End-to-End Secure Federated Learning for Cross-Client Credit Card Fraud Detection

Freddie Faulkner*, Amar Nath Patra†, Raj Mani Shukla*, and Suman Bhunia‡
*Computing and Information Science, Anglia Ruskin University, Cambridge, UK
freddie.faulkner@student.aru.ac.uk, raj.shukla@aru.ac.uk
†School of Computing and Information Sciences, Radford University, Virginia, USA
apatra@radford.edu
‡Department of Computer Science and Software Engineering, Miami University, Ohio, USA
bhunias@miamioh.edu

***Abstract*—Credit card fraud continues to impose substantial financial losses on consumers and institutions, motivating the development of intelligent detection systems that are both accurate and privacy preserving. While machine learning–based fraud detection has shown strong performance in centralized settings, regulatory constraints, data sensitivity, and institutional data silos limit the feasibility of centralized model training in real-world financial environments. Federated learning (FL) addresses these challenges by enabling collaborative model training across multiple clients without exchanging raw transaction data. However, the practical deployment of FL for financial fraud detection remains hindered by security vulnerabilities, susceptibility to poisoning attacks, and a lack of guidance on end-to-end, cloud-deployable system design. This paper presents an end-to-end secure federated learning framework for cross-client credit card fraud detection that integrates multiple complementary security mechanisms within a real time, cloud-based pipeline. The proposed approach combines homomorphic encryption to protect client model updates, hash-based integrity verification to ensure update authenticity, and Fool's Gold based trust scoring to detect and suppress anomalous or adversarial client contributions during aggregation. The framework is implemented and evaluated as a proof of concept within an Amazon Web Services (AWS) sandbox environment, demonstrating how secure federated training can be operationalized using widely available cloud infrastructure. Experimental results using open-source credit card transaction datasets show that the proposed system achieves competitive fraud detection performance while significantly enhancing resilience against data poisoning and inference threats. By unifying privacy preservation, robustness, and deployability in a single architecture, this work provides practical design guidance for secure federated learning in financial applications. It highlights a viable path toward real world adoption of collaborative, privacy aware fraud detection systems.**

***Index Terms*—Federated learning, privacy-preserving, financial security, differential privacy, cloud computing.**

I. Introduction

Credit card fraud remains a significant threat to the financial sector, contributing to substantial economic losses each year. Recent data from the Federal Trade Commission indicate that consumers lost more than $12.5 billion to fraud in 2024, reflecting a 25% increase over the previous year [1]. As fraud tactics continue to evolve, financial institutions require advanced Artificial Intelligence (AI) and machine learning (ML) systems capable of adapting to emerging malicious behaviors in real time [2].

Traditional centralized ML approaches for fraud detection rely on aggregating sensitive data into a single repository, raising concerns related to privacy, regulatory compliance, and security. Federated learning (FL) offers a promising alternative by enabling institutions to collaboratively train a shared model without exchanging raw data. This distributed paradigm enhances privacy preservation while leveraging diverse datasets that would otherwise remain siloed. However, deploying FL in financial environments introduces challenges such as infrastructure limitations, exposure to model poisoning attacks, and the need for secure global model hosting.

This study investigates how federated learning (FL) can be applied to improve credit card fraud detection while addressing the security and operational challenges of real-world deployment. Across the literature, three recurring needs emerge, including scalable and accurate fraud detection, strong privacy and security guarantees, and real-time deployability in cloud environments [3]. While FL has demonstrated promise in improving accuracy and preserving privacy, existing work provides limited guidance on end-to-end secure FL pipelines, ensuring model transparency, or deploying resource-efficient, real-time fraud detection systems. To address these gaps, this study uniquely integrates homomorphic encryption [4], hash-based integrity verification [5], and Fool's Gold–based defenses [6] into a real-time, cross-client federated learning framework for credit card fraud detection. By combining multiple-layered security measures in a cloud-deployable pipeline within an Amazon Web Services (AWS) sandbox environment [7], the proposed approach mitigates data poisoning and inference attacks while enabling practical operational deployment. Drawing on existing literature, this work presents a proof-of-concept framework demonstrating how secure federated models can be effectively implemented using cloud technologies. This paper makes the following key contributions

- Design and implementation of a secure federated learning framework for real-time credit card fraud detection that enables cross-client collaboration without sharing raw transactional data.
- Integration of layered security mechanisms, including ho-

momorphic encryption, data integrity hashing, and Fool's Gold–based anomaly detection, to mitigate poisoning and inference attacks in federated training.
- Experimental evaluation using open-source public datasets in a cloud-based sandbox environment to demonstrate the operational feasibility and robustness of the proposed framework.

