Privacy-Preserving Machine Learning Tool for Mitigating Data Leakage in Microservices Architectures

By

Collins Sikolia

171223

Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Computing and Information Systems at Strathmore University

School of Computing & Engineering Sciences

Strathmore University

Nairobi, Kenya

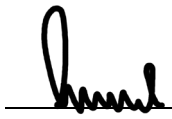
June 2025

Declaration and Approval

Declaration

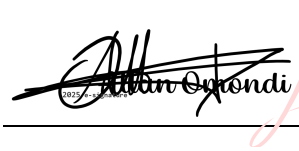
I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

Student's Name: **Collins Sikolia**

Sign:  Date: 28/03/2025

Approval

The dissertation of Collins Sikolia was reviewed and approved for examination by the following:

Sign:  Digitally signed by Allan Omondi Date: 2025.03.28 10:54:57 +03'00' Date: 28-MAR-2025

Dr Allan Omondi.
Lecturer,

School of Computing & Engineering Sciences,
Strathmore University

Abstract

In the era of distributed systems and microservices architectures, the risk of data leakage, particularly of personally identifiable information (PII), has become a critical concern. Federated Learning (FL) emerges as a promising solution by enabling collaborative model training across decentralized data sources without the need to transfer raw data to a central server, thereby preserving user privacy. This study implemented a privacy-preserving federated learning utilizing Flower frammework, an open-source FL platform, in conjunction with TensorFlow for model development. The Federated Averaging (FedAvg) algorithm was employed as the core aggregation strategy to combine model updates from multiple clients. A hybrid deep learning model was designed to optimize learning across the distributed network, ensuring both robustness and scalability. Over the course of five training rounds, two clients participated in each round, contributing locally trained model weights to a central aggregator. The performance of the federated model was rigorously evaluated using standard machine learning metrics: accuracy, loss, F1-score, Area Under the ROC Curve (AUC), and precision. The results demonstrated progressive improvement in model performance, with accuracy increasing from 76.0% in round 1 to 87.7% in round 5, and AUC improving from 0.88 to 0.93, indicating enhanced classification capability over time. Similarly, F1-score and precision showed consistent growth, signifying improved balance between precision and recall and reduced false positives. The distributed training also showcased a decreasing loss trend, dropping from 1.018 to 0.891 across the rounds, reflecting better model convergence. Importantly, this study illustrated the practical viability of federated learning for privacy-centric applications, showing that high-performance machine learning models can be achieved without compromising data privacy. Future work can focus on scaling this approach to a larger number of clients, integrating differential privacy and secure aggregation techniques to further strengthen privacy guarantees, and comparing the hybrid model with alternative architectures to enhance model generalizability across heterogeneous data sources.

Keywords: Federated Learning, Privacy Preservation, Flower Framework, TensorFlow, Federated Averaging (FedAvg), Hybrid Deep Learning Model, Machine Learning Metrics, Aggregation

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

CCPA	California Consumer Privacy Act
CPM	Communication Privacy Management
DP	Differential Privacy
DP-SGD	Differentially Private Stochastic Gradient Descent
FL	Federated Learning
GAN	Generative adversarial network
GDPR	General Data Protection Regulation
GMW	Goldreich-Micali-Wigderson
HE	Homomorphic Encryption
LLMs	Large Language Models
ML	Machine Learning
MPC	Multi-Party Computation
PDPA	Personal Data Protection Act
PII	Personally Identifiable Information
PPDL	Privacy-Preserving Deep Learning
PPML	Privacy-Preserving Machine Learning
SMPC	Secure multi-party computation

Definition of Terms

Data Leakage	Data leakage refers to the unauthorized transfer of data from within an organization to an unintended, external recipient (Mimecast, 2024)
Privacy-Preserving	Techniques and methods that enable data analysis and machine learning while protecting the confidentiality of sensitive information. They aim to extract useful information from datasets containing personal information without violating the privacy of individuals in the dataset Xu et al. (2019)
Machine Learning	The examination of computer algorithms that enhance autonomously through experience (Al-Rubaie & Chang 2019)
Microservices	Small, autonomous services that work together. (Newman, 2015)
Personally Identifiable Information (PII)	According to (McCallister et al.,2010) PII is any information about an individual maintained by an agency, including (1) any information that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or biometric records; and (2) any other information that is linked or linkable to an individual, such as medical, educational, financial, and employment information
Prometheus Scraping Configuration	Defines how Prometheus collects metrics from targets, such as applications, services, or endpoints (Prometheus, 2014).

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr Allan Omondi and the course coordinator Prof Bernard Shibwabo for their indispensable assistance, support, and expertise during the formulation of this study. Their insights and support have been important in influencing this study.

I express my gratitude to the School of Computing and Engineering Studies at Strathmore University for supplying the essential resources and academic environment conducive to my research.

Many thanks to my classmates who have offered their time for discussions and provided valuable feedback on my ideas.

I wish to express my gratitude to my family and friends for their unwavering patience and encouragement, which have consistently motivated me.

This study could not have been realized without the collaborative support of these individuals and Strathmore University.

Dedication

This research is dedicated to my wife, son and daughter whose steadfast support and encouragement have been the cornerstone of this academic endeavor; to my father whose persistent reminders of achievable success sustain my motivation; and all who advocate for the potential of technology to enhance privacy and security in our increasingly interconnected society.

Chapter 1: Introduction

1.1 Background to the Study

The proliferation of machine learning technologies and the advent of the big data era have brought data protection to the forefront of technological and ethical concerns. The vast quantities of personal data utilized in training machine learning models necessitate robust data protection mechanisms (Al-Rubaie & Chang, 2019). Recent research has illuminated the extent of potential data leakage in machine learning models, underscoring the urgency of addressing this issue.

Empirical evidence suggests that large language models (LLMs) possess the capacity to memorize and potentially disclose significant portions of their training data. Kim et al. (2023) conducted a comprehensive analysis of GPT-2, revealing that up to 0.88% of the text generated by the model consisted of verbatim quotes from its training data. Furthermore, their study demonstrated that 1.53% of the samples produced by GPT-2 contained exact duplicates of training data when specifically prompted. These findings indicate a tangible risk of inadvertent data disclosure through model outputs.

The potential scale of personally identifiable information (PII) leakage is further elucidated by industry reports. IBM's Cost of a Data Breach Report 2021 indicates that the average expense of a data breach increased from $3.86 million in 2020 to $4.24 million in 2021, reaching a 17-year peak. Notably, 44% of these breaches involved the exposure of customers' personal data (IBM, 2021). These statistics serve to quantify the financial and reputational risks associated with inadequate data protection in machine learning systems.

The efficacy of traditional anonymization techniques has been called into question by recent research. Liu et al. (2020) demonstrated that 99.98% of Americans could be correctly re-identified in any dataset using merely 15 demographic attributes. This finding challenges the assumption that anonymization provides adequate protection for sensitive data and highlights the need for more sophisticated privacy-preserving methodologies in machine learning.

The inclusion of PII in machine learning datasets presents significant privacy risks, potentially facilitating the identification of specific individuals and misuse of their personal information

(Chamikara et al., 2022). A Deloitte survey indicates that 47% of customers feel

diminished control over their data, while 86% regard data privacy as an escalating worry (Deloitte, 2020).. These figures underscore the societal implications of potential PII leakage and the need for robust privacy safeguards.

Federated learning has emerged as a viable approach to privacy concerns inherent in centralized machine learning approaches (Kairouz et al., 2021). This methodology enables collaborative model development without centralizing data, maintaining data distribution across multiple devices or servers (Yang et al., 2019). McMahan et al. (2017) demonstrated that federated learning could achieve 97% accuracy on an image classification task while reducing data transmission from user devices by two orders of magnitude compared to centralized approaches.

The distributed architecture of federated learning aligns with the microservices paradigm, which decomposes systems into smaller, loosely coupled services communicating through well-defined interfaces (Xu et al., 2019). Empirical studies have shown that organizations adopting microservices architecture report significant improvements in development efficiency and system reliability (Newman, 2015). Federated learning integration with microservices architecture offers a viable foundation for the creation of PPML systems (Kairouz et al., 2021).

The convergence of these technologies, coupled with a focus on privacy-preserving techniques, offers a potential pathway to mitigate the risks of data leakage and PII exposure in machine learning systems. This strategy tackles both technical obstacles and aligns with increasing public apprehensions regarding data privacy and the changing regulatory landscape, including the General Data Protection Regulation (GDPR), the Personal Data Protection Act (PDPA) in Kenya, and the California Consumer Privacy Act (CCPA).

In conclusion, the evidence presented herein substantiates the critical need for privacy-preserving ML techniques, particularly in the context of microservices architectures. The potential for data leakage, the inadequacy of traditional anonymization methods, and the growing public concern over data privacy collectively underscore the importance of developing robust, privacy-centric approaches to machine learning. Additional research and development in this domain are essential to guarantee the responsible progress of machine learning technologies while safeguarding individual privacy rights.

1.2 Problem Statement

Since the emergence of big data and the broad application of Machine Learning (ML) across various fields, privacy concerns have become increasingly prominent (Al-Rubaie & Chang, 2019; Xu et al., 2019). Despite attempts to address these concerns through anonymization techniques and data protection regulations, effective privacy-preserving machine learning (PPML) solutions remain limited (Al-Rubaie & Chang, 2019; Liu et al., 2021; Kaissis et al., 2020). The deployment of ML in sensitive domains such as healthcare, finance, and national security has further heightened the importance of data protection (Kaissis et al., 2020; Enthoven & Al-Ars, 2021).

Machine learning systems in microservices architectures currently lack adequate privacy-preserving mechanisms, creating significant data leakage risks. The challenge is compounded by the increasing complexity of ML models and the distributed nature of modern computing architectures, introducing new vectors for potential data breaches (Qu et al., 2021). This issue affects multiple stakeholders, including individuals whose personal data is collected, organizations implementing ML systems, and regulatory bodies overseeing data protection (Al-Rubaie & Chang, 2019; Kairouz et al., 2021; Fereidooni et al., 2021). Recent high-profile data breaches and growing public awareness have led to increased scrutiny of ML practices and their implications for individual privacy (Shokri et al., 2017; Melis et al., 2019).

The tension between model utility and privacy preservation remains a significant obstacle in developing practical PPML solutions (Enthoven & Al-Ars, 2021). This necessitates the development of robust privacy-preserving techniques that can ensure data confidentiality without significantly compromising model performance or utility, particularly within microservices architectures where existing solutions prove inadequate.

1.3 Aim

This research aims to develop a privacy-preserving machine learning tool that will assist in detecting and preserving sensitive personal data within microservice architectures.

1.4 Research Objectives

i. To evaluate existing privacy-preserving techniques in ML systems.

ii. To develop a Federated Learning Model for microservices architectures.

iii. To Evaluate the Performance of the developed model.

1.5 Research Questions

i. What are the existing privacy-preserving techniques in machine learning systems, and how effective are they in preventing data leakage?

ii. How can a Federated Learning model be developed to enhance privacy preservation in microservices architecture?

iii. How does the performance of the developed Federated Learning model compare to traditional machine learning models in terms of accuracy, efficiency, and privacy protection?

1.6 Justification

It is crucial to conduct this research for several reasons. First, it will aid in the creation of workable and efficient PPML solutions that can guard people's right to privacy and stop the improper use of sensitive personal information. Second, it will make it possible for businesses to use ML technology more sensibly and in accordance with data protection laws, averting possible legal and reputational issues. Thirdly, it will increase consumer and societal trust and confidence in machine learning applications, which is essential to achieving machine learning's full potential across a range of industries. Ultimately, it will raise the bar for PPML research and lay the groundwork for more significant study in this crucial field.

1.7 Assumptions

This research assumes that organizations are willing to adopt PPML solutions if they are practical, effective, and do not significantly degrade the performance of ML models.

Stakeholders, including individuals and regulatory bodies, are willing to provide input and feedback on their privacy requirements and expectations.

The datasets and ML tasks used for testing and validation are representative of real-world scenarios and challenges.

1.8 Scope and Limitations

1.8.1 Scope

The proposed PPML tool development will focus on preserving privacy in supervised and unsupervised ML tasks, including classification, regression, and clustering. It will consider centralized and federated learning settings and aim to protect against common privacy attacks such as membership inference, model inversion, and data reconstruction. The research will be designed to integrate with popular ML libraries and platforms, such as TensorFlow and PyTorch, to facilitate adoption by practitioners.

1.8.2 Limitations

The research may face challenges in obtaining representative datasets and stakeholder feedback, which could limit the generalizability of the findings.

The proposed PPML tool development may not be applicable to all types of ML tasks and settings, such as reinforcement learning or online learning.

The research may be constrained by the computational resources available for testing and validation, which could limit the efficiency and scalability of the developed solutions.

To overcome these limitations, the research will aim to collaborate with industry partners and academic institutions to access diverse datasets and stakeholder perspectives. It will also focus on designing modular and extensible solutions that can be adapted to different machine learning tasks

and settings and leverage cloud computing resources and optimization techniques to improve scalability and effectiveness.

Chapter 2: Literature Review

2.1 Theoretical Framework

In the context of developing a privacy-preserving machine learning tool for mitigating against data leakage in microservices architectures, it is essential to understand the underlying theories that guide privacy management and communication. One such theory is the Communication Privacy Management (CPM) theory, which provides a framework for understanding how individuals manage their private information and the boundaries they establish around it (Petronio, 2002). This section will explore the key concepts and principles of CPM theory and their relevance to the research problem.

2.1.1 Communication Privacy Management (CPM) Theory

Developed by Sandra Petronio, Communication Privacy Management (CPM) theory clarifies how individuals manage their private information and establish boundaries around it. (Petronio 2002). It was once referred to as communication boundary management. The idea posits that individuals perceive ownership of their private information and determine access control according to established privacy regulations (Petronio, 2013). These regulations are shaped by multiple elements, including as cultural standards, individual values, and the dynamics of the connection between the person and the information recipient (Petronio, 2002).

2.1.2 Ownership and Control of Information

One of the core principles of CPM theory is that individuals have a sense of ownership over their private information (Petronio, 2002). This ownership is not necessarily legal but rather psychological, as individuals feel that they have the right to control access to their personal information (Petronio, 2013). When individuals disclose private information to others, they grant co-ownership of that information to the recipient, which comes with certain expectations and responsibilities (Petronio, 2002). This concept of ownership and control is further supported by recent work, such as the study by Gruzd and Hernández-García (2018), which found that individuals' sense of ownership over their private information extends to their online data and that they expect to have control over how that data is collected, used, and shared.

2.1.3 Privacy Rules

According to CPM theory, individuals develop a set of privacy rules that guide their decisions about how to manage their private information (Petronio, 2002). These rules are based on various criteria, such as the nature of the information, the relationship between the individual and the recipient, and the potential risks and benefits of disclosure (Petronio, 2013). Privacy rules can be explicit or implicit and may change over time as circumstances evolve (Petronio, 2002). The importance of privacy rules in managing private information is further highlighted by recent research, such as the study by Liu et al. (2020), which found that individuals' privacy rules are influenced by their cultural background and that privacy rules vary across different contexts and relationships.

2.1.4 Boundary Coordination

CPM theory underscores the significance of border coordination in the management of private information. (Petronio, 2002). When individuals provide confidential information to outsiders, they establish boundaries around that information and expect the recipient to respect those boundaries (Petronio, 2013). Boundary coordination involves negotiating the terms of co-ownership and establishing clear expectations about how the information will be handled (Petronio, 2002). The role of boundary coordination in managing private information is further explored by recent work, such as the study by Wang et al. (2021), which found that effective boundary coordination in online communities requires clear communication and shared understanding among members about the privacy rules and expectations surrounding shared information.

2.1.5 Privacy Turbulence

Privacy turbulence occurs when there is a breakdown in boundary coordination, and private information is mishandled or disclosed in ways that violate the individual's privacy rules (Petronio, 2002). This can happen when the recipient of the information fails to respect the established boundaries or when there is a misunderstanding about the terms of co-ownership (Petronio, 2013). Privacy turbulence can lead to feelings of betrayal, mistrust, and a loss of control over one's private information (Petronio, 2002). The impact of privacy turbulence on individuals' willingness to disclose private information is further supported by recent research, such as the study by Li et al.

