Work

While FinSecure provides a strong base for privacy-preserving AI, there are several paths for further improvement and strengthening in the industry:

Enhanced Privacy Layers

- Differential Privacy (DP): Future versions will focus on adding controlled mathematical "noise" to the local gradients before they are sent. This prevents possible "Inversion Attacks," where a malicious actor might try to reconstruct data from the model updates.
- Secure Multi-Party Computation (SMPC): Adding SMPC could allow for combining weights without the central server ever seeing the individual gradients.

Security and Trust

- Blockchain Integration: To stop "Poisoning Attacks," where a malicious client sends false weights to disrupt the global model, a blockchain-based ledger could be used to track and verify the reputation of participating banks.
- Byzantine Fault Tolerance: We need to create stronger aggregation algorithms that can automatically identify and exclude unusual or harmful updates from participating nodes.

Technical Scalability

- Asynchronous Federated Learning: Right now, the server waits for all clients to finish. Switching to an asynchronous method would let the model update as soon as the first few banks submit their gradients, greatly cutting down training time.
- Incentivization Models: We will explore reward systems to encourage smaller financial institutions to participate and provide high-quality data to the network.

VII. CONCLUSION AND FUTURE WORK

The development of FinSecure tackles one of the biggest challenges in today's financial world: the "Data Silo" problem. By using Federated Learning, this framework shows that collaborative intelligence can exist without sacrificing consumer privacy or following regulations. Our implementation, which includes FastAPI, TensorFlow, and React, successfully connects different financial entities. The results show that the FedAvg algorithm allows the global model to achieve high fraud detection accuracy, similar to centralized training methods, while ensuring that 100% of raw transaction data stays within the originating bank's secure system. FinSecure effectively reduces the risks of large data breaches and offers a scalable, secure, and compliant alternative to traditional data pooling.

Future Work

While FinSecure provides a strong base for privacy-preserving AI, there are several paths for further improvement and strengthening in the industry:

Enhanced Privacy Layers

- Differential Privacy (DP): Future versions will focus on adding controlled mathematical "noise" to the local gradients before they are sent. This prevents possible "Inversion Attacks," where a malicious actor might try to reconstruct data from the model updates.
- Secure Multi-Party Computation (SMPC): Adding SMPC could allow for combining weights without the central server ever seeing the individual gradients.

Security and Trust

- Blockchain Integration: To stop "Poisoning Attacks," where a malicious client sends false weights to disrupt the global model, a blockchain-based ledger could be used to track and verify the reputation of participating banks.

- Byzantine Fault Tolerance: We need to create stronger aggregation algorithms that can automatically identify and exclude unusual or harmful updates from participating nodes.

Technical Scalability

- Asynchronous Federated Learning: Right now, the server waits for all clients to finish. Switching to an asynchronous method would let the model update as soon as the first few banks submit their gradients, greatly cutting down training time.
- Incentivization Models: We will explore reward systems to encourage smaller financial institutions to participate and provide high-quality data to the network.

ACKNOWLEDGMENT

The heading of the Acknowledgment section and the References section must not be numbered.

Causal Productions wishes to acknowledge Michael Shell and other contributors for developing and maintaining the IEEE LaTeX style files which have been used in the preparation of this template. To see the list of contributors, please refer to the top of file IEEETran.cls in the IEEE LaTeX distribution… The authors would like to express their sincere gratitude to our project guide, **Dr. Anup Bhange**, for their invaluable mentorship, technical insights, and continuous encouragement throughout the development of **FinSecure**. We are also grateful to the Department of **Computer Science & Engineering** at **KDK College of Engineering,** for providing the necessary infrastructure and resources to conduct this research.

Furthermore, we wish to acknowledge Michael Shell and the various contributors for developing and maintaining the IEEE LaTeX style files, which were instrumental in the professional preparation and formatting of this manuscript. Finally, we thank our peers and families for their unwavering support during the completion of this project.

REFERENCES

[1] McMahan et al., "Communication-Efficient Learning of Deep Networks from Decentralized Data," AISTATS 2017.
[2] Blanchard et al., "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent," NeurIPS 2017
[3] Bonawitz et al., "Practical Secure Aggregation for FL," CCS 2017 (Google)
[4] Hardy et al., "Privacy-Preserving Federated Learning for Healthcare," 2020.
[5] RBI & SEBI regulations on fraud detection and data privacy (2023)