The remainder of this paper is organized as follows. Section II reviews related work, including existing applications of federated learning, identified limitations, and security mechanisms for enhancing model robustness. Section III details the system architecture and implementation. Section IV presents and analyzes the experimental results. Finally, Section V concludes the paper and outlines for future research directions.

II. Literature Review

Existing research on credit card fraud detection demonstrates growing interest in advanced ML and privacy-preserving methods as fraud becomes more prevalent in increasingly cashless economies. Traditional approaches often apply data mining, fuzzy logic, and rule-based systems. For example, Chaudhary et al. [8] used genetic algorithms to refine rule-based detection, achieving high accuracy but offering a limited discussion of security vulnerabilities, highlighting a critical gap in the robustness of conventional systems.

Several studies identify persistent challenges in fraud detection. Kulatilleke [9] emphasizes issues such as class imbalance, limited availability of high-quality financial datasets, and the lack of cross-organizational data sharing due to privacy constraints. FL is frequently proposed as a solution because it enables collaborative training without sharing raw data, addressing the data-silo and privacy issues. However, additional work is needed to support secure real-world deployment.

Deep learning techniques have also been explored. Alarfaj et al. [10] evaluated convolutional neural networks (CNNs) and long short-term memory (LSTM) networks, demonstrating improved performance over classical ML models but not addressing how such models can be securely scaled. Mariam [11] further highlights the importance of high-quality data and feature engineering, showing significant impacts on detection accuracy and false-positive rates.

Several surveys compare centralized and federated approaches. Abdulrahman et al. [12] show that centralized learning poses privacy risks and discourages collaboration, whereas FL improves privacy but may be vulnerable to anomalous client updates. Riviera et al. [13] analyze horizontal and vertical FL, noting that horizontal FL often trains faster, while vertical FL may be more suitable for financial data distributions.

Security remains a major concern. Sun et al. [14] outline poisoning attacks targeting both local datasets and global aggregation processes, underscoring the need for robust defenses. Nandakumar et al. [15] propose fully homomorphic encryption for secure computation, though it introduces significant computational overhead. Fereidooni et al. [16] discuss vulnerability to inference attacks and propose SafeLearn, a more communication-efficient secure aggregation protocol.

Further research validates the benefits of FL for fraud detection. Yang et al. [17] show that FL improves performance by roughly 10% over centralized models, and Bin Sulaiman et al. [18] demonstrate that hybrid neural network–FL architectures can improve accuracy while preserving privacy. Awosika et al. [19] integrate FL with explainable AI (XAI), improving transparency—an important requirement in regulated financial environments.

Scalability is another challenge. Huba et al. [20] find that FL models scale effectively, with asynchronous FL converging faster and using fewer resources than synchronous methods. For deployment, Loconte et al. [21] propose serverless cloud platforms to host FL models efficiently, while McDonnell et al. [22] highlight the importance of reproducible cloud infrastructure using Infrastructure as Code (IaC).

Across the literature, three recurring needs emerge: (1) scalable and accurate fraud detection, (2) strong privacy and security guarantees, and (3) real-time deployability in cloud environments. Although FL has shown promise in improving accuracy and preserving privacy, existing work provides limited guidance on securing FL pipelines end-to-end, ensuring model transparency, or deploying resource-efficient real-time fraud detection systems. These gaps motivate the present study, which examines secure, cloud-based FL architectures for real-time credit card fraud detection.

III. Methodology

The design of the proposed fraud detection system is driven by deployment, scalability, and security requirements. Cloud platforms offer elastic compute, operational resilience, and reduced capital costs [23]; serverless architectures, in particular, provide event-driven scalability and pay-per-use efficiency well suited for real-time transaction scoring [24], while Terraform enables reproducible infrastructure deployment through infrastructure as code (IaC) [25]. FL enables privacy-preserving collaboration across distributed datasets, but its exposure to inference and poisoning attacks necessitates strong protections.