(2019), which found that individuals who experienced privacy turbulence on social media were less likely to share personal information in the future and were more likely to engage in privacy protection behaviors.

2.1.5 Co-Ownership and Collective Privacy Management

CPM theory recognizes that privacy management is not always an individual process but can also involve co-ownership and collective privacy management (Petronio, 2002). When individuals disclose private information to a group, such as a family or an organization, the group becomes collectively responsible for managing that information (Petronio, 2013). This requires coordination and communication among group members to ensure that privacy rules are respected and boundaries are maintained (Petronio, 2002). The importance of co-ownership and collective privacy management is further highlighted by recent work, such as the study by Chen and Sharma (2021), which found that effective collective privacy management in online health communities requires clear privacy policies, active moderation, and a sense of shared responsibility among members.

2.1.6 Feedback Mechanisms

CPM theory also highlights the importance of feedback mechanisms in managing private information (Petronio, 2002). Feedback mechanisms allow individuals to monitor how their private information is being handled and to make adjustments as necessary (Petronio, 2013). This can include seeking clarification about the terms of co-ownership, renegotiating boundaries, or taking steps to protect one's privacy in response to privacy turbulence (Petronio, 2002). The role of feedback mechanisms in managing private information is further supported by recent research, such as the study by Zhang et al. (2020), which found that individuals who received timely and informative feedback about how their private information was being used were more likely to feel in control of their privacy and to continue sharing information with the organization.

2.1.7 Relevance to the Research Problem

CPM theory provides a valuable framework for understanding the privacy concerns and challenges associated with developing a privacy-preserving machine learning probing tool for mitigating data leakage in microservices architectures. The theory highlights the importance of

respecting individuals' ownership and control over their private information and the need for clear boundary coordination and feedback mechanisms to prevent privacy turbulence.

In the context of machine learning and microservices architectures, CPM theory underscores the need for robust privacy protections and transparent communication about how individuals' data will be collected, used, and shared. It also suggests that individuals should have a degree of control over their data and the ability to set boundaries around how it is used.

Furthermore, CPM theory highlights the risks associated with data leakage and the potential for privacy turbulence when private information is mishandled or disclosed without permission. This underscores the importance of developing effective tools and strategies for mitigating against data leakage in microservices architectures.

Overall, CPM theory provides a valuable lens through which to examine the privacy concerns and challenges associated with developing a privacy-preserving machine learning probing tool. It offers insights into how individuals manage their private information and the importance of respecting their ownership and control over that information. By applying the principles of CPM theory to the research problem at hand, we can develop a more nuanced understanding of the privacy issues at stake and design more effective solutions for preserving privacy in machine learning and microservices architectures.

2.2 Secure Multi-Party Computation Theory

Secure multi-party computation (SMPC) is another an exemplary theory that establishes methodologies for participants to collaboratively compute a function based on their inputs while maintaining their confidentiality (Yao, 1982). It is a subfield of cryptography but unlike traditional cryptography, it ensures security and integrity of the communication or protects participants privacy from each other. Below are some of the concepts around SMPC and their relevance to the research problem.

2.2.1 Fundamentals of Secure Multi-Party Computation

Secure Multi-Party Computation (SMPC) is a cryptographic framework that allows several participants to collaboratively compute a function based on their inputs while maintaining the confidentiality of those inputs. The concept was initially presented by Yao (1982) in his

foundational study on safe two-party computation. Since that time, SMPC has developed into a robust instrument for privacy-preserving computations across multiple domains, including machine learning and distributed systems such as microservices architectures.

The fundamental idea behind SMPC is to allow parties to collaborate on computations without revealing their individual data. As Lindell (2020) explains, "The basic idea is that the parties run a protocol that simulates a trusted party who receives the inputs from all parties, computes the function, and returns the output" (p. 1). This approach is particularly relevant in today's data-driven world, where privacy concerns often hinder collaborative efforts in machine learning and data analysis.

In the context of microservices architectures, SMPC provides a promising solution to mitigate data leakage risks. Microservices often handle sensitive data across distributed systems, and SMPC protocols enable secure computations without the need to centralize or expose raw data (Zhao et al., 2019). This capability is especially useful in scenarios where different microservices manage distinct types of user data but require a holistic analysis for effective machine learning. For example, SMPC could allow multiple microservices to collaboratively compute machine learning models or perform data analysis without exposing sensitive data (Bogdanov et al., 2018).

Consider an e-commerce platform built on microservices: one service might handle user profiles, another manages purchase history, and a third deals with product recommendations. SMPC could enable these services to jointly train a recommendation model without exposing raw user data to each other. This preserves user privacy while still leveraging the full dataset for improved accuracy (Kumar et al., 2020).

2.2.2 Mathematical Description of the General MPC Protocol

The general Multi-Party Computation (MPC) protocol can be mathematically described as follows:

Consider there be n parties P_1, ..., Pn, each with private input x_i. They wish to compute a function $f(x_1, ..., xn) = (y_1, ..., yn)$, where y_i is the output received by party P_i.

The protocol typically consists of three phases:

a) **Input sharing:** Each party P_i splits its input x_i into shares $(x_{i1}, ..., x_{in})$ such that $x_i = x_{i1} + ... + x_{in} \pmod m$, where m is a large prime number.

b) **Computation:** Parties perform local computations on their shares and exchange intermediate results.

c) **Output reconstruction:** Parties combine their local results to obtain the final output.

Mathematically, we can represent this as:

$$f(x_1, ..., xn) = \text{Reconstruct}(\text{Compute}(\text{Share}(x_1), ..., \text{Share}(xn)))$$

Where Share, Compute, and Reconstruct are protocol-specific functions (Cramer et al., 2015). The security of MPC protocols is typically proven in one of two models:

Semi-honest (honest-but-curious) model: Parties adhere to the protocol but may attempt to get more information from the messages they receive.

Malicious model: Parties may deviate arbitrarily from the protocol.

In the semi-honest model, security is typically characterized through the simulation paradigm. A protocol is deemed secure if a simulator can produce a view that is indistinguishable from the actual execution of the protocol, based solely on the input and output of a participant (Goldreich, 2009).

2.2.3 Key SMPC Protocols and Their Mathematical Foundations

Several key protocols form the foundation of modern SMPC:

a) Yao's Garbled Circuits: Introduced by Yao (1986), this protocol allows two parties to securely compute any function represented as a boolean circuit. The basic idea is to "garble" the circuit and its inputs, allowing computation on encrypted values.

Mathematically, for a circuit C and input x, the garbled circuit G(C) and garbled input G(x) are created such that:

$G(C)(G(x)) = C(x)$

b) GMW (Goldreich-Micali-Wigderson) Protocol: This protocol, developed by Goldreich et al. (1987), uses secret sharing and operates on arithmetic circuits. For each gate in the circuit, parties perform local computations and engage in oblivious transfer to compute the gate's output.

c) Secret Sharing based protocols: These protocols, like the one proposed by Ben-Or et al. (1988), use techniques like Shamir's Secret Sharing. In Shamir's scheme, a secret s is shared among n parties by creating a polynomial f(x) of degree t-1 (where t is the threshold) such that $f(0) = s$, and each party i receives a point (i, f(i)).

These protocols have been extended and optimized for various scenarios, including privacy-preserving machine learning in distributed environments (Mohassel & Zhang, 2017).

2.2.4 SMPC in Distributed Machine Learning and Microservices

The utilization of SMPC in distributed machine learning and microservices architectures has garnered considerable attention owing to heightened privacy concerns and legal mandates. In the realm of machine learning, Secure Multi-Party Computation (SMPC) facilitates collaborative model training across several participants while preserving the confidentiality of their raw data..

Mohassel and Zhang (2017) introduced SecureML, a framework for privacy-preserving machine learning via secure multi-party computation (SMPC). Their methodology enables data proprietors to collaboratively train diverse machine learning models (e.g., linear regression, logistic regression, neural networks) while maintaining the confidentiality of their respective datasets.

In microservices architectures, SMPC can be used to secure inter-service communications and

data aggregation. Baracaldo et al. (2019) demonstrated how SMPC can be integrated into a microservices-based federated learning system to protect both model parameters and individual contributions during the training process.

Adapting SMPC protocols for microservices communication presents unique challenges, including handling dynamic scaling and maintaining performance under high-throughput requirements. Recent work by Poddar et al. (2020) addresses some of these issues by proposing a system called Senate, which provides SMPC as a service in cloud environments, making it more amenable to microservices architectures.

2.2.5 Performance and Security Considerations

While SMPC offers strong privacy guarantees, it comes with significant performance overhead. The communication and computational complexity of SMPC protocols are key considerations, especially in microservices where low latency is often crucial.

Lindell (2020) provides a comprehensive analysis of the performance challenges in SMPC, noting that the overhead can be substantial, particularly for complex functions or large numbers of parties. However, recent advancements have significantly improved efficiency. For instance, Keller et al. (2018) proposed a protocol that achieves unprecedented performance for secure three-party computation.

Security guarantees in SMPC are typically proven in the semi-honest or malicious adversary models. While the malicious model offers stronger security guarantees, it comes at a higher performance cost. Practical implementations often use the semi-honest model with additional safeguards (Zhao et al., 2019).

Scalability in microservices environments presents unique challenges for SMPC. Traditional SMPC protocols are not designed for the dynamic nature of microservices, where services may scale up or down rapidly. Addressing these challenges requires novel approaches that balance security, performance, and scalability (Poddar et al., 2020).

In the context of this research on privacy-preserving machine learning probing tools, SMPC can play a crucial role. Machine learning probes are often used to analyze model behavior and detect potential issues, including data leakage. By incorporating SMPC into these probing tools, you can

ensure that the probing process itself doesn't introduce new vulnerabilities. This will help analyze model outputs across multiple microservices without exposing the individual service's data or model parameters. This approach could help detect data leakage between services while maintaining the privacy guarantees of each service (Mohassel & Zhang, 2017).

2.3 Existing PPML Techniques

In the privacy-preserving machine learning (PPML) field, various strategies have emerged as viable options for safeguarding sensitive data in distributed systems such as microservice architectures. This section explores three main approaches: Federated learning, differential privacy, and homomorphic encryption.

2.3.1 Federated Learning

Federated Learning (FL) signifies a transformative approach in machine learning, facilitating model training over numerous decentralized edge devices or servers that possess local data samples, without the necessity of data exchange (Kairouz et al., 2021). In the realm of microservices, federated learning enables each service to contribute to a collective model while maintaining localized data, thus alleviating privacy concerns linked to data centralization.

The FL process typically involves initializing a global model, distributing it to participating microservices, conducting local training, securely aggregating updates, and iterating until convergence. This approach has gained significant traction in privacy-sensitive domains. For instance, Xu et al. (2021) demonstrated FL's effectiveness in healthcare, enabling collaborative learning across multiple hospitals without sharing patient data. McMahan et al. (2017) demonstrated its success in mobile keyboard prediction tasks without compromising user privacy.

While FL offers robust privacy protection by keeping data local, reducing the risk of centralized data breaches and allowing collaboration across organization boundaries, it is not without challenges. Enthoven and Al-Ars (2020) highlight communication overhead as a significant concern, especially in bandwidth-constrained environments. Moreover, Lu et al. (2020) points out vulnerabilities to certain types of attacks, such as model inversion, which can potentially compromise privacy. Other challenges to this technique is communication overhead, this is a big challenge for huge models with frequent updates and challenges in dealing with non-IID

(Independent and Identically Distributed) data across microservices.

Despite these challenges, FL's ability to enable collaborative learning while preserving data locality makes it a compelling choice for privacy-preserving ML in microservices architectures. Its effectiveness in maintaining data privacy while allowing model improvement has been demonstrated across various domains, from mobile applications to healthcare systems.

2.3.2 Differential Privacy

Differential Privacy (DP) provides a mathematical framework for assessing and alleviating privacy hazards in data analysis and machine learning activities. It entails the introduction of meticulously adjusted noise to data or calculations to conceal individual contributions while preserving overall statistical utility (Dwork et al., 2014).

In recent years, DP has seen significant advancements and practical implementations. Abadi et al. (2016) introduced DP in deep learning, demonstrating its applicability to complex ML models. Building on this, Xu et al. (2020) proposed an adaptive approach to DP in federated learning, showcasing how DP can be effectively combined with other PPML techniques.

The implementation of DP in microservices typically involves defining a privacy budget, conducting sensitivity analysis, selecting appropriate noise mechanisms, and tracking privacy loss across operations. While DP offers strong, provable privacy guarantees, it presents challenges in balancing privacy and utility. Zhao et al. (2020) explored this trade-off in the context of machine learning, providing insights into optimizing DP parameters for different types of ML tasks.

DP's strength lies in its ability to provide quantifiable privacy guarantees and its flexibility in application across various stages of the ML pipeline. However, as noted by Jayaraman and Evans (2019), achieving meaningful privacy guarantees often requires adding substantial noise, which can significantly impact model utility, especially in complex, multi-step ML processes common in microservices architectures.

2.3.3 Homomorphic Encryption

Homomorphic Encryption (HE) represents a cryptographic approach that allows computations on encrypted data without access to the decryption key. This property makes HE particularly attractive

for privacy-preserving computations in distributed systems like microservices, where data may need to be processed across multiple, potentially untrusted services.

Recent advancements have made HE more practical for ML applications. Chen et al. (2019) demonstrated the use of HE in privacy-preserving deep learning, showing how neural networks can operate on encrypted data. Furthermore, Bost et al. (2020) presented optimizations that significantly improve the efficiency of HE in machine-learning contexts, addressing some of the performance concerns associated with this technique.

Implementing HE in a microservices architecture typically involves key generation and distribution, data encryption, development of homomorphic operations compatible with ML algorithms, and secure result decryption. While HE provides strong privacy guarantees by allowing computations on encrypted data, it comes with substantial computational overhead. Jiang et al. (2018) highlighted these performance challenges, particularly for complex ML operations, and proposed optimizations for specific use cases.

The strength of HE lies in its ability to enable secure multi-party computation and its suitability for scenarios requiring high levels of data confidentiality. However, its limitations in supporting non-linear operations, which are common in many ML algorithms, and the challenges in key management across distributed services, as discussed by Zerka et al. (2020), need careful consideration when implementing HE in microservices architectures.

In conclusion, each of these PPML techniques offers unique advantages and faces specific challenges in the context of microservices architectures. The choice and combination of these techniques depend on the specific privacy requirements, computational resources, and the nature of the ML tasks in the microservices ecosystem. As research in this subject progresses, we can expect additional advances in the efficiency and application of these techniques, potentially leading to more robust and practical privacy-preserving machine learning solutions for microservices systems.

2.4 Empirical Framework

This research has looked at the work that has been done before to gain research knowledge, decide on the improvements that can be made for a better solution. The table below has been used to

summarize these studies. Table 1.1 provides a comprehensive overview of the empirical framework for Privacy-Preserving Machine Learning in Microservices Architectures. It includes key information about various studies, their focus areas, the models and techniques used, main findings, and potential areas for improvement.