A. Design and Development

The proposed cloud-based system ingests anonymized credit card transactions, which include features such as transaction type, merchant, timestamp, and location. Exploratory data analysis (EDA) is first conducted to identify patterns and anomalies through graphical visualization. The data then moves through a preprocessing pipeline: categorical features are label-encoded, values are normalized using a standard scaler to improve convergence and limit variance effects, and the dataset is split into feature (X) and label (y) sets before being divided into training and testing subsets. To simulate FL, the training data is further partitioned across five clients and converted into tensors for efficient processing.

In the federated learning (FL) setup, each client trains a local model on its own data and transmits only encrypted

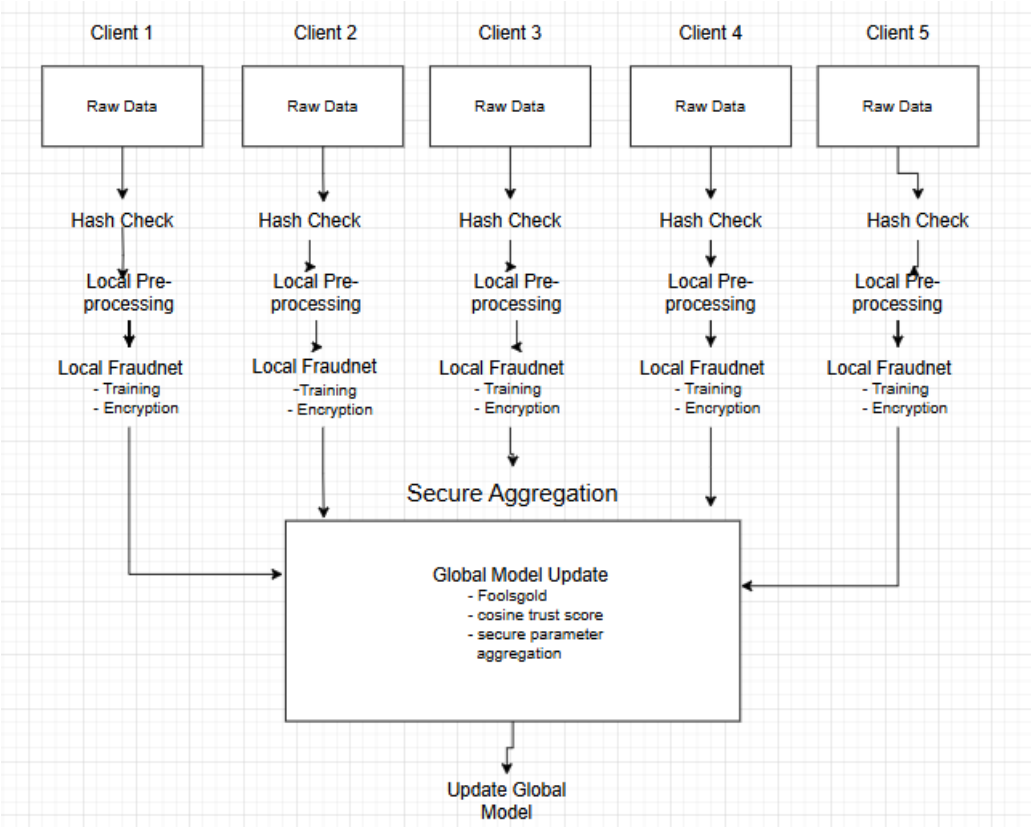

Fig. 1. Secure Federated Learning architecture

model parameters to a central server for aggregation [26]. To meet financial-sector privacy and security requirements, homomorphic encryption protects model parameters in transit and enables computation over encrypted values [4], while cryptographic hashing is used to verify dataset integrity prior to training [5]; any hash mismatch halts the workflow to prevent poisoned data from entering the training pipeline.

To further protect the global model from corrupted or adversarial clients, the system employs Fool's Gold, which applies cosine-similarity-based trust scoring to identify anomalous client updates and exclude them from aggregation [6]. This layered defense strategy enhances the robustness of the federated system against data poisoning and other model-targeted attacks.

B. Initial Modeling and Implementation

Five clients are configured with distinct data partitions and begin by verifying data integrity using a hash-based check (Fig. 1, Fig. 2). Each dataset is initially hashed using the SHA-256 algorithm, and the resulting value is stored; before training, the data are rehashed and compared against the stored reference, following standard cryptographic integrity verification practices [5]. A match allows training to proceed, while a mismatch indicates potential data compromise, triggers a warning, and halts training to prevent poisoning attacks.