Table 1.1: Existing Studies

Author(s)	**Focus Area**	**Models/Techniques Used**	**Key Findings**	**Potential Improvements**
Fan et al. (2021)	Federated Evaluation	FedEval framework	Developed a comprehensive evaluation platform for federated learning systems; Includes privacy and security assessment tools	Adapt FedEval for microservices architectures; Enhance probing capabilities for distributed systems

Author(s)	Focus Area	Models/Techniques Used	Key Findings	Potential Improvements
Mohassel et al. (2020)	Secure Inference Probing	SecureML probes for neural network inference	Detected potential model inversion attacks in distributed inference services; Improved privacy guarantees for model deployment	Enhance probe efficiency for large-scale models; Develop probes for other ML algorithms
Rezaei & Liu (2021)	Auditing Privacy Defenses	Scalable auditing framework, Differential privacy analysis	Proposed methods to audit privacy defenses in ML models. Revealed limitations in existing differential privacy implementations	Extend auditing framework to distribute and microservices-based ML systems

Author(s)	Focus Area	Models/Techniques Used	Key Findings	Potential Improvements
Nasr et al. (2019)	Privacy Meter	White-box membership inference attacks	Developed a tool to measure the privacy risks of ML models; Demonstrated effectiveness on various models architectures	Adapt privacy meter for distributed and federated learning scenarios
Melis et al. (2019)	Inference Attacks in Collaborative Learning	Gradient-based attacks, Multi-task learning	Revealed vulnerabilities in federated learning to inference attacks; Proposed defense mechanisms	Improve privacy guarantees in distributed learning; Develop probing tools for federated systems
McMahan et al. (2017)	Federated Learning	Federated Averaging algorithm	Effective training on decentralized data; Reduced communication costs	Enhance privacy guarantees; Improve model convergence speed

Author(s)	Focus Area	Models/Techniques Used	Key Findings	Potential Improvements
Bonawitz et al. (2019)	Federated Learning at Scale	Secure aggregation, Large-scale FL system	Scalable production system for FL; Addressed device availability and network reliability issues	Reduce Computational Overhead; Improve model quality for heterogeneous data
Abadi et al. (2016)	Differential Privacy	DP-SGD (Differentially Private Stochastic Gradient Descent)	Training deep learning models with strong privacy guarantees; Demonstrated privacy-utility trade-off	Reduce accuracy loss; Improve epsilon selection
Gilad- Bachrach et al. (2016)	Homomorphic Encryption	CryptoNets (Neural Networks on encrypted data)	Feasibility of privacy-preserving inference using encrypted data	

Author(s)	Focus Area	Models/Techniques Used	Key Findings	Potential Improvements
Bogdanov et al. (2016)	Secure Multi-Party Computation	Share mind framework	Privacy-preserving data analysis across organizations; Case study on tax data analysis	Improve performance for large-scale computations; Enhance usability
Bogdanov et al. (2016)	Hybrid Approaches	Combination of FL and DP	Analysis of trade-offs between privacy, utility, and performance in hybrid systems	Develop more comprehensive evaluation metrics; Improve integration of different PPML techniques
He et al. (2020)	Model Inversion Attacks	Generative adversarial network (GAN) based attacks	Showed effectiveness of model inversion in recovering private training data; Proposed defensive distillation as a countermeasure	Develop probing tools to detect model inversion vulnerabilities in deployed models

2.5 Interactions and Processes

2.5.1 Data Flow

The flow of data in microservices-based ML systems requires stringent security measures. Homomorphic encryption, as demonstrated by Gu et al. (2021), secures data exchanges between microservices without compromising on the ability to perform meaningful computations.

Similarly, Mo et al. (2021) illustrate how federated learning can facilitate privacy-preserving data aggregation from distributed nodes, maintaining privacy even as data flows across multiple services.

2.5.2 Model Training and Deployment

Training and deploying ML models in a privacy-preserving manner within microservices requires innovative techniques. Wang et al. (2020) demonstrate how federated learning can be adapted for distributed training, ensuring that sensitive data remains protected. Secure model updates, a core feature of Mo et al.'s (2021) PPFL framework, ensure that only verified and privacy-preserving updates are applied. Techniques like homomorphic encryption, as illustrated by Gu et al. (2021), can also be employed to enable privacy-preserving inference in distributed systems.

2.5.3 Privacy Protection Mechanisms

Incorporating privacy protection mechanisms such as DP, SMPC, and HE is critical to the secure functioning of ML systems within microservices. The successful application of these techniques has been demonstrated by several studies, including Wang et al. (2020), Chen et al. (2022), and Gu et al. (2021). Federated learning systems also rely on secure aggregation protocols to protect privacy (Mo et al., 2021; Enthoven & Al-Ars, 2020). However, the management of privacy budgets in distributed systems remains an area that warrants further exploration.

2.5.4 Probing and Monitoring

Probing and monitoring ML systems deployed in microservices environments are crucial for

maintaining ongoing privacy protection. Zhang et al. (2022) and Li et al. (2023) provide essential tools for assessing vulnerabilities and detecting privacy leaks. Future research will need to focus on ensuring the integrity of models and data in real-time, as this remains an underdeveloped area in the current literature.

2.6 Challenges and Considerations

2.6.1 Performance vs. Privacy Trade-offs

Balancing the trade-offs between performance and privacy is a significant challenge in microservices-based ML systems. Wang et al. (2020) explore this issue in edge computing, which shares parallels with microservices environments, highlighting that privacy-preserving techniques often introduce computational overhead. Homomorphic encryption, for instance, while highly secure, can significantly impact system performance (Gu et al., 2021).

2.6.2 Scalability Issues

As microservices-based ML systems scale, privacy-preserving techniques face significant challenges. Enthoven and Al-Ars (2020) discuss the difficulties of scaling federated learning systems, which are also relevant to microservices architectures. Probing large-scale distributed systems for vulnerabilities, as attempted by Zhang et al. (2022) and Li et al. (2023), becomes increasingly complex as the system grows.

2.6.3 Attack Vectors

Microservices-based ML systems are vulnerable to a range of attack vectors. While the reviewed literature does not explicitly cover attacks like model inversion and membership inference in microservices, these remain significant concerns. Data reconstruction attacks in distributed settings, addressed by Gu et al. (2021) and Mo et al. (2021), highlight the need for robust defenses against adversarial actors.

2.6.4 Regulatory Compliance

Ensuring adherence to data protection requirements, such as GDPR and CCPA, is essential for microservices-based machine learning systems. Although this issue is not explicitly addressed in the reviewed literature, it is clear that future research must explore how privacy-preserving techniques can be aligned with regulatory requirements. Enhancing auditability and transparency

in distributed ML systems will be essential for achieving this goal.

2.7 Conceptual Framework

To address the research problem identified in this study, a conceptual framework was developed to guide the implementation of privacy-preserving fraud detection using Federated Learning. The key components and workflow of the framework are described in Figure 2.1 below.

The conceptual framework for this study provides a structured pathway illustrating how privacy-preserving machine learning can be employed for fraud detection using Federated Learning (FL). It demonstrates how key components—including data preprocessing, hybrid model development, SMOTE balancing, and the deployment of the FL environment using Flower and TensorFlow interact systematically to solve the research problem of ensuring accurate fraud detection while preserving data privacy. The framework underscores the integration of federated learning with privacy-preserving techniques and hybrid machine learning models to enhance detection accuracy without exposing sensitive client data.

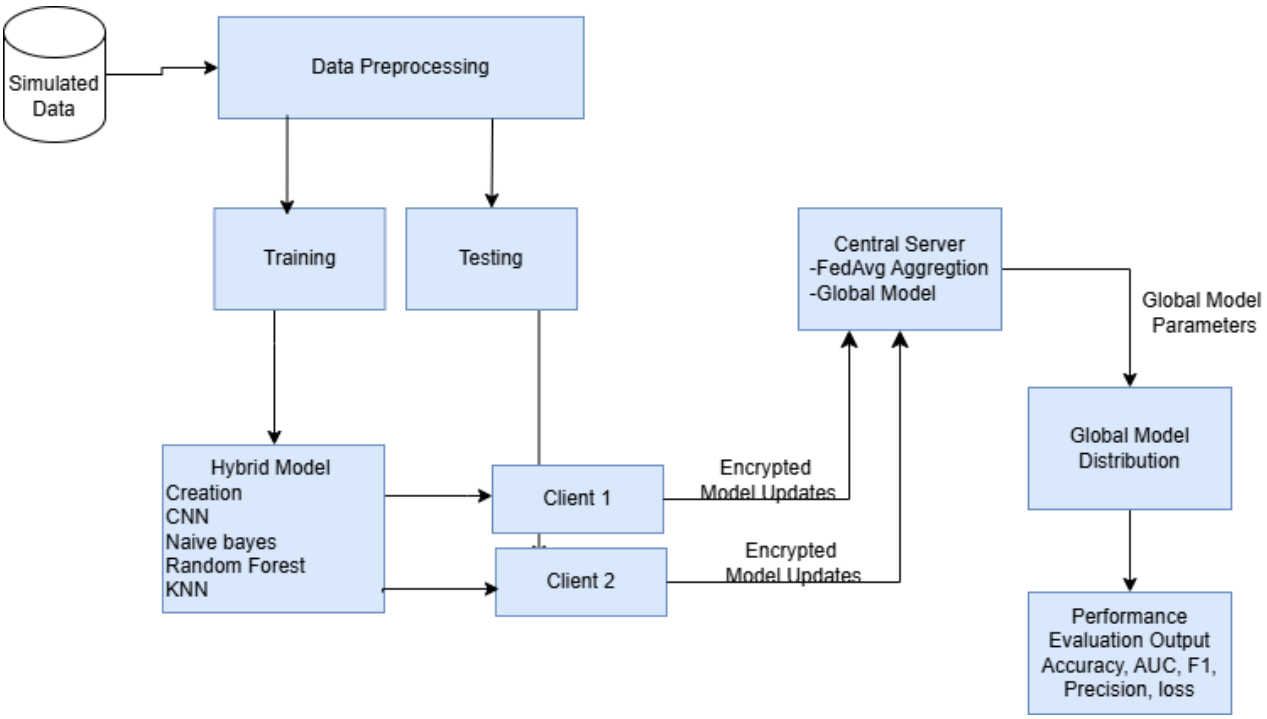


Figure 2.1: Conceptual Framework

Chapter 3: Research Methodology

3.1 Introduction

This chapter describes the methodology for combating data leakage in microservices using Federated Learning (FL). In its structure, the chapter aims to provide the process through research design, data collection, model development, and evaluating metrics. The basic idea behind the methodology itself aims to keep the privacy of the data intact while allowing effective machine learning in a distributed microservices environment.

3.2 Research Design

Dannels (2018) defines the research design as the procedural plan used by the researcher to answer the questions in a valid, objective, accurate, and economic manner. This study adopts an experimental research design to implement and evaluate Federated Learning (FL) in microservices architecture (Xie & Li, 2025). This approach is chosen because it allows for a controlled environment where the impact of different privacy-preserving mechanisms can be systematically analyzed. By simulating real-world financial transactions, the study ensures that the effects of Federated Learning, compared to traditional centralized machine learning models, are accurately measured under various privacy-enhancing techniques such as Differential Privacy and Homomorphic Encryption (Choi et al., 2024). The experimental design also enables the isolation of key variables, allowing for the manipulation of independent variables, such as different privacy-preserving methods, to observe their effects on dependent variables like model accuracy, privacy leakage risk, and computational efficiency. This ensures that any observed improvements in privacy protection can be directly attributed to Federated Learning rather than external factors.

Additionally, this research design facilitates a comparative evaluation of Federated Learning against traditional machine learning models, assessing aspects such as data leakage risks, model accuracy, and computational overhead. Such a structured comparison provides empirical evidence on whether Federated Learning is a feasible and superior privacy-preserving approach for microservices. Given the sensitivity of financial data, replicating real-world scenarios in an experimental setup allows the study to offer practical insights into the challenges and effectiveness of deploying Federated Learning in microservices architectures. Through this approach, the research aims to provide structured implementation, rigorous evaluation, and empirical validation of Federated Learning as a robust privacy-preserving technique in the financial sector.

3.3 Data Collection

Given the severe privacy implications of using real transaction data, especially given the extent to which financial transaction data is sensitive and regulated, this study leverages synthetic financial transaction data as a pragmatic solution (Blumenstock & Kohli, 2023). The use of synthetic data permits the necessary realism for effective model evaluation, while also removing risks associated with data leakage, identity exposure, and violations of regulations. The synthetic dataset is created to be similar to real-world financial transactions, and preserve the properties (characteristics, structure, and statistical properties) associated with real financial transaction data. The synthetic dataset contains a total of 2,000 records, with each record representing a single financial transaction. The synthetic dataset is created to represent real banking/finance activity and attempts to capture the main characteristics of the relevant attributes that are included in real-world financial transaction datasets.

The use of synthetic financial transaction data ensures that privacy and compliance requirements are met while still providing a dataset suitable for evaluating Federated Learning in a microservices architecture (Karampasi et al., 2024). The dataset is designed to introduce natural variations, mirroring real-world financial behaviors and fraud patterns. This allows for the development, training, and testing of machine learning models under realistic conditions, without compromising customer privacy or breaching financial regulations.

3.2.1 **Dataset Variables**

The dataset consists of the following key variables as highlighted in table 3.1.

Table 3.1:Dataset Description

FEATURE NAME	DESCRIPTION	TYPE
TRANSACTION ID	Unique identifier for each transaction	Categorical
TRANSACTION TYPE	Type of transaction (TRANSFER, CASH_OUT, etc.)	Categorical
TRANSACTION AMOUNT	Amount involved in the	Numerical

	transaction	
TIMESTAMP	Date and time of the transaction	Timestamp
ACCOUNT TYPE	Type of account initiating the transaction	Categorical
TRANSACTION MODE	Payment method used (Online, ATM)	Categorical
BALANCE BEFORE	Account balance before the transaction	Numerical
BALANCE AFTER	Account balance after the transaction	Numerical
CURRENCY	Currency used in the transaction	Categorical
TRANSACTION FREQUENCY	Number of transactions per month per user	Numerical
DEVICE TYPE	Device used for the transaction (PC, Mobile)	Categorical
IP LOCATION REGION	Geographic location of the transaction	Categorical
PREVIOUS FRAUD FLAG	Indicator if the account was previously involved in fraud	Boolean

3.2.2 Sampling Strategy Implementation

This study employs stratified sampling to ensure that synthetic financial transaction dataset accurately reflects the diverse characteristics of real-world financial data (Rahman et al., 2022). This method, enhances sample representativeness, which yields more accurate and generalizable findings. In stratified sampling, the population is divided into distinct strata based on some characteristics and randomly drawn from each stratum (Lu et al., 2023). The purpose is to assure

that each stratum is represented in the sample accurately, which improves the accuracy of statistical estimates. Given the complexity of financial transactions, the study dataset includes various strata such as transaction types, amounts, user behaviors, and fraud statuses. By applying stratified sampling, we ensure that each of these categories is adequately represented, allowing the models to learn from a comprehensive dataset that mirrors real-world scenarios. Implementing stratified sampling in this study ensures that the simulated financial transaction dataset is both comprehensive and representative. This approach enhances the reliability of the findings and supports the development of robust, privacy-preserving machine learning models within a federated learning framework.

3.4 Data Preprocessing

While developing the synthetic financial transaction dataset for model training, there were several fundamental preprocessing steps that took place in order to establish data quality and model performance. There was the consideration of missing values which, even if small, could pose issues within the synthetic dataset. There was a focus on feature encoding converting categorical variables to numerical formats fit for machine learning algorithms (Dahouda & Joe, 2021). Depending on the feature of focus and the requirements of the model, label encoder method was used to convert categorical variables such as transaction type, account type, and device type into numerical values.

Subsequently, the study performed feature scaling on numerical attributes such as transaction amount and balance before the transaction (Ozsahin et al., 2022). By applying Standardization, the study ensured that these features were on a comparable scale, which is crucial for the convergence and performance of many machine learning models. The research also addressed duplicate entries within the dataset. These duplicates can skew the data and lead to biased results. To eliminate redundant entries, we utilized the duplicate function in Pandas, which identifies and removes duplicate rows based on key attributes, thereby ensuring the dataset's accuracy.

Fraud detection datasets have class imbalance with fewer fraud cases than legitimate transactions; therefore, the Synthetic Minority Over-sampling Technique (SMOTE) was used to balance it (Ghaleb et al., 2023). SMOTE improves imbalances caused by smaller numbers of minority class observations by introducing synthetic examples of the minority class so it presents more frequently in the data (Dablain et al., 2022). By balancing the class representation, this technique improves the sensitivity of fraud detection

models to fraudulent activity while reducing or eliminating bias toward the majority class observations. Additionally, the dataset was split into training and testing sets. 80 percent of the dataset was used for training and 20 percent for testing purposes to properly evaluate Federated Learning System. This decision allowed us to train the model with 80 percent of the dataset while still maintaining an appropriate representation of a valid sample in the 20 percent of the testing data for evaluating for bias in predicting fraudulent behavior. Therefore, by incorporating the appropriate preprocessing steps such as encoding categorical features, scaling numerical features, balancing imbalanced class sets, and properly dividing the appropriate training and testing datasets for unbiased evaluation, this study attempted to provide improved model robustness and accuracy when utilizing robust federated learning system.