Following successful verification and local preprocessing, each client initializes and trains a local model, encrypts its parameters using homomorphic encryption, and securely transmits the encrypted updates to the aggregation function, consistent with the federated learning paradigm [26] and privacy-preserving computation techniques [4]. To further protect against data-poisoning attacks during federated learning, the aggregator applies Fool's Gold anomaly detection with cosine-based trust scoring to evaluate client gradient updates (Fig. 3) [6]. Fool's Gold measures the similarity of client updates relative to prior global model changes and assigns deviation-based trust scores, which are further normalized. Updates that fall outside the acceptable trust range are penalized and excluded from aggregation, while the remaining updates are incorporated into a global model that is subsequently initialized and securely distributed to all clients.

A thread pool executor is used to simulate the five clients in parallel [27]. During each round, clients verify data integrity via hashing, train locally, and return updated weights. The implementation also supports injecting noise into a randomly chosen client to simulate an attack. The system computes client gradients, applies Fool's Gold and cosine trust scoring to detect anomalies, excludes compromised clients, then encrypts and passes valid parameters to the secure aggregation function. After aggregation, the global model is updated using the decrypted weights.

The live model is stored in an Amazon S3 bucket and loaded by an AWS Lambda function, which receives transaction data

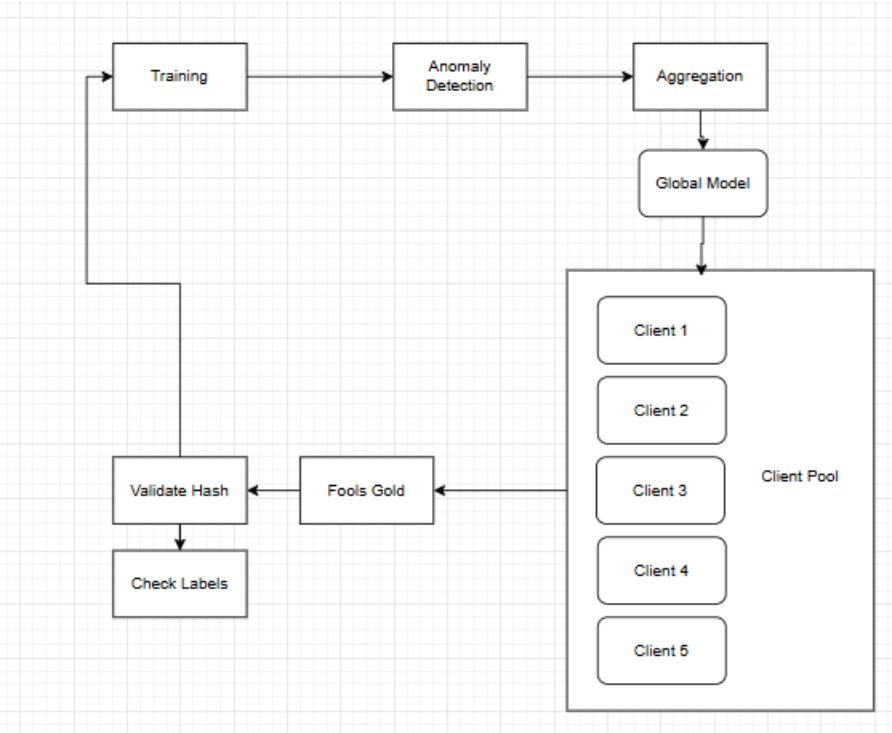

Fig. 2. End-to-end secure federated learning architecture

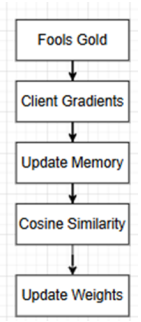

Fig. 3. Anomaly detection process

from S3 and returns a fraud or non-fraud prediction. An API Gateway HTTP trigger invokes the Lambda function, enabling real-time inference through a REST endpoint. All cloud resources, including the S3 bucket, API Gateway, HTTP trigger, and Lambda configuration, are provisioned using Terraform to ensure consistent and repeatable deployment across sandbox environments through infrastructure as code (IaC) [25], [28].

When invoked, the Lambda function retrieves the model from S3, loads it in evaluation mode, parses the incoming transaction, extracts the features, and converts them into a tensor. The model then generates an output, applies SoftMax to compute class probabilities, and returns the predicted label in an HTTP response. If the Lambda function fails, for example, due to insufficient permissions, API Gateway returns an HTTP 500 error along with the exception message for debugging [28].