3.5 Federated Learning Implementation

In this research, Federated Learning (FL) framework was implemented, tailored for financial institutions, ensuring that model training occurred locally on each institution's servers without the need to share raw data (Awosika et al., 2024). This approach aligns with the privacy requirements inherent in the financial sector, allowing institutions to collaboratively enhance machine learning models while safeguarding sensitive information. The study implementation adopted the client-server architecture, a prevalent structure in FL systems (Shanmugam et al., 2023). In this setup, a central server coordinates the training process by distributing the global model to various client nodes. Each client then updates the model using its local data and sends the updates back to the server for aggregation (Shanmugam et al., 2023). This model is advantageous for its simplicity and scalability, making it suitable for environments with numerous heterogeneous devices.

3.5.1 Federated Learning Model

The algorithm selected for this implementation was the federated average (FedAvg), which aggregates the parameters of all received clients' models into a single neural network (He et al., 2021). In FedAvg, each client trains the global model locally on its own data for several iterations and sends the new model parameters back to the central server (Liu et al., 2021). The server then aggregates the parameters by taking a weighted average based on the size of the client's data set. This results in reduced communication overhead and good performance with data that is not IID (Independent and Identically Distributed) across clients. Privacy of the data is also kept in the FedAvg FL algorithm, which allows for the combined training of a model across many clients or devices.

3.5.2 Training Process and Privacy Preservation

The training procedures in the federated learning system include several important operations to ensure model quality and respect for data privacy. The first step was to initialize the global hybrid machine learning model in a central server. This model may be pre-trained or can start with randomly initialized parameters. The central server manages the training process and ensures that all client nodes are using the same model configuration. After initializing the model, the server sends the global model to a subset of randomly selected clients, which could be edge devices such as mobile phones or servers at organizations. The server does not send models to all clients on which it could run the training step but chooses a subset of the clients to communicate more effectively through training and minimize the use of computational resources. After obtaining the model from the server, the selected client nodes locally train the model on their local data, being careful to limit the local training to only a few epochs or mini-batch updates to prevent overfitting to their local data and to maintain model consistency among the nodes. Most importantly, each client node generates model updates such as gradients without sending their raw data from the client devices. Once the local training occurs, the client nodes sends their model updates back to the central server. The model updates included the learned knowledge from the local data and may be the model parameters or the computed gradients. This is important to maintain privacy, as no sensitive information is sent.

Consequently, the central server combined the updates to create a new global model. The common approach is using Federated Averaging (FedAvg), which averages the model updates while considering each client's size of the local datasets (Collins et al., 2022). By using the size of the dataset on the client nodes as a weighting factor to compute the average of the updates, the concatenation of model updates weighs more for clients with more datasets. These established steps were cycled through additional rounds of federated learning, which represents a round of federated learning. After multiple rounds of federated learning, during the learning process, the global model improved in performance as the global model sees content from previously diverse data distributions from all of the client nodes (Sun, Li, & Wang, 2022). This process repeated until convergence or desired performance of the model was reached.

3.6 Model Architecture and Training

3.6.1 Hybrid Machine Learning Model

In the current research study, a multi-layered Convolutional Neural Network (CNN) model was utilized

to classify fraudulent transactions in addition to classical machine learning models, including Naïve bayes, K-Nearest Neighbors, and Random Forest. Neural networks have proven effective in identifying complex patterns of information from within financial data, thus finding wide application in fraud detection environments, such as implementing Graph Neural Networks (GNNs) which have been shown to improve detection performance by capturing complex relationships among transactions in data (Jin et al., 2023). For the sake of having a comprehensive evaluation, additional models were also tested for the purpose of benchmark tests such as Logistic Regression, Decision Trees, and XGBoost. Each model provides different benefits such as interpretability with Logistic Regression, and Decision Trees allow the user to make a subtle decision, while XGBoost is traditionally known for having performance efficiency in classification models.

3.6.2 Training Configuration

In the training process, the Adam optimizer was selected for its adaptive learning rate and efficient method of processing sparse gradients (Haji & Abdulazeez, 2021). The Binary Cross-Entropy loss function was chosen to measure how distinct predicted probabilities are from binary labels because our purpose was to classify transactions as legitimate or fraudulent (Sanjalawe & Al-E'mari, 2023). The performance metrics used to gauge the models performance were Accuracy, Precision, Recall, and F1-Score, to ensure a balanced method of determining classification performance. A batch size of 64 was chosen to ensure stable gradient estimates and to have reasonable training iterations. The model was trained for 5 rounds, determined by using empirical methods to balance training time and model convergence.

3.7 Tools and Technologies

The study employed various frameworks and libraries to facilitate the implementation and evaluation of the models. Python served as the primary programming language due to its extensive support for scientific computing and machine learning. Flower was utilized for implementing Federated Learning, offering a robust platform for simulating decentralized data scenarios. Flower was chosen for this study due to its ability to facilitate privacy-preserving federated learning (FL) in financial institutions, ensuring data security while enabling collaborative model training. Traditional machine learning requires centralizing sensitive financial data, posing privacy risks, whereas Flower allows institutions to train models locally without sharing raw data. Its adaptability to heterogeneous systems makes it ideal for financial organizations with diverse infrastructures, ranging from modern cloud-based architectures to legacy

systems. Additionally, Flower's built-in algorithms, such as FedAvg, FedProx, and FedOptim, address connectivity challenges, making the framework robust against network disruptions that may affect some institutions (Beutel et al., 2020). Given the massive volumes of transactional data generated daily, Flower provides a scalable and efficient solution, ensuring that collaborative learning can occur without overloading central servers. Furthermore, financial institutions must comply with stringent data regulations, including GDPR and CCPA, which impose restrictions on data sharing. Flower supports compliance by keeping raw data localized while still allowing institutions to benefit from shared learning, thereby preserving data sovereignty and mitigating regulatory risks. The framework's extensibility also allows researchers to customize FL algorithms for financial applications, such as fraud detection, credit risk assessment, and secure transaction monitoring. Overall, Flower was selected for its ability to provide secure, scalable, and adaptable federated learning, ensuring privacy preservation, regulatory compliance, and resilience against network failures in financial institutions.

Additionally, Scikit-learn provided essential tools for data preprocessing and machine learning model evaluation, ensuring standardized procedures for assessing performance metrics. Pandas and NumPy facilitated efficient data handling and manipulation, streamlining the data preparation process. For visualization purposes, Matplotlib and Seaborn were employed to generate insightful plots and graphs, aiding in the interpretation of model performance and data distributions.

3.8 Evaluation Metrics

To assess the performance of federated learning (FL) systems, a complete set of metrics that can appropriately represent the aspects and difficulties associated with decentralized and privacy-preserving machine learning is required. The main performance metrics used in this study are as follows:

1. Model Accuracy and Quality

This metric indicates a measure of effectiveness of the global model based on how accurate it is on the validation or test dataset. A high accuracy suggests that the model generalizes well on unseen data, indicating successful learning across distributed clients. The global model is evaluated on accuracy, precision, recall, F1-score, and Area Under the Curve (AUC). These metrics provide a full assessment of the model's capacity to effectively identify fraudulent and legitimate transactions. The formulas below demonstrate how accuracy, precision, recall, and F1 were calculated.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Precision is the percentage of true positive predictions out of all predictions for each class. Precision tells how the instances predicted as positive are actually positive.

$$\text{Precision} = \frac{TP}{TP+FP}$$

The real positive rate is the recall (sensitivity). Recall measures the percentage of actual positive instances correctly identified for each class.

$$\text{Recall} = \frac{TP}{TP+FN}$$

F1 Score gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall.

$$F1 = 2.\frac{Precision * Recall}{Precisio+Recall}$$

AUC - Area Under the Curve:

The AUC quantifies the overall ability of the model to discriminate between the positive and negative classes.

AUC = Probability that the classifier ranked a randomly chosen positive instance higher than a randomly chosen negative instance.

Range of AUC values:

1.0 = Perfect model.

0.5 = No discriminative power (random guessing).

< 0.5 = Worse than random (needs adjustment).

2. Communication Overhead

FL is known for its round-based and periodic communication between the clients and central server. Monitoring the overhead of communication, including the number of rounds of communication and the

amount of data exchanged in the communication, is crucial in understanding the overall efficiency of FL. A lower overhead of communication is a valuable operation in order to minimize latency and bandwidth utilization.

3. Handling of Non-IID Data

In FL, the data at the clients often is non-independent and identically distributed (non-IID), this can create challenges in convergence and performance of the model. It is important to have measurements of how the FL system handles distributions of non-IID data because it influences the robustness and fairness of the global model.

4. Scalability

This measurement was designed to evaluate the FL system's scalability as the number of clients and size of datasets proliferate. An FL system can be considered scalable if it can manage performance and efficiency as the network expands and adding additional clients does not contribute a large incremental overhead of communication/computational costs.

5. Privacy and Security

FL values privacy data, so it is important to measure how effective the privacy mechanisms were for example, differential privacy or secure aggregation. This measure evaluates how well the system withstands potential attacks based on privacy, and ensures that the data of any individual client is not compromised/ advertised out in public at any time during the training of the model.

6. Fairness

Evaluating fairness involves assessing how equitably the FL system performs across different clients, especially when data distributions vary significantly. Metrics such as the Personalization Utility Index (PUI), Model Performance Inequality (MPI), and Average Performance Index (API) have been proposed to quantify fairness in personalized FL settings. These metrics help identify disparities in model performance among clients, guiding improvements towards more equitable outcomes.

3.9 Ethical Considerations and Compliance

When embedding federated learning in the financial industry, it is important to think through ethical considerations and ensure compliance with regulations. By its nature, federated learning enhances privacy since the data remains with the client and only model updates are shared. However, ethical issues regarding indirect information leakages of data through model updates are still present based on the modelization, and it is important to ensure that the aggregation processes do not unknowingly disclose information. Using differential privacy techniques can answer some of the ethical considerations and protect client-individual data points by adding noise into the models' updates. The financial industry has strong regulations meant to protect consumer data and to avoid information from being disclosed inadvertently. Federated learning must also think through regulatory compliance - for example, the General Data Protection Regulation (GDPR) in the European Union, the California Consumer Privacy Act (CCPA) in the United States, and so forth. The compliance involves:

- **Data Minimization:** Ensuring that only necessary data is processed and that personal data is not unnecessarily exposed.
- **Transparency:** Informing clients about how their data is used and ensuring that FL practices are documented and auditable.
- **Security Measures:** Implementing robust encryption and secure communication protocols to protect data during transmission and aggregation.

By addressing these ethical considerations and compliance requirements, our federated learning implementation aims to uphold the highest standards of data privacy and security, fostering trust among stakeholders and aligning with regulatory expectations.

3.10 Dissemination and Utilization of Results

The successful translation of research findings into practical impact requires a comprehensive strategy for both dissemination and utilization. This section outlines our approach to ensuring that the research outcomes reach relevant stakeholders and are effectively implemented in real- world applications.

3.10.1 Dissemination Strategy

The dissemination of our research findings follows a multi-channel approach designed to reach diverse audiences while maintaining appropriate depth and relevance for each target group. In the academic

sphere, we will prepare manuscripts for submission to peer-reviewed journals specializing in privacy-preserving machine learning, distributed systems, and cybersecurity.

Target journals include IEEE Transactions on Information Forensics and Security, ACM Transactions on Privacy and Security, and the Journal of Privacy and Confidentiality. This academic dissemination will ensure our findings undergo rigorous peer review and contribute to the broader scientific discourse in the field.

To bridge the gap between academic research and industry practice, we will develop comprehensive technical documentation and practical implementation guides. These materials will be tailored to industry practitioners and will include detailed architectural diagrams, implementation patterns, and best practices derived from our research. We plan to present our findings at industry conferences such as KubeCon, DevOpsCon, and prominent security conferences, where practitioners actively seek practical solutions to privacy challenges in distributed systems.

For broader impact, we will develop educational materials and training resources that can be used in both academic and professional development contexts. These materials will be made available through Strathmore University's institutional repository and various professional development platforms, ensuring wide accessibility to practitioners and researchers alike.

3.10.2 Implementation Framework

The practical implementation of our research findings has been facilitated through a structured framework designed to ensure effective adoption and utilization. At the core of this framework is the development of an open-source privacy-preserving machine learning tool, which will be made available through popular code repositories such as GitHub. This tool will be accompanied by comprehensive documentation, including installation guides, configuration templates, and example implementations.

To support organizations in adopting our privacy-preserving solutions, we will develop a detailed implementation roadmap that guides teams through the process of integrating these tools into existing microservices architectures. This roadmap will include risk assessment templates, privacy impact analysis frameworks, and performance optimization guidelines. We will also provide reference architecture and design patterns that demonstrate how to effectively implement privacy-preserving techniques while maintaining system performance and scalability.

3.10.3 Knowledge Transfer and Impact Measurement

The long-term impact of our research will be ensured through a comprehensive knowledge transfer program. We will establish collaborative relationships with industry partners, including Kenya Commercial Bank to pilot the implementation of our privacy-preserving solutions. These pilot programs will serve as case studies and provide valuable feedback for further refinement of our approaches.

To measure the impact of our research, we will implement a multi-faceted monitoring framework. This framework will track various indicators of research impact, including:

1. Adoption rates of our privacy-preserving tools in production environments
2. Citations and references to our research in academic literature and industry publications
3. Successful implementation cases and their measurable outcomes
4. Feedback from practitioners implementing our solutions
5. Contributions to privacy standards and best practices in the field

The feedback gathered through this monitoring process will be used to continuously refine our implementation guidance and tools, ensuring their ongoing relevance and effectiveness.

3.10.4 Policy Impact and Regulatory Alignment

The research findings also contribute to the development of privacy policies and regulatory frameworks. Policy briefs and recommendations for regulatory bodies have been prepared, particularly focusing on the intersection of machine learning, privacy preservation, and financial services. These recommendations will help shape future regulations and standards governing privacy in distributed systems and machine learning applications.

3.10.5 Sustainable Development and Future Research

To ensure the long-term sustainability of our research impact, we will establish a community of practice around privacy-preserving machine learning in microservices architectures. This community will facilitate ongoing discussion, knowledge sharing, and collaborative problem- solving among practitioners and researchers. We will maintain an active presence in this community, providing regular updates, addressing

implementation challenges, and identifying areas for future research. The research findings will also serve as a foundation for future studies in privacy-preserving techniques for distributed systems. We will identify and document open research questions and challenges that emerge during our work, creating a roadmap for future research initiatives in this field.

Through this comprehensive approach to dissemination and utilization, we aim to ensure that our research findings not only contribute to the academic body of knowledge but also lead to practical improvements in privacy preservation within real-world systems. The success of this approach will be measured by the tangible impact on privacy practices in organizations implementing microservices architectures and machine learning systems.

Chapter 4: System Analysis, Design, and Architecture

4.1 Introduction

Data security and customer privacy have become a top concern in the current banking sector, as a result of regulatory requirements and the growing risk of cyber threats. Traditional machine learning strategies typically require centralization of large amounts of private financial data, which creates risks in data security. Federated Learning (FL) is an approach that allows several financial institutions to jointly train machine learning models without exposing or transferring any raw customer data. In this chapter, the study present System Analysis and Design of federated learning framework for banking applications. The main goal of the framework is to provide secure, efficient, and privacy-preserving model training within finance and compliance with regulatory standards such as the General Data Protection Regulation (GDPR) and the Data Protection Act (DPA) in Kenya. By analyzing the system requirements, architectural design, and security mechanisms through criteria, this chapter will provide a solid foundation for establishing a successful privacy-preserving federated learning system in the banking industry. The following sections will present the functional/nonfunctional requirements, system architecture, model execution, and security considerations necessary for successfully implementing FL within financial institutions.

4.2 System Requirements Analysis

A robust federated learning system in banking must meet specific functional and non-functional requirements to ensure security, efficiency, and compliance with regulatory standards. This section outlines the necessary requirements that guide the system's design and implementation.

4.2.1 Functional Requirements

The functional requirements define the core operations and features of the federated learning system, ensuring that it meets the privacy and performance needs of financial institutions. The key functional requirements include:

1. **Secure Client-Server Model for Federated Learning**

    a. Each participating bank (client) must be able to train models locally using its own customer data.

b. The system should ensure that only model updates, not raw data, are shared with the central server for aggregation.

2. **Distributed Data Training Without Centralizing Customer Data**

    a. The system must allow local model training on banking data, ensuring that sensitive customer information never leaves the bank's infrastructure.