Finally, testing the model in the cloud involves automating the submission of selected transactions and capturing the responses. It begins by loading previously unseen transactions and converting them into a format that the Lambda function can process. Then it invokes the function for each transaction and records the predictions returned by the cloud-hosted model.

C. Data Gathering

An open-source credit card transaction dataset [29] from Kaggle is used to train and test the model. It contains more than 1.85 million rows and 24 feature columns, providing sufficient data for this work. The dataset includes features such as transaction type, amount, location, and recipient information, allowing it to approximate the kind of inputs a production model would receive. However, it exhibits a significant class imbalance, with most transactions labeled as non-fraudulent (Fig. 4). Such an imbalance can bias a model toward predicting the majority class [30]. To address this, the Synthetic Minority Oversampling Technique (SMOTE) is applied to generate synthetic samples of the minority (fraud) class, improving class balance in the training set [31]. Fig. 5 shows the class distribution after applying SMOTE.

D. Local Training Step

For FL to work, each client must first train the model locally. The client trains its local model on its own data, using either GPU or CPU resources. During training, the data are divided into mini-batches, passed through the model to generate predictions, and used to update the model's parameters.

Fig. 4. Class imbalance before SMOTE

Fig. 5. Class imbalance After SMOTE

Gradients are cleared at the start of each epoch to ensure proper optimization.

E. Train Client Function

The client training process defines the steps required to train each local model and produce weights and metrics for aggregation into the global model. Training begins by instantiating the client model and tracking the training duration.The Adam optimizer is used for its fast convergence [32]. The model then trains locally for several epochs, iterating through batches of client data, updating parameters, and computing the resulting weights. Once training is complete, the client returns its updated weights for aggregation into the global model.

F. Secure Aggregation Function

A secure aggregation function is used after clients submit their locally trained model parameters to the global server. It decrypts the client weights, computes the aggregated sum of their trust scores and parameters, and prepares an updated global model. A model template is then returned for distribution back to the clients.

G. Federated Loop

The FL loop orchestrates the entire process by coordinating client training, secure weight exchange, and anomaly detection. It evaluates performance across different training rounds, recording metrics for comparison. As discussed before, a thread pool is used to simulate parallel client training. During each round, one client is randomly selected to receive noise in its updates to mimic poisoned data. Client gradients are computed and evaluated using Fool's Gold anomaly detection,

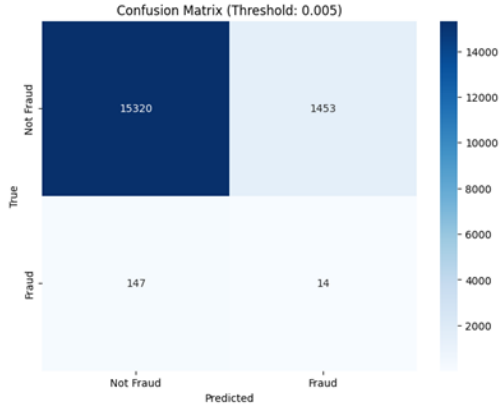

Fig. 6. Confusion matrix for global model

which applies cosine-based trust scores to identify unreliable clients [6]. The least trustworthy client is removed from the aggregation pool. Secure aggregation is then performed, the global model is updated, and the updated model template is sent back to all clients.

IV. Results and Discussion

The system is deployed using a serverless, cloud-based architecture to ensure real-time inference, efficient resource use, and consistent redeployment across environments. The following are the tools and technologies used.

- Programming Language: Python
- Libraries: NumPy, pandas, random, Matplotlib, seaborn, scikit-learn, TensorFlow, PyTorch, Crypto, pickle, ThreadPoolExecutor
- Cloud Provider: AWS
- Infrastructure Tools: Terraform, AWS CLI
- Software: Visual Studio Code, Anaconda Navigator

The global federated model achieved 90.55% accuracy, 98.12% precision, 90.55% recall, and an F1-score of 94.15%. These results indicate strong performance in distinguishing fraudulent from non-fraudulent transactions. The high precision suggests that flagged transactions are very likely to be truly fraudulent, while the high recall indicates the model can correctly identify most fraudulent cases. Although these metrics appear strong, it remains important to verify that they are not inflated by issues such as overfitting. The confusion matrix below summarizes the model's prediction outcomes.

To evaluate the security features of the model, simulated attacks were introduced, and the performance of the model with security enabled was compared to a baseline without security. The Fool's Gold anomaly-detection mechanism improved overall accuracy by correctly identifying and excluding malicious or poisoned client updates (Fig. 7). This shows that incorporating Fool's Gold increases the robustness of the global model and reduces the impact of adversarial data on its performance.

A final experiment examined how increasing the number of federated training rounds affected the performance of the

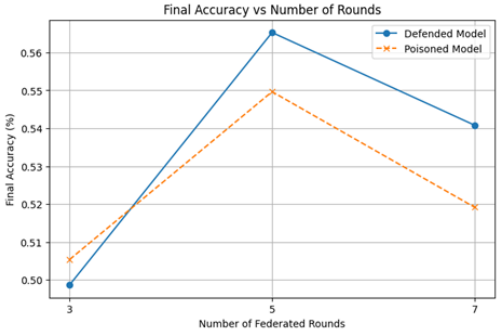

Fig. 7. Comparison of performance between secure vs non-secure model

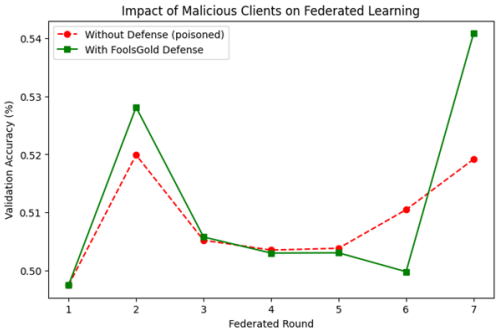

Fig. 8. Impact of malicious client per federated round on secure and non-secure model

TABLE I
COMPARISON OF MODEL METRICS

Metric	FraudNet	MLP	CNN
Accuracy	90.55%	94.58%	90.39%
Precision	98.12%	94.59%	91.01%
Recall	90.55%	94.58%	90.39%
F1-Score	94.15%	94.58%	90.36%

global model. Raising the rounds from five to seven substantially improved accuracy, whereas the improvement from three to five rounds was minimal (Fig. 8). Additional rounds also strengthened the effectiveness of security mechanisms, likely because more rounds provide more information for computing reliable trust scores. After seven rounds, the protected model again outperformed the unprotected model.

The results also show that trust scores among clients are initially widely dispersed but converge as the number of rounds increases, reaching their most stable point around round six before diverging slightly at round seven (Fig. 9), suggesting that six rounds may be optimal for trust-score stability.

Further tests were conducted on three model architectures to compare the impact of different security methods (Table I). The best-performing model was the MLP. The CNN performed the worst, with the lowest overall metrics, and the FraudNet-inspired model performed between the two. In general, all three models were effective in detecting fraudulent transactions.

Testing across different training rounds showed that the CNN was the most sensitive to changes in the number of rounds, while the FraudNet model was the least affected. This suggests that additional training iterations have a minimal impact on FraudNet's performance compared with the MLP and CNN (Figs. 10-12). Overall, the MLP performed the most consistently when using the previously implemented anomaly detection and security mechanisms. The CNN's performance declined toward the end of training, whereas the FraudNet model maintained higher accuracy after the seventh round with security features enabled. These results indicate that the security mechanisms helped preserve model accuracy under attack (Figs. 13-15).