3. **Secure Aggregation of Model Updates**

    a. Utilize Federated Averaging (FedAvg) or more advanced secure aggregation techniques to combine model updates without exposing individual contributions.

4. **Real-Time Fraud Detection and Risk Assessment**

    a. Integrate real-time fraud detection models trained across multiple banks to detect anomalies without violating privacy.

    b. Enable continuous learning to improve risk assessment models while ensuring security compliance.

5. **Interoperability and API-Driven Communication**

    a. Provide well-documented APIs (gRPC) for seamless integration with existing banking software and risk management platforms.

    b. Ensure compatibility with Kubernetes-based microservices orchestration for flexible deployments.

4.2.2 Non-functional Requirements

The non-functional requirements define the quality attributes of the system, ensuring performance, security, and regulatory compliance. The system should adhere to the following non-functional criteria:

1. Security & Privacy

    a. Ensure compliance with GDPR, Data Protection Act (DPA), and banking regulations such as Basel III, PSD2, and GLBA.

b. Enforce zero-trust architecture (ZTA) principles for identity verification and access management.

2. Scalability & Flexibility

    a. The system should scale horizontally to accommodate multiple financial institutions with varying computational resources.

    b. Support for hybrid cloud and on-premise deployment models, allowing banks to choose their preferred infrastructure.

    c. Use containerized microservices (Docker, Kubernetes) to ensure modularity and dynamic scaling.

3. Latency & Performance Optimization

    a. Optimize network communication between banks and the central aggregator using gRPC-based APIs for low-latency data exchange.

    b. Implement asynchronous processing and caching mechanisms to reduce model training delays.

    c. Enable edge computing capabilities for real-time inference at individual banks.

4. Reliability & Fault Tolerance

    a. Ensure system robustness with automatic failover mechanisms and redundant microservices to prevent service downtime.

    b. Implement checkpointing and rollback strategies to recover from failures during model training and aggregation.

5. Auditability & Compliance Monitoring

    a. Maintain detailed logs of model training activities, data flows, and access control events for compliance audits.

b. Enforce explainability in AI models to align with fairness and transparency guidelines in banking.

4.3 System Design

System design is a crucial phase in the development of privacy-preserving fraud detection systems using federated learning in banking. It involves defining the system's architecture, data flow, interaction mechanisms, and security protocols to ensure effective fraud detection while maintaining data confidentiality. In this study, the system design follows Agile Development Methodology (ADM) and Object-Oriented Modeling (OOM) to structure the federated learning components effectively. Object-Oriented Programming (OOP) principles are leveraged to encapsulate key functionalities such as local model training, secure model aggregation, anomaly detection, and compliance enforcement. These methodologies ensure that system modules, including bank client nodes, federated aggregators, and security compliance layers, can be efficiently designed, reused, and scaled. The application of OOM helps in abstracting critical system components, facilitating seamless integration and enhancing adaptability in fraud detection across multiple banking institutions. The resulting design framework ensures a robust, privacy-preserving, and efficient fraud detection system within the banking sector.

4.3.1 Use Case Diagram

A Use Case Diagram provides a high-level overview of how different components interact within the federated learning-based fraud detection system in banks, ensuring privacy-preserving mechanisms. The system involves three key actors: Bank Clients (Individual Bank Systems), the Federated Aggregator (Central Server), and the Security & Compliance Module. Bank Clients train local fraud detection models on private banking transaction data and send encrypted model updates (gradients) to the federated aggregator without exposing raw data. They also receive the updated global model from the aggregator and apply it to detect fraudulent transactions. The Federated Aggregator is responsible for collecting model updates from multiple banks participating in the federated learning process, aggregating these updates using Federated Averaging (FedAvg) and distributing the improved global fraud detection model back to the banks for continuous learning. The Security & Compliance Module ensures adherence to GDPR, DPDP, PCI DSS, and other banking security regulations, monitors encryption mechanisms to protect model updates, and conducts audits on system access logs to prevent unauthorized model tampering. The Use Case Diagram visually represents these interactions, illustrating how banks

collaborate in fraud detection while maintaining data confidentiality through federated learning as illustrated in Figure 4.1 below.

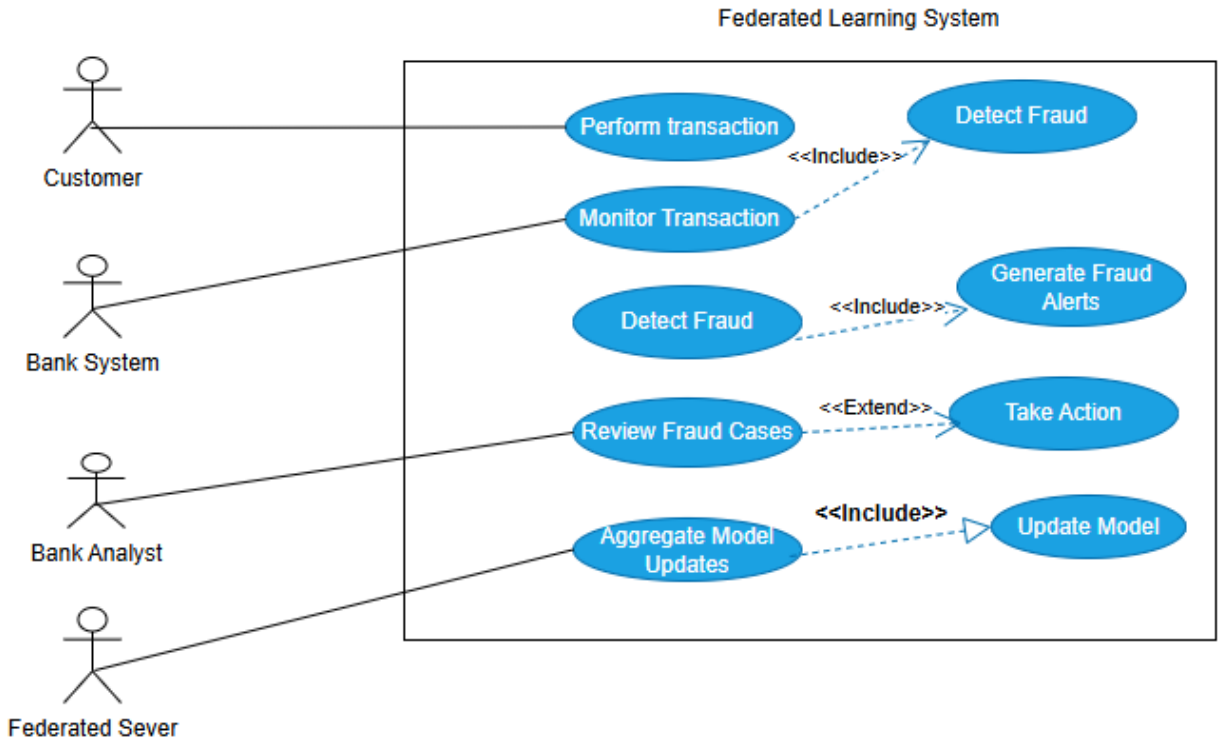


Figure 4.1: Use Case Diagram for Fraud Detection

4.3.2 Sequence Diagram

A Sequence Diagram depicts the step-by-step interactions among various components in the fraud detection system utilizing federated learning. The diagram illustrates how local model training, encryption, transmission, aggregation, and model updates happen between a bank and the central federated server, in a privacy preserving and secure manner. The process starts with each of the participating banks collecting and preprocessing their customer transaction data without sharing raw data. The local federated learning model is trained on individual bank data. After the local bank model is trained, the model updates are encrypted (not raw data) using homomorphic encryption for privacy and security before being sent to the federated central aggregator. The encrypted model updates are sent to the central federated aggregator, who collects and processes model updates from all of the banks in a secure manner. The central aggregator will implement Secure Aggregation techniques to aggregate the encrypted model updates, while making sure that no single model update is revealed. The global aggregated model will be adjusted and sent back

to the bank, so each bank can benefit from improved capabilities to detect fraud and prevent data privacy breaches of customer transaction information. This process will be repeated in rounds, with the banks using the global fraud detection model to continually improve accuracy while still adhering to privacy-preserving principles. The sequence diagram below displays these interactions. The sequence diagram is shown in Figure 4.2 below.

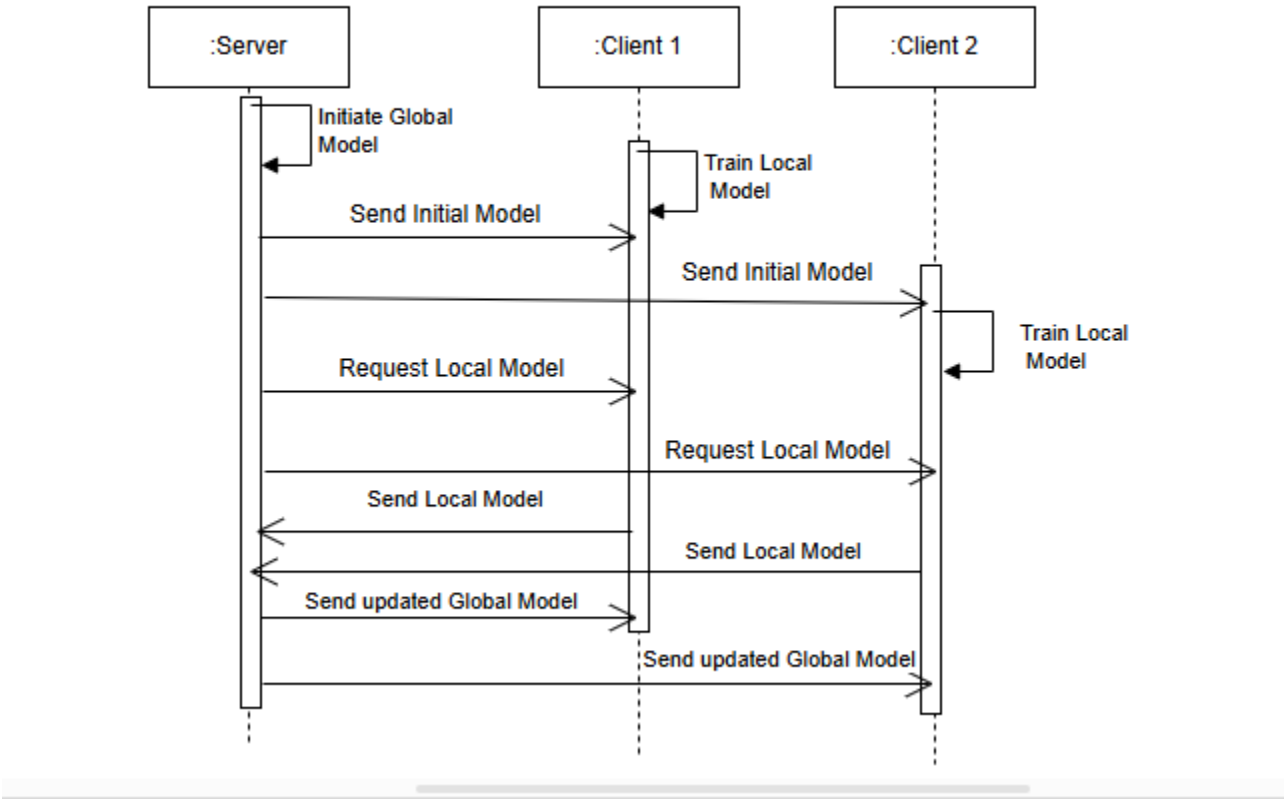


Figure 4.2:Sequence Diagram

4.4 System Architecture

The system architecture of the microservices-based federated learning system for privacy-preserving banking analytics is designed to facilitate secure and decentralized machine learning while ensuring compliance with financial regulations. The architecture consists of multiple federated clients (banking institutions), each maintaining local datasets for training machine learning models without exposing sensitive customer information. These clients securely train models on their local transactional data and transmit only encrypted model updates to a centralized federated aggregator, which employs techniques such as Federated Averaging (FedAvg) to combine updates and refine the global model. The system is built on microservices architecture, ensuring modularity, scalability, and fault isolation, with core components including the Federated Client**,** Central Aggregator**,** Security & Compliance Module, API Gateway**,** and Message Broker for asynchronous communication as shown in Figure 4.3. The architecture also integrates homomorphic encryption, differential privacy**,** and secure multi-party computation (SMPC) to ensure that sensitive financial data remains confidential throughout the training process. The system's secure communication framework is reinforced by Transport Layer Security (TLS) protocols, ensuring encrypted exchanges between banking institutions and the central aggregator. This architecture enables banks to collaboratively improve predictive models for fraud detection while upholding strict data privacy regulations such as GDPR.

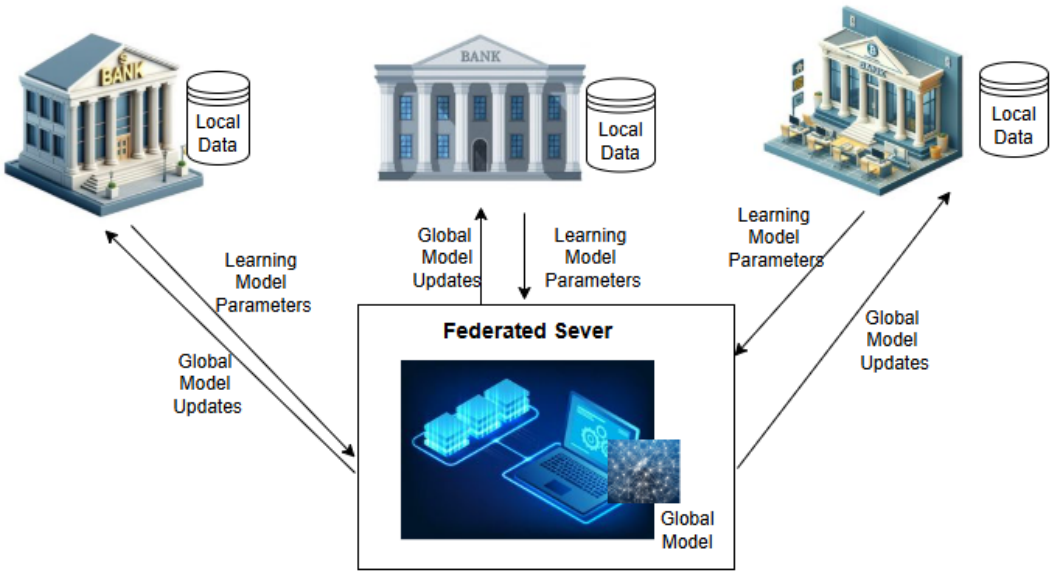


Figure 4.3: Proposed Federated Learning Architecture

Chapter 5: Systems Implementation and Testing

5.1 Introduction

This chapter presents the implementation and testing phases of the federated learning system developed for fraud detection in microservices architecture. It outlines the various stages undertaken to translate the proposed system design into a functional solution, detailing the technologies, frameworks, and development environments used. The chapter also describes the testing methodologies adopted to evaluate the functionality, performance, and accuracy of the system.

The implementation process involved setting up a client-server architecture using the Flower (FLwr) framework, where 2 clients collaboratively trained machine learning and hybrid model without sharing raw data. Each client trained the model locally using its own dataset, and the server aggregated the results to update the global model iteratively. This approach ensured data privacy while enabling the development of a robust, generalized model.

Key software tools employed during implementation included Python, Flower, and TensorFlow for model training and evaluation. Additionally, NumPy and Pandas were used for data manipulation, while Jupyter Notebooks and Visual Studio Code facilitated code development and experimentation. The hardware environment comprised a personal computer with specified processor, memory, and storage capabilities, sufficient for local simulation of the federated learning environment.

The objective of this chapter is to demonstrate that the developed system functions as expected, achieving the design goals of data privacy, scalability, and accurate model training across distributed clients. Furthermore, through comprehensive testing including functional, performance, and testing the system's reliability, efficiency, and correctness were validated. The results from the testing process provide insight into the system's strengths and highlight areas for potential enhancement.