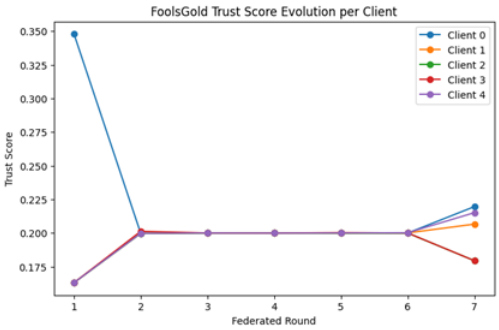

Fig. 9. Trust score evolution per client

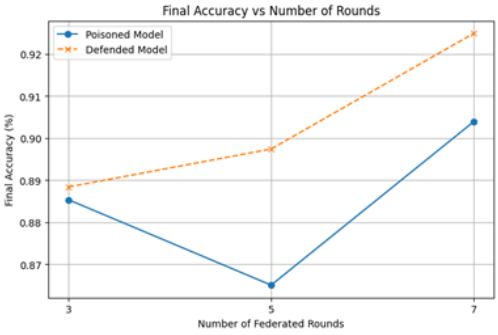

Fig. 10. Model accuracy over number of rounds for CNN

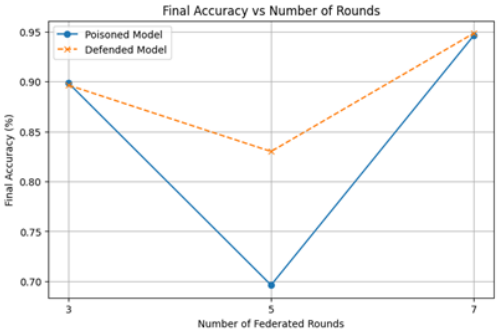

Fig. 11. Model accuracy over number of rounds for MLP

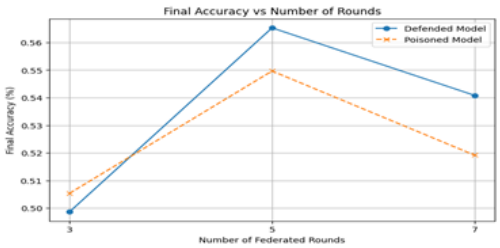

Fig. 12. Model accuracy over number of rounds for FraudNet

Fig. 13. Comparison of MLP with and without security measures

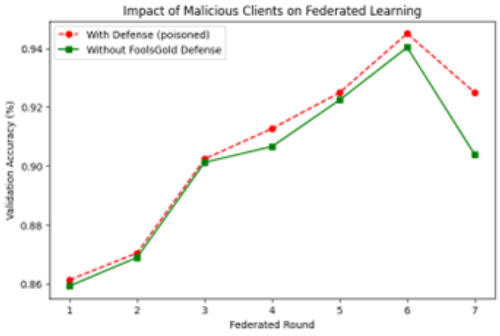

Fig. 14. Comparison of CNN with and without security measure

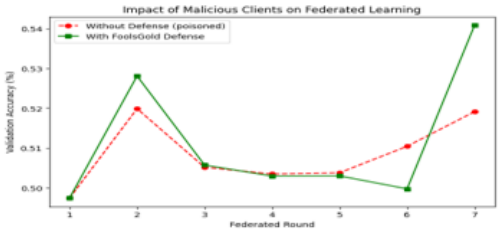

Fig. 15. Comparison of Fraudnet with and without security measures

Fig. 16. Terraform template output

Fig. 17. Output of the Lambda function predicting a passed transaction

A final test simulated a real-world production environment by deploying the model in the cloud and invoking it through an AWS Lambda function [28]. The model received transaction data and returned a prediction indicating whether the transaction was fraudulent or not, demonstrating how it would operate in production. A Terraform template was used to manage and track infrastructure changes (Fig. 16). When triggered, the Lambda function processed each transaction through the model and returned the corresponding prediction (Fig. 17). These results confirm that the public cloud can reliably host the model and support real-time inference workloads.

V. Conclusions and Future Work

This work presents a proof-of-concept federated learning (FL) system for credit card fraud detection that integrates cryptographic and machine-learning–based security mechanisms to enhance robustness against poisoning and inference attacks. Model parameters and training data are protected using homomorphic encryption and cryptographic hashing, while anomaly detection successfully identifies malicious client updates. The results demonstrate that FL enables collaborative model training without centralized data storage while maintaining strong detection performance.

Experimental evaluation shows that the model accurately classifies fraudulent transactions and remains resilient under adversarial conditions. Deployment in a cloud environment further confirms its feasibility for real-time inference in a production-like setting. These findings support the use of FL as a secure and effective approach for fraud detection in privacy-sensitive financial environments.

Limitations include evaluation against only a single malicious client and the use of static security mechanisms, which may not fully reflect distributed or adaptive attacks [3], [6]. Future work will focus on improving model performance, integrating continuous learning and adaptive anomaly detection, and extending security through techniques such as device fingerprinting [3]. Additional enhancements include exploring hybrid or multi-cloud deployments to improve system resilience against provider outages [23].

References

[1] Federal Trade Commission, “New FTC Data Show a Big Jump in Reported Losses to Fraud to $12.5 Billion in 2024.” https://shorturl.at/EwncB, March 2025.

[2] I. Y. Hafez, A. Y. Hafez, A. Saleh, A. A. Abd El-Mageed, and A. A. Abohany, “A systematic review of ai-enhanced techniques in credit card fraud detection,” *Journal of Big Data*, vol. 12, no. 1, p. 6, 2025.