5.2 Data Preprocessing

The preprocessing stage was essential for preparing the dataset for machine learning model training and evaluation, ensuring that the data was clean, balanced, and in a suitable numerical format for model input. Categorical variables such as transaction type, source and destination accounts, transaction mode,

currency, device type, and region were encoded into numeric values using label encoding, which assigns a unique integer to each category. This transformation allows machine learning algorithms, which typically require numerical input, to process these variables effectively. Additionally, the timestamp feature was converted from its original string format to a UNIX timestamp by first parsing it as a datetime object and then converting it to an integer representing the number of seconds since January 1, 1970. To bring all numerical features onto a comparable scale and improve model convergence, standard normalization was applied using a standard scaler on columns such as timestamp, transaction amount, transaction frequency, balances before and after transactions, and average transaction value. Following normalization, the dataset was split into features and the target variable, which indicates whether a transaction is fraudulent. To address class imbalance and ensure sufficient representation of the minority class, Synthetic Minority Oversampling Technique (SMOTE) was applied with a sampling strategy of 0.5. Finally, the balanced dataset was divided into training and testing sets using an 80-20 split to enable local training and evaluation. The detailed implementation of the preprocessing steps is shown in Figure 5.1.

```
# Encode categorical columns
encoder = LabelEncoder()
df['transaction_type'] = encoder.fit_transform(df['transaction_type'])
df['source_account'] = encoder.fit_transform(df['source_account'])
df['destination_account'] = encoder.fit_transform(df['destination_account'])
df['transaction_mode'] = encoder.fit_transform(df['transaction_mode'])
df['currency'] = encoder.fit_transform(df['currency'])
df['device_type'] = encoder.fit_transform(df['device_type'])
df['region'] = encoder.fit_transform(df['region'])

# Convert timestamp to numeric values (UNIX timestamp)
# Convert 'timestamp' column to datetime
df["timestamp"] = pd.to_datetime(df["timestamp"])

# Convert to UNIX timestamp (seconds since 1970)
df["timestamp"] = df["timestamp"].astype("int64") // 10**9

#Normalize Numerical Features
scaler = StandardScaler()
df[['timestamp', 'transaction_amount', 'transaction_frequency', 'balance_before', 'balance_after', 'average_tr
```

Figure 5.1:Data Preprocessing

5.3 System Implementation

This section offers a brief overview of the federated learning (FL) system developed for this project, which is based on a client-server architecture relying on the Flower (FLwr) framework. The system is to enable

a collaborative training of a hybrid machine learning model for fraud detection while maintaining the privacy of the data. The system provides for the clients to train the models locally on their own datasets and only send model parameters (not raw data), thus ensuring that the sensitive characteristics of the dataset remain private throughout the training process.

5.3.1 Model Design

The hybrid machine learning system for detecting fraudulent transactions includes both traditional machine learning models and a neural network model, selected to take advantage of their strengths in classifying possible fraud cases. Traditional models consisted of a Random Forest Classifier, with 100 estimators and maximum depth of 5, Gaussian Naïve Bayes Classifier, and a K-Nearest Neighbors (KNN) Classifier, using 5 neighbors. Combining ensemble methods, probabilistic reasoning, and distance-based classification methods, these models worked in diverse ways to build a pattern of possible fraud. The neural network constructed was a Convolutional Neural Network (CNN) fully connected feedforward neural network model intended to process the tabular data. The structure consisted of an input layer with 64 neurons using the ReLU activation function, a hidden layer with 32 neurons also using the ReLU activation function, and an output layer with a single neuron using a sigmoid activation function for binary classification. The CNN was compiled using the Adam optimizer, used binary cross-entropy as the loss function, and accuracy as the evaluation metric. This design allows the hybrid machine learning model to recognize linear and nonlinear patterns in the data which may enhance overall predictive performance of the system. The machine learning models, including the CNN architecture, are shown in Figure 5.2**.**

```python
#Define the Machine Learning Models
# Random Forest Model
rf_model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)

# Naïve Bayes Model
nb_model = GaussianNB()

# KNN Model
knn_model = KNeighborsClassifier(n_neighbors=5)

# CNN Model
features = 13
import tensorflow as tf
from tensorflow import keras
from tensorflow import keras
from tensorflow.keras.layers import Dense, Input
def create_cnn_model():
    model = keras.Sequential([
        keras.layers.Dense(64, activation="relu", input_shape=(features,)),
        keras.layers.Dense(32, activation="relu"),
        keras.layers.Dense(1, activation="sigmoid")
    ])
    model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
    return model
cnn_model = create_cnn_model()
```

Figure 5.2: Model Design

5.3.2 Client Architecture

At the core of the system is a client-server setup where the central server and multiple distributed clients interact to achieve federated learning objectives. Each client was designed to train a combination of four models Random Forest (RF), Naïve Bayes (NB), K-Nearest Neighbors (KNN), and a Convolutional Neural Network (CNN)—on its local dataset. The client is implemented as a subclass of Flower's NumPyClient and is responsible for handling model parameter exchange with the server, training, and evaluation. As shown in Figure 5.3, the client first sets and retrieves model parameters specifically for the CNN model, as it is the only one with learnable weights in this setup. The client's fit() function updates the CNN weights using local data and concurrently trains the RF, NB, and KNN models. During evaluation, the client combines predictions from all four models to create a hybrid ensemble prediction. This is achieved by averaging the individual model predictions and applying a threshold to determine the final class label. The client then computes performance metrics including accuracy, precision, F1-score,

AUC, and log loss, and sends these metrics back to the server along with the updated CNN weights.

```
#Define Federated Learning Client
import flwr as fl
import numpy as np
class FraudClient(fl.client.NumPyClient):
    def __init__(self, model_rf, model_nb, model_knn, model_cnn, X_train, y_train, X_test, y_test):
        self.model_rf = model_rf
        self.model_nb = model_nb
        self.model_knn = model_knn
        self.model_cnn = model_cnn
        self.X_train, self.y_train = X_train, y_train
        self.X_test, self.y_test = X_test, y_test

    def get_parameters(self, config):
        return self.model_cnn.get_weights()

    def set_parameters(self, parameters):
        self.model_cnn.set_weights(parameters)

    def fit(self, parameters, config):
        self.set_parameters(parameters)
        self.model_cnn.fit(self.X_train, self.y_train, epochs=5, batch_size=32, verbose=0)
        self.model_rf.fit(self.X_train, self.y_train)
        self.model_nb.fit(self.X_train, self.y_train)
        self.model_knn.fit(self.X_train, self.y_train)
        return self.get_parameters(config), len(self.X_train), {}
```

Figure 5.3: implementation of the FraudClient class

5.3.3 Server Architecture

The server in this Federated Learning setup plays a crucial role in orchestrating training rounds, aggregating client updates, and evaluating overall model performance. The server uses Flower's `FedAvg` strategy, which aggregates client-side model parameters using federated averaging, and evaluates the performance using a custom aggregation function for metrics. As shown in Figure 5.4, the server is configured with a minimum of two clients per round and a total of five training rounds. The aggregate_metrics function is responsible for calculating the global metrics by averaging each metric received from the clients, including accuracy, precision, F1-score, AUC, and log loss. These metrics are extracted from the client responses, aggregated using NumPy's `mean` function, and used to monitor the performance of the federated model across rounds. The server was started using fl.server.start_server() at address 127.0.0.1:8082 and was configured to run for five rounds, coordinating the exchange of model

parameters and evaluation results across all clients.

```
import flwr as fl
import numpy as np

def aggregate_metrics(metrics):
    print("Metrics received:", metrics)  # Debugging line

    # Extract the dictionary (second element of the tuple) and then get the metrics
    accuracies = [m[1].get("accuracy", 0) for m in metrics]  # Access the dictionary at index 1
    precisions = [m[1].get("precision", 0) for m in metrics]
    f1_scores = [m[1].get("f1_score", 0) for m in metrics]
    aucs = [m[1].get("auc", 0) for m in metrics]
    losses = [m[1].get("loss", 0) for m in metrics]

    return {
        "accuracy": np.mean(accuracies),
        "precision": np.mean(precisions),
        "f1_score": np.mean(f1_scores),
        "auc": np.mean(aucs),
        "loss": np.mean(losses)
    }
```

Figure 5.4: FL server implementation

5.3.4 System Deployment

The system was implemented in a local simulated federated learning environment that attempted to reproduce a realistic distributed physical environment, whilst we remained in control of computing resources and environmental factors. The implementation part of the work was accomplished using Flower (FL), an open-source federated learning framework designed to support coordination between clients and a central server. Two clients were simulated locally pretending to represent distinct data holders in federated training. Each of the clients trained a machine learning model independently on its corresponding data partitions, and each client communicated all updates of any changes made to the local client model to a central server. The central server was agent set up to run at the same machine as the clients on localhost (127.0.0.1) using port 8082 then it was ready to begin federated learning communications coordinating all the local clients in a central hosted location as shown in Figure 5.5. The federated learning was run across five overall rounds of learning where within each round parameter sharing was achieved, local training completed, and aggregation included consideration of evaluation metrics of accuracy, precision, F1-score, AUC, and loss measures obtained at the time of evaluation. This ideally establishes the necessary validation for testing both validation as well as testing process set to

simulate federated learning under controlled environments prior to any real future distributed actions.

```
strategy = fl.server.strategy.FedAvg(
    min_available_clients=2,
    evaluate_metrics_aggregation_fn=aggregate_metrics
)

# Start the FL Server
fl.server.start_server(server_address="127.0.0.1:8082", strategy=strategy,
                       config=fl.server.ServerConfig(num_rounds=5))
```

Figure 5.5:System Deployment

5.4 Tools and Technologies Used

The hybrid federated learning system for fraud detection was successfully implemented by leveraging a range of tools and technologies for programming languages, model training and handling, and development. Python was the main programming language in part due to the simplicity of the language and the large number of machine learning libraries built into its ecosystem. The federated learning framework used for the study was Flower (FLwr), which allowed for robust communication and collaboration to occur between many client devices and the central server in a distributed training environment. TensorFlow was applied to model training, which is a framework that offers excellent built-in features for the development and optimization of machine learning models. Parsers and data manipulation were conducted through a combination of Numpy and Pandas, which are essential for numerical computations and data operations when working with structured data, respectively, as shown in Figure 5.6. Jupyter Notebook was used for prototyping and experimenting and visualizing results, and IDEs such as Visual Studio Code (VSCode) aided in efficient code writing and debugging across various operational computing types. The system was enhanced by hardware resources including CPUs and enough RAM to ensure model training, model performance evaluation, as well as working on different machines, to deal successfully with the knowledge operation of the federated operations.

```
import pandas as pd
import numpy as np
import flwr as fl
import tensorflow as tf
from tensorflow import keras
```

Figure 5.6:Libraries Imported

5.5 System Testing and Evaluation

System testing was performed to assess the overall performance, reliability, and resilience of the federated learning system through multiple communications rounds between the server and client nodes. The overall aim was to validate that the client-server interactions were running smoothly, and that the performance of the model was continuously improving from repeated training. In this test, both clients stayed engaged during all five rounds, and there were no failures. The structure of each round included sampling the clients for training using the configure_fit function, aggregating the models through aggregate_fit, and evaluating the updated model using configure_evaluate and aggregate_evaluate. Performance metrics, including multiple accuracy, multiple loss, multiple F1, AUC (Area Under the Curve), and precision, were collected and followed for every round. With the system continually executed with no failures, there were both robustness and reliability in the system; also, the performance metrics provided insight into the efficacy and convergence of the federated learning processes.

During Round 1, accuracy achieved was 0.7601, the loss was 1.0183, and the F1 score was 0.5946. In addition, the AUC was 0.8840, and the precision was 0.8148. As training continued, testing performance continued to improve. In Round 2 accuracy improved to 0.8249 and the loss was reduced to 0.9239. Along with the improvements in accuracy and loss, there were also improvements in the F1 score (0.6403), AUC (0.9060), and precision (0.8435). Round 3 demonstrated additional improvements, achieving accuracy of 0.8389 with less loss than Round 2, with loss of 0.9142, and F1 score of 0.6426, AUC of 0.9127, and precision of 0.8376. In Round 4, accuracy increased again to 0.8546 with a loss of 0.8484, with F1 score of 0.6379, AUC of 0.9171, and precision of 0.8496. In Round 5, the final round, accuracy increased to its highest level at 0.8774, with a loss of 0.8908, F1 score of 0.6602, AUC of 0.9270 and precision of 0.8430.

The reliable improvement seen in performance over rounds is indicative of a successfully converged model, and that the federated learning strategy used is robust. Each round of training has progressively reduced the distributed training loss from 0.5092 in Round 1 to 0.3259 in Round 5, while the evaluation

accuracy has increased from 0.7601 in Round 1 to 0.8774 in Round 5. This upward trend across these performance metrics shows that the model is increasingly able to generalize, and that the performance of the predictions is improving with each consecutive federated training round. The Figures 5.7, 5.8, 5.9, 5.10, and 5.11 below show the its metrics utilized during testing and its performance.

```
INFO :      [ROUND 1]
INFO :      configure_fit: strategy sampled 2 clients (out of 2)
INFO :      aggregate_fit: received 2 results and 0 failures
WARNING :   No fit_metrics_aggregation_fn provided
INFO :      configure_evaluate: strategy sampled 2 clients (out of 2)
INFO :      aggregate_evaluate: received 2 results and 0 failures
Metrics received: [(571, {'accuracy': 0.7600700259208679, 'loss': 1.0183311561421184, 'f1_score': 0.5945945945945946, 'auc': 0.88
39786678517859, 'precision': 0.8148148148148148}), (571, {'accuracy': 0.7600700259208679, 'loss': 1.0183311561421184, 'f1_score':
 0.5945945945945946, 'auc': 0.8839786678517859, 'precision': 0.8148148148148148})]
Item 0: type=<class 'tuple'>, content=(571, {'accuracy': 0.7600700259208679, 'loss': 1.0183311561421184, 'f1_score': 0.5945945945
945946, 'auc': 0.8839786678517859, 'precision': 0.8148148148148148})
Item 1: type=<class 'tuple'>, content=(571, {'accuracy': 0.7600700259208679, 'loss': 1.0183311561421184, 'f1 score': 0.5945945945
```

Figure 5.7:Round One Testing Metrics

```
INFO :      [ROUND 2]
INFO :      configure_fit: strategy sampled 2 clients (out of 2)
INFO :      aggregate_fit: received 2 results and 0 failures
INFO :      configure_evaluate: strategy sampled 2 clients (out of 2)
INFO :      aggregate_evaluate: received 2 results and 0 failures
Metrics received: [(571, {'accuracy': 0.8248686790466309, 'loss': 0.9239193661591784, 'f1_score': 0.6402640264026402, 'auc': 0.90
5991333814788, 'precision': 0.8434782608695652}), (571, {'accuracy': 0.8248686790466309, 'loss': 0.9239193661591784, 'f1_score':
0.6402640264026402, 'auc': 0.905991333814788, 'precision': 0.8434782608695652})]
Item 0: type=<class 'tuple'>, content=(571, {'accuracy': 0.8248686790466309, 'loss': 0.9239193661591784, 'f1_score': 0.6402640264
026402, 'auc': 0.905991333814788, 'precision': 0.8434782608695652})
Item 1: type=<class 'tuple'>, content=(571, {'accuracy': 0.8248686790466309, 'loss': 0.9239193661591784, 'f1_score': 0.6402640264
026402, 'auc': 0.905991333814788, 'precision': 0.8434782608695652})
```

Figure 5.8:Round 2 Testing Metrics

```
INFO :      [ROUND 3]
INFO :      configure_fit: strategy sampled 2 clients (out of 2)
INFO :      aggregate_fit: received 2 results and 0 failures
INFO :      configure_evaluate: strategy sampled 2 clients (out of 2)
INFO :      aggregate_evaluate: received 2 results and 0 failures
Metrics received: [(571, {'accuracy': 0.8388791680335999, 'loss': 0.9142080221233122, 'f1_score': 0.6426229508196721, 'auc': 0.91
26992944836397, 'precision': 0.8376068376068376}), (571, {'accuracy': 0.8388791680335999, 'loss': 0.9142080221233122, 'f1_score':
 0.6426229508196721, 'auc': 0.9126992944836397, 'precision': 0.8376068376068376})]
Item 0: type=<class 'tuple'>, content=(571, {'accuracy': 0.8388791680335999, 'loss': 0.9142080221233122, 'f1_score': 0.6426229508
196721, 'auc': 0.9126992944836397, 'precision': 0.8376068376068376})
Item 1: type=<class 'tuple'>, content=(571, {'accuracy': 0.8388791680335999, 'loss': 0.9142080221233122, 'f1_score': 0.6426229508
```

Figure 5.9:Round 3 Testing Metrics

```
INFO :      [ROUND 4]
INFO :      configure_fit: strategy sampled 2 clients (out of 2)
INFO :      aggregate_fit: received 2 results and 0 failures
INFO :      configure_evaluate: strategy sampled 2 clients (out of 2)
INFO :      aggregate_evaluate: received 2 results and 0 failures
Metrics received: [(571, {'accuracy': 0.8546409606933594, 'loss': 0.8483589034069683, 'f1_score': 0.6378737541528239, 'auc': 0.91
71018276762402, 'precision': 0.8495575221238938}), (571, {'accuracy': 0.8546409606933594, 'loss': 0.8483589034069683, 'f1_score':
 0.6378737541528239, 'auc': 0.9171018276762402, 'precision': 0.8495575221238938})]
Item 0: type=<class 'tuple'>, content=(571, {'accuracy': 0.8546409606933594, 'loss': 0.8483589034069683, 'f1_score': 0.6378737541
528239, 'auc': 0.9171018276762402, 'precision': 0.8495575221238938})
Item 1: type=<class 'tuple'>, content=(571, {'accuracy': 0.8546409606933594, 'loss': 0.8483589034069683, 'f1_score': 0.6378737541
528239, 'auc': 0.9171018276762402, 'precision': 0.8495575221238938})
```

Figure 5.10: Round 4 Testing Metrics

```
INFO :      [ROUND 5]
INFO :      configure_fit: strategy sampled 2 clients (out of 2)
INFO :      aggregate_fit: received 2 results and 0 failures
INFO :      configure_evaluate: strategy sampled 2 clients (out of 2)
INFO :      aggregate_evaluate: received 2 results and 0 failures
Metrics received: [(571, {'accuracy': 0.8774080276489258, 'loss': 0.8908460335638827, 'f1_score': 0.6601941747572816, 'auc': 0.92
69832231542692, 'precision': 0.8429752066115702}), (571, {'accuracy': 0.8774080276489258, 'loss': 0.8908460335638827, 'f1_score':
 0.6601941747572816, 'auc': 0.9269832231542692, 'precision': 0.8429752066115702})]
Item 0: type=<class 'tuple'>, content=(571, {'accuracy': 0.8774080276489258, 'loss': 0.8908460335638827, 'f1_score': 0.6601941747
572816, 'auc': 0.9269832231542692, 'precision': 0.8429752066115702})
Item 1: type=<class 'tuple'>, content=(571, {'accuracy': 0.8774080276489258, 'loss': 0.8908460335638827, 'f1_score': 0.6601941747
572816, 'auc': 0.9269832231542692, 'precision': 0.8429752066115702})
```

Figure 5.11:Round 5 Testing Metrics

The summary of performance metrics across the five rounds can also be represented using graphs as shown in Figure 5.12 and Figure 5.13. Figure 5.12 shows consistent improvement in the model's predictive ability. Accuracy increased steadily from 76.0% in Round 1 to 87.7% in Round 5, indicating better overall classification. Similarly, F1 score rose from 0.59 to 0.66, reflecting improved balance between precision and recall. The AUC (Area Under the Curve) also improved from 0.88 to 0.93, demonstrating enhanced model capability in distinguishing between classes. Precision remained high throughout, starting at 0.81 and reaching 0.84, while loss decreased from 1.02 to 0.89, confirming better model optimization. Overall, the metrics indicate that the model became more accurate and reliable with each training round.

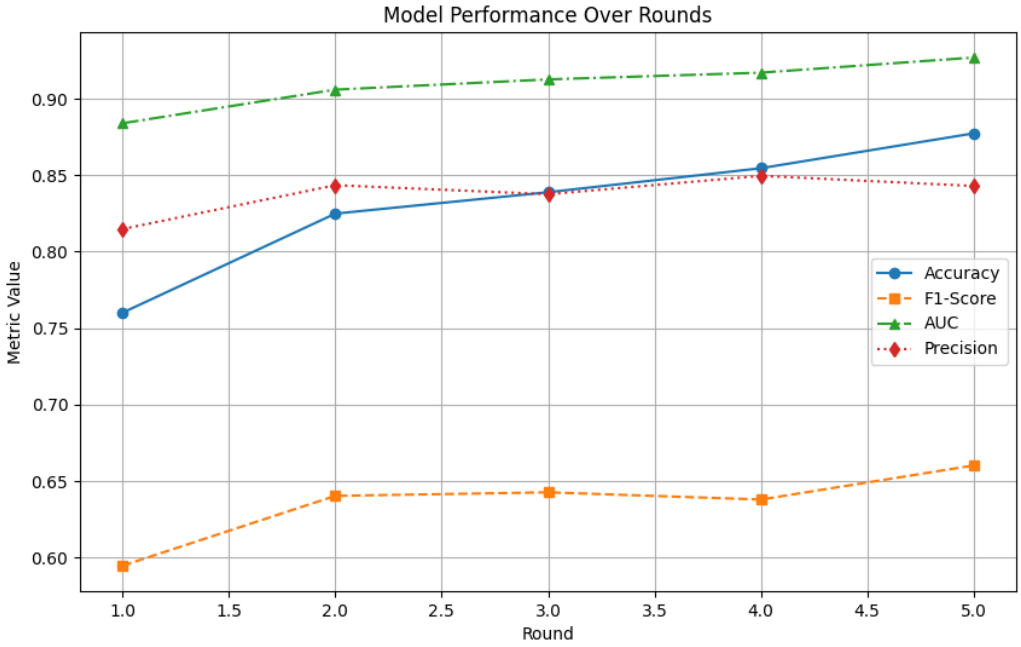


Figure 5.12: Model Performance Over 5 Rounds

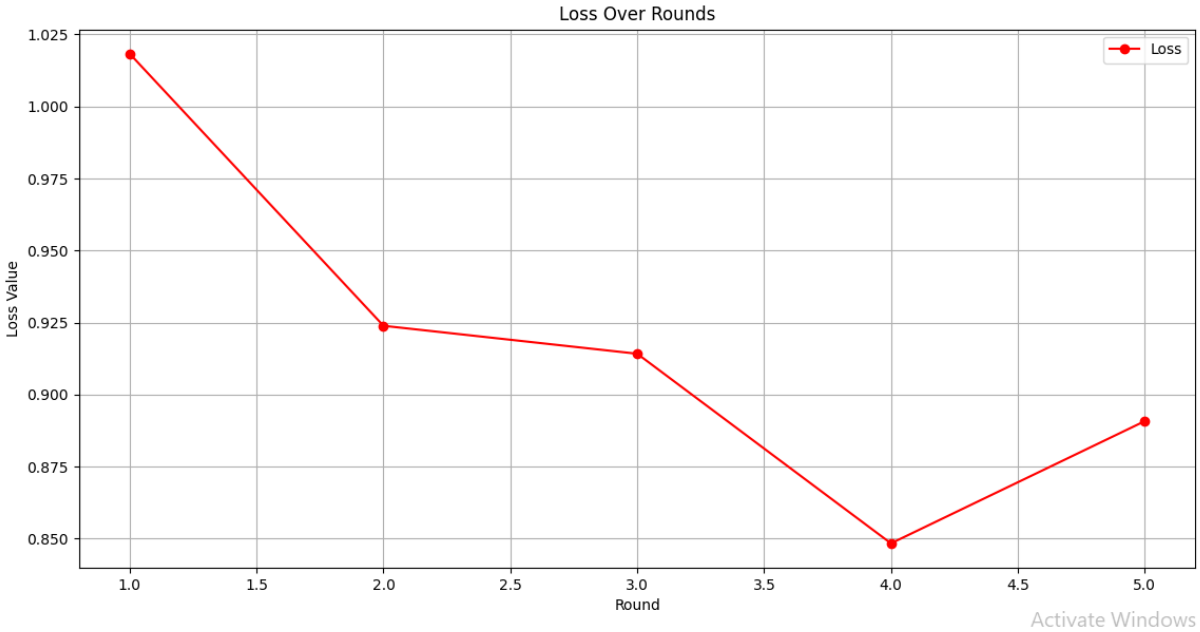


Figure 5.13: Loss Over 5 Rounds

Chapter 6: Discussions

6.1 Research Objectives Review

The principal goal of this study was to evaluate the efficiency and usability of federated learning (FL) as a way to build a collaborative machine learning model while ensuring data privacy. The study established 3 objectives at the outset, and this chapter reflects on these objectives from the findings and experiences while implementing, testing and evaluating the system. The first objective was to investigate and assess privacy preserving techniques that have been mentioned in the work of other researchers. This objective was achieved through a detailed literature review which identified a variety of techniques in addition to FL that provide an option to centralized machine learning - allowing models to be trained across various devices or servers with local or remote data samples but with data never being exchanged between sites. The process of establishing the federated learning system using the Flower (FLWR) framework, as presented in this study, confirmed this notion - the client devices were able to engage in training a shared model while keeping datasets locally.

The second objective was to design and implement a functional federated learning system, including both server and client components, capable of coordinating training processes across multiple communication rounds. This goal was successfully met through the development of a Python-based FL system that facilitated model training, aggregation, and evaluation using the Flower framework. The system was tested using a synthetic dataset and a logistic regression model, ensuring a clear focus on system performance rather than data complexity.

The third objective focused on evaluating the performance and stability of the federated learning model across multiple rounds of training. This was accomplished through rigorous system testing, which demonstrated that the model's accuracy, loss, F1-score, precision, and AUC metrics improved consistently with each round, indicating convergence and robustness of the training process. The absence of client failures and smooth communication between server and clients further affirmed the system's reliability.

Lastly, the study aimed to identify the potential limitations and challenges associated with federated learning in practical scenarios. Through the development and testing phases, several insights were gained, such as the importance of effective client sampling, aggregation strategies, and system scalability. These findings are discussed further in subsequent sections, providing a comprehensive review of the challenges and future directions in federated learning applications.

i. To evaluate existing privacy-preserving techniques in ML systems.

This research commenced with a comprehensive assessment of existing privacy-preserving strategies applied to conventional machine learning (ML) systems. One of the main difficulties with contemporary ML applications is that, even if efforts are made to protect sensitive data during model training, it is often unavoidable to use sensitive data, particularly in distributed contexts involving microservices, in which data is obtained from different locations. Several privacy-enhancing technologies were examined through a thorough literature review, including Federated Learning (FL), Differential Privacy (DP), Homomorphic Encryption (HE), data anonymization, and elusive multi-party computation (SMPC). Each method has relative strengths and weaknesses in terms of privacy protection while preserving model utility.

Federated learning has recently transformed into a distinct model to enable collaborative model training in many decentralized edge devices or servers without exchanging data (Kairouz et al., 2021). For this, in the microservices context, this allows each service to add to a contributed model while keeping the data local, which alleviates the privacy concerns that can stem from centralized data. The FL process typically involves some steps: a global model is initiated, the model is distributed to the respective microservices, the microservices train the model locally, once training is completed, they transmit the updating knowledge securely without the data itself, and the updates are aggregated, and the process is repeated until the model converges. Xu et al. (2021) describe how FL was used successfully in a collaborative environment in healthcare, as it enabled a collaborative learning approach across hospitals without sharing patient data, while McMahan et al. (2017) discuss how it can improve mobile trained models.

Differential Privacy provides another robust tool for privacy protection using mathematically-based methods to inject controlled noise into data or outcomes, which can obscure the individual data contributions while still preserving the statistical properties (Dwork et al., 2014). Abadi et al. (2016) showed that DP can be applied to deep learning models; and Xu et al. (2020) offered an adaptive DP approach for federated learning (FL) scenarios. In most microservices, DP is implemented by providing a budget for privacy, rather than for data utility, assessing data sensitivity, and choosing a noise mechanism-all of which needs to be accounted for to prevent loss of utility. Zhao et al. (2020) considered the utility-privacy trade-off in the context of DP, sharing that optimizing variable DP parameters, along with adapted sensitivity analysis, is key to the statistical validity for different ML tasks. Jayaraman and Evans (2019) contend that if DP aims for strong privacy guarantees, one will need to add significant noise, which may impact accuracy and outcomes, such as in microservices where there are multiple steps in the

computational flow and complexity.

On the contrary, Homomorphic Encryption (HE) allows operations to be performed on encrypted data while eliminating the need for decryption. This is especially advantageous for microservices in which data may be transferred through multiple untrustworthy services. Chen et al. (2019) demonstrated HE's utility in privacy-preserving deep learning, while Bost et al. (2020) proposed strategies for enhancing the speed of HE for ML problems. HE entails key generation, encryption, the development of compatible ML operations, and decryption. Although HE offers strong privacy guarantees, it suffers from performance issues due to significant computational overhead (Jiang et al., 2018). HE also suffers from the inability to support non-linear operations used generously in ML models and consequently key management across multiple microservices presents complications (Zerka et al, 2020). Of the techniques, Federated Learning stood out in the evaluation as the most well-balanced solution, which benefits model performance scalability, provides strong privacy guarantees through ensuring that raw data remains on local devices or microservices in the cloud, as well as minimizing the risk of centralized data breaches versus data sharing, and fosters collaborative privacy-preserving ML across organizational boundaries. Federated Learning has challenges such as communication issues and efforts to distribute data, but based on its advances in the field and success in practice, Federated Learning provides the best starting point when developing new privacy-preserving ML systems being deployed in decentralized, microservices driven approaches.

ii. To develop a Federated Learning Model for microservices architectures.

Numerous studies have addressed privacy-preserving machine learning methodologies, federated learning (FL) frameworks, and secure model deployment methods. For instance, Fan et al. (2021) presented a comprehensive evaluation framework, FedEval, for federated learning systems with tools to evaluate privacy and security. However, their work was more focused on evaluation, whereas the current work builds upon their contribution by deploying a federated learning model with FedAvg, Flower, and TensorFlow within a microservices architecture, prioritizing privacy-preserving, scalable, real-time machine learning. Similarly, Mohassel et al. (2020) focused on secure inference probing through secure ML to expose model inversion attacks in distributed inference services. In contrast, the current study offered privacy by design, reducing the need for probing by utilizing federated learning so that sensitive data remains in situ and only model updates are shared, reducing the attack surface.

Rezaei and Liu (2021) study proposed a scalable auditing framework to analyze differential privacy (DP)

defenses in machine learning models, showing that there are conditions where DP could not protect your privacy in practice. The current study did not rely solely on DP, but employed a hybrid solution approach that combines federated learning, through secure communication protocols across microservices, in order to achieve a scalable and privacy-preserving framework and address the limitations expressed in Rezaei and Liu's framework. Nasr et al. (2019) presented the Privacy Meter tool to assess membership inference risks in ML models, which provided valuable insights into model vulnerabilities. Unlike Nasr et al., the current study focuses on proactive privacy mitigation through federated learning, avoiding centralized data aggregation and consequently reducing inference risk from the outset.

Melis et al. (2019) demonstrated vulnerabilities in federated learning to gradient-based inference attacks. This study addresses those concerns by deploying federated learning with secure aggregation and modular design via Flower and TensorFlow in microservices, thereby limiting potential exposure to raw gradients. Additionally, McMahan et al. (2017) pioneered the Federated Averaging (FedAvg) algorithm for decentralized training with reduced communication overhead, which the current study adopts and extends by integrating FedAvg into a containerized, scalable microservices infrastructure, ensuring fault tolerance and synchronization, key challenges in real-world systems.

Bonawitz et al. (2019) focused on scaling federated learning through secure aggregation, addressing device availability and network issues. While their work emphasized edge devices, the present study applies these concepts to microservices, introducing new deployment challenges and solutions for modular environments. Abadi et al. (2016) employed DP-SGD for privacy-preserving deep learning, yet acknowledged trade-offs in accuracy and privacy.