[3] Y. Zhang, D. Zeng, J. Luo, Z. Xu, and I. King, "A survey of trustworthy federated learning with perspectives on security, robustness, and privacy," *arXiv preprint arXiv:2302.10637*, 2023.

[4] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pp. 169–178, 2009.

[5] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology – CRYPTO '87*, pp. 369–378, 1988.

[6] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, 2018.

[7] Amazon Web Services, "Aws sandbox environment." https://aws.amazon.com/, 2024. Accessed: 2025-12-20.

[8] K. Chaudhary, J. Yadav, and B. Mallick, "A review of fraud detection techniques: Credit card," *International Journal of Computer Applications*, vol. 45, no. 1, pp. 39–44, 2012.

[9] G. K. Kulatilleke, "Challenges and complexities in machine learning based credit card fraud detection," *arXiv preprint arXiv:2208.10943*, 2022.

[10] F. K. Alarfaj, I. Malik, H. U. Khan, N. Almusallam, M. Ramzan, and M. Ahmed, "Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms," *IEEE Access*, vol. 10, pp. 39700–39715, 2022.

[11] B. Mariam, A. Okunola, and A. Lanoa, "The impact of data quality and feature engineering on the performance of ai fraud detection system," 2025.

[12] S. AbdulRahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5476–5497, 2020.

[13] W. Riviera, I. B. Galazzo, and G. Menegaz, "Flavourite: Horizontal and vertical federated learning setting comparison," *IEEE Access*, 2025.

[14] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, and J. Liu, "Data poisoning attacks on federated machine learning," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11365–11375, 2021.

[15] K. Nandakumar, N. Ratha, S. Pankanti, and S. Halevi, "Towards deep neural network training on encrypted data," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 0–0, 2019.

[16] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame, *et al.*, "Safelearn: Secure aggregation for private federated learning," in *2021 IEEE security and privacy workshops (SPW)*, pp. 56–62, IEEE, 2021.

[17] W. Yang, Y. Zhang, K. Ye, L. Li, and C.-Z. Xu, "Ffd: A federated learning based method for credit card fraud detection," in *International conference on big data*, pp. 18–32, Springer, 2019.

[18] R. Bin Sulaiman, V. Schetinin, and P. Sant, "Review of machine learning approach on credit card fraud detection," *Human-Centric Intelligent Systems*, vol. 2, no. 1, pp. 55–68, 2022.

[19] T. Awosika, R. M. Shukla, and B. Pranggono, "Transparency and privacy: the role of explainable ai and federated learning in financial fraud detection," *IEEE access*, vol. 12, pp. 64551–64560, 2024.

[20] D. Huba, J. Nguyen, K. Malik, R. Zhu, M. Rabbat, A. Yousefpour, C.-J. Wu, H. Zhan, P. Ustinov, H. Srinivas, *et al.*, "Papaya: Practical, private, and scalable federated learning," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 814–832, 2022.

[21] D. Loconte, S. Ieva, A. Pinto, G. Loseto, F. Scioscia, and M. Ruta, "Expanding the cloud-to-edge continuum to the iot in serverless federated learning," *Future Generation Computer Systems*, vol. 155, pp. 447–462, 2024.

[22] M. McDonnell, E. Cranfill, J. Kincl, and A. M. Thakur, "Infrastructure-as-code for scientific software and its importance towards federated infrastructure," 2021.

[23] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[24] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, "Cloud programming simplified: A berkeley view on serverless computing," *arXiv preprint arXiv:1902.03383*, 2019.

[25] HashiCorp, "Terraform: Infrastructure as code." https://www.terraform.io, 2024. Accessed: 2025-12-20.

[26] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282, 2017.

[27] A. Wahab, "Python ThreadPoolExecutor: Use Cases for Parallel Processing." https://tinyurl.com/3676sean, April 2023.

[28] Amazon Web Services, "Aws documentation." https://docs.aws.amazon.com/, 2024. Accessed: 2025-12-20.

[29] Lee, R, "Credit Card Transactions Dataset." https://www.kaggle.com/datasets/priyamchoksi/credit-card-transactions-dataset/data, December 2024.

[30] Tang, T., "Class Imbalance Strategies — A Visual Guide with Code." https://medium.com/data-science/class-imbalance-strategies-a-visual-guide-with-code-8bc8fae71e1a, April 2023.

[31] S. Matharaarachchi, M. Domaratzki, and S. Muthukumarana, "Machine learning with applications," *Elsevier*, vol. 18, 2024.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, 2015.