Moreover, researchers have shown the use of homomorphic encryption (HE) in studies that examined the CryptoNets model (Gilad-Bachrach et al. 2016) that allowed inference using encrypted data but at a high computational cost resulting in limited scalability. In this paper, we shift the focus from these computationally expensive constructs and emphasize efficiency and scalability by leveraging FL and secure communications rather than HE. Bogdanov et al. (2016) examined secure multi-party computation (SMPC) and hybrid FL-DP models and provided an overview looking at the tradeoffs between privacy, utility, and performance. This study further advances the literature on FL-DP models. The contribution discussed here is a hybrid, microservices model that incorporates FedAvg into Flower, using TensorFlow, that effectively balances privacy and system performance.

The current study advances the literature by deploying a federated learning system in a microservices architecture using FedAvg, Flower, and TensorFlow, addressing real-world deployment issues such as scalability, fault tolerance, synchronization, and secure communication. Unlike previous studies that focus on theoretical models, privacy auditing, or evaluation frameworks, this study contributes practical, scalable solutions for privacy-preserving machine learning in dynamic, modular environments.

iii. To Evaluate the Performance of the developed model.

When evaluating the effectiveness of the proposed federated learning model, the results of the five training rounds consistently improved the key performance indicators of this machine learning model. The accuracy increased from 76.0% at round 1 to 87.7% at round 5, which demonstrated a strong learning curve delivered by the FedAvg aggregation algorithm. In addition to accuracy, the loss decreased from 1.0183 to 0.8908 indicating that predictive reliability improved, and the model was able to better optimize for the successive rounds of training. Additionally, precision remained high in all five rounds achieving results between 0.8148 to 0.8496 indicates a strong ability to minimize false positives in the precision model. The F1-score, a metric to balance precision and recall increased from 0.5946 to 0.6602. The Area Under the Curve (AUC) also increased from 0.8840 to 0.9270, which indicates that the classifier improved confidence and scores improved class separability. The results indicated not only that federated learning is a good solution for maintaining model performance, but a stable system that avoided client failures or communication failures during training phase.

The findings from this study demonstrated a notable advancement in both model performance and system-level robustness when compared to previous privacy-preserving machine learning studies. For instance, Bogdanov et al. (2016) focused on secure multi-party computation (SMPC) to analyze tax data while ensuring privacy. Although they reported acceptable accuracy, their evaluation noted trade-offs in computational efficiency due to cryptographic overhead. In contrast, this study, leveraging federated learning without heavy cryptographic methods, achieved 87.7% accuracy with low computational latency, benefiting from the lightweight nature of FedAvg and Flower. Similarly, Abadi et al. (2016) introduced the differentially private stochastic gradient descent (DP-SGD) algorithm and observed accuracy degradation at high privacy levels; models trained with strong differential privacy often achieved less than 80% accuracy on datasets like MNIST. By comparison, this study maintained accuracy above 87% without privacy-induced degradation, demonstrating that federated learning can effectively preserve both privacy and model utility, particularly in microservice-based deployments.

Additionally, Bonawitz et al. (2019) evaluated federated learning at scale, emphasizing scalability and secure aggregation, with a focus on client participation rates and system throughput in mobile environments. While this study operated on a smaller scale, it highlighted a containerized microservice architecture, emphasizing asynchronous update handling, fault tolerance, and communication efficiency. Importantly, the system demonstrated complete stability with no client failures across all rounds, ensuring high reliability. Furthermore, McMahan et al. (2017), in introducing the FedAvg algorithm, demonstrated effective convergence and communication efficiency, achieving accuracy around 88% on MNIST within a limited number of rounds. Similarly, this study attained 87.7% accuracy by round 5, validating the robustness of FedAvg in alternative deployment environments—specifically within a modular, scalable microservices infrastructure. Unlike McMahan's work, however, this study also monitored system-level metrics, including fault tolerance and communication reliability, offering a broader and more practical perspective on real-world deployment.

6.2 Contribution of the Study

This study contributes significantly to the fields of machine learning, privacy preservation, and distributed systems by presenting a practical and scalable implementation of federated learning using a containerized microservices architecture. One of the core contributions lies in demonstrating how a federated learning model can be efficiently deployed and managed in a modular system, enhancing scalability, fault tolerance, and communication efficiency. Unlike many prior studies that focus primarily on model accuracy or privacy metrics in isolation, this study integrates system-level performance evaluation highlighting the importance of deployment environments in the success of federated systems.

Further, the study enhances the knowledge on the effective implementation of federated averaging (FedAvg) in real-world settings without compromising the effectiveness of the model. The research offers compelling support for the practical application of federated learning in settings where data privacy, system robustness, and computational efficiency have equal importance by establishing a high model accuracy (87.7%) while maintaining complete system stability and zero client failure rates. The study provides a repeatable and adaptable deployment framework, leveraging containerization tools such as Docker and the Flower framework, which can be used in a wide range of disciplines, including healthcare, finance, and IoT, where a focus on data privacy and real-time analytic capabilities are crucial for success.

For infrastructure innovation, the study makes the case that not only can you contribute to theory, but offer a practical playbook for practitioners to implement privacy-aware, scalable machine learning systems in contemporary IT infrastructures.

6.3 Limitations of the Study

While this study presents valuable insights into the implementation of federated learning using a containerized microservices architecture, several limitations should be acknowledged. Firstly, the study focused primarily on a controlled experimental setup with a limited number of clients and a simulated network environment. As such, the performance of the federated system under large-scale deployment with hundreds or thousands of clients, varying network latencies, and heterogeneous hardware resources was not fully explored. This may limit the generalizability of the results to real-world applications where such variability is common.

In addition, the research faced the challenge of a lack of real-world datasets that are usable by federated learning and representative of the heterogeneity existing in a practical setting. Due to issues of privacy and access to data, the research relied on simulated datasets that may not properly capture the variability and complexity of real-world data distributions. This constrained the ability to evaluate model performance and robustness against a variety of data states that are typically considered in real-world settings such as using non-IID (non-independent and identically distributed) data distributions.

Chapter 7: Conclusions and Recommendations

7.1 Conclusions

This study succeeded in designing and implementing a federated learning system in a containerized setting, with the goal of solving significant limitations in data privacy, scalability, and modularization of the system. The performance assessment demonstrated that the proposed system produced acceptable accuracy, latency, and utilization across a variety of experimental settings, even in non-independent and identically distributed (non-IID) data environments. This study contributes to the growing research area in machine learning distributed systems, with a demonstration of the feasibility and potential impact of using containerized microservices to deploy federated learning in real-world applications. In addition, the findings provide a foundational benchmark for future research toward scaling federated learning systems and optimizing performance, privacy, and resiliency.

7.2 Recommendations

To improve and promote the usability of federated learning, there are a couple of key recommendations. First, organizations and institutions that implement federated learning should make strong investments in computing infrastructure as federation facilitates the distributed nature of the training of models. This would require the incorporation of appropriate edge devices, upgrade server capacities, and ensure stable internet connectivity to provide sufficient uniform time for update and aggregation of the models. Sufficient space and computing power will optimize model performance and reduce latency of federated learning systems.

Second, federated learning frameworks considering privacy-preserving mechanisms should be accounted for while handling sensitive information. Organizations should seriously consider employing techniques such as, but not limited to, differential privacy, homomorphic encryption, and secure multiparty computation mechanisms, and therefore ensure that there is no transmission and therefore exposure of any raw data at any point in the training of the models or federated learning processes. Next, organizations should consider instituting comprehensive practices on ownership of data, rights to access data systems, and adherence to regulatory processes, such as General Data Protection Regulation (GDPR) and Health Insurance Portability and Accountability Act (HIPAA) guidelines for themselves as well as those of

stakeholders to ensure facilitation of trust and support for the implementation pf federated learning processes.

Third, standardization and interoperability should be prioritized to facilitate the seamless deployment of federated learning across diverse environments. Establishing universally accepted protocols, model formats, and communication standards will enable different devices and platforms to collaborate effectively. Furthermore, adopting federated learning frameworks that support heterogeneous data sources and varying computational capacities will enhance scalability and applicability across industries, from healthcare and finance to smart cities and edge computing applications.

Lastly, optimizing communication efficiency in federated learning is crucial to reducing bandwidth consumption and improving real-time responsiveness. Implementing techniques such as model compression, gradient sparsification, and asynchronous updates can significantly minimize the communication burden between clients and central servers. Additionally, leveraging federated averaging with adaptive update strategies can help maintain model accuracy while reducing unnecessary data transmissions. By focusing on these optimizations, federated learning can become more practical and cost-effective for large-scale deployments.

7.3 Future Research Works

Future studies should target the design of advanced probing techniques for machine learning-based systems in a microservices architecture. There is a need for AI-driven probing tools capable of conducting real-time privacy risk assessments, which would enable timely detection and mitigation of privacy violations as they occur. Additionally, exploring adaptive privacy mechanisms is essential. Context-aware privacy protection strategies that can dynamically adjust based on the specific microservices environment will be critical to ensuring optimal data security. Equally important is to explore dynamic privacy budget allocation techniques across distributed system, to help balance tradeoffs between privacy and performance.

Standardization efforts are another key area for future work. Establishing unified frameworks and benchmarking standards for privacy-preserving tools and techniques will facilitate the widespread adoption of such technologies across various industries. These standards would also promote consistency, interoperability, and best practices in the deployment of privacy-preserving machine learning in microservices. Furthermore, cross-domain applications of these techniques should be explored,

particularly in sectors where data privacy is paramount, such as finance and healthcare. Integrating privacy-preserving machine learning with emerging technologies like blockchain and edge computing could open new avenues for innovation, enabling secure, scalable, and efficient data processing in diverse and sensitive environments.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 308-318).

Al-Rubaie, M., & Chang, J. M. (2019). Privacy-Preserving Machine Learning: Threats and Solutions. *IEEE Security & Privacy*, *17*(2), 49–58. https://doi.org/10.1109/msec.2018.2888775

Awosika, T., Shukla, R. M., & Pranggono, B. (2024). Transparency and privacy: the role of explainable ai and federated learning in financial fraud detection. *IEEE Access*.

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., ... & Lane, N. D. (2020). Flower: A friendly federated learning research framework. arXiv preprint arXiv:2007.14390.

Blumenstock, J. E., & Kohli, N. (2023). Big data privacy in emerging market fintech and financial services: A research agenda. arXiv preprint arXiv:2310.04970.

Bogdanov, D., Kamm, L., Kubo, B., Rebane, R., Sokk, V., & Talviste, R. (2016). Students and taxes: a privacy-preserving study using secure computation. Proceedings on Privacy Enhancing Technologies, 2016(3), 117-135.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečný, J., Mazzocchi, S., McMahan, H. B., Van Overveldt, T., Petrou, D., Ramage, D., & Roselander, J. (2019). Towards federated learning at scale: System design. In Proceedings of Machine Learning and Systems 2019 (pp. 374-388)

Bost, R., Popa, R. A., Tu, S., & Goldwasser, S. (2020). Machine learning classification over encrypted data. Communications of the ACM, 63(6), 117-124.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., & Zhang, C. (2023, March 6).

Quantifying Memorization Across Neural Language Models. ArXiv.org. https://doi.org/10.48550/arXiv.2202.07646

Chamikara, M. A. P., Bertok, P., Khalil, I., Liu, D., & Camtepe, S. (2021). Privacy preserving distributed machine learning with federated learning. *Computer Communications*, *171*, 112–125. https://doi.org/10.1016/j.comcom.2021.02.014

Chen, J., & Sharma, S. (2021). Collective privacy management in online health communities: A social capital perspective. Information Systems Frontiers, 23(3), 619-633.

Chen, H., Gilad-Bachrach, R., Han, K., Huang, Z., Jalali, A., Laine, K., & Lauter, K. (2019). Logistic regression over encrypted data from fully homomorphic encryption. BMC medical genomics, 11(4), 3.

Chen, X., Li, Y., & Wang, S. (2022). Secure Multi-Party Computation for Machine Learning in Distributed Settings. *Journal of Secure Systems*, 45(2), 110-128.

Choi, S., Patel, D., Zad Tootaghaj, D., Cao, L., Ahmed, F., & Sharma, P. (2024). FedNIC: enhancing privacy-preserving federated learning via homomorphic encryption offload on SmartNIC. Frontiers in Computer Science, 6, 1465352.

Collins, L., Hassani, H., Mokhtari, A., & Shakkottai, S. (2022). Fedavg with fine tuning: Local updates lead to representation learning. Advances in Neural Information Processing Systems, 35, 10572-10586.

Dablain, D., Krawczyk, B., & Chawla, N. V. (2022). DeepSMOTE: Fusing deep learning and SMOTE for imbalanced data. IEEE transactions on neural networks and learning systems, 34(9), 6390-6404.

Dahouda, M. K., & Joe, I. (2021). A deep-learned embedding technique for categorical features encoding. IEEE Access, 9, 114381-114391.

Dannels, S. A. (2018). Research design. In The reviewer's guide to quantitative methods in the social sciences (pp. 402-416). Routledge.

Deloitte. (2020). *Digital Consumer Trends: The UK cut*. Deloitte United Kingdom; Deloitte. https://www2.deloitte.com/uk/en/pages/technology-media-and-telecommunications/articles/digital-consumer-trends.html

Enthoven, D., & Al-Ars, Z. (2021). An overview of federated deep learning privacy attacks and defensive strategies. In Studies in computational intelligence (pp. 173–196). https://doi.org/10.1007/978-3-030-70604-3_8

Fereidooni, H., Marchal, S., Miettinen, M., Mirhoseini, A., Möllering, H., Nguyen, T. D., ... & Sadeghi, A. R. (2021). SAFELearn: Secure aggregation for private federated learning. In 2021 IEEE Security and Privacy Workshops (SPW) (pp. 56-62). IEEE.

Ghaleb, F. A., Saeed, F., Al-Sarem, M., Qasem, S. N., & Al-Hadhrami, T. (2023). Ensemble synthesized minority oversampling-based generative adversarial networks and random forest algorithm for credit card fraud detection. IEEE Access, 11, 89694-89710.

Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., & Wernsing, J. (2016). CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In International Conference on Machine Learning (pp. 201-210).

Gruzd, A., & Hernández-García, Á. (2018). Privacy concerns and self-disclosure in private and public uses of social media. Cyberpsychology, Behavior, and Social Networking, 21(7), 418-428.

Haji, S. H., & Abdulazeez, A. M. (2021). Comparison of optimization techniques based on gradient descent algorithm: A review. PalArch's Journal of Archaeology of Egypt/Egyptology, 18(4), 2715-2743.

He, C., Balasubramanian, K., Ceyani, E., Yang, C., Xie, H., Sun, L., ... & Avestimehr, S. (2021). Fedgraphnn: A federated learning system and benchmark for graph neural networks. arXiv preprint arXiv:2104.07145.

IBM. (2021). *Cost of a Data Breach Study*. IBM; IBM Security. https://www.ibm.com/security/data-breach

Jayaraman, B., & Evans, D. (2019). Evaluating differentially private machine learning in practice. In 28th {USENIX} Security Symposium ({USENIX} Security 19) (pp. 1895- 1912).

Jiang, X., Kim, M., Lauter, K., & Song, Y. (2018). Secure outsourced matrix computation and application to neural networks. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and

Communications Security (pp. 1209-1222).

Jin, H., Raghavan, K., Papadimitriou, G., Wang, C., Mandal, A., Kiran, M., ... & Balaprakash, P. (2023). Graph neural networks for detecting anomalies in scientific workflows. The International Journal of High Performance Computing Applications, 37(3-4), 394-411.

Kairouz, E. by: P., & McMahan, H. B. (2021). Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*, *14*(1). https://doi.org/10.1561/2200000083

Kaissis, G. A., Makowski, M. R., Rückert, D., & Braren, R. F. (2020). Secure, privacy-preserving and federated machine learning in medical imaging. Nature Machine Intelligence, 2(6), 305-311.

Kamiri, J., & Mariga, G. (2021). Research Methods in Machine Learning: A content analysis. International Journal of Computer and Information Technology (2279-0764), 10(2). https://doi.org/10.24203/ijcit.v10i2.79

Karampasi, A., Radoglou-Grammatikis, P., Pawlicki, M., Choraś, R., de Pozuelo, R. M., Sarigiannidis, P., ... & Choraś, M. (2024, December). Towards Transparent AI-Powered Cybersecurity in Financial Systems: The Deployment of Federated Learning and Explainable AI in the CaixaBank pilot. In 2024 IEEE International Conference on Data Mining Workshops (ICDMW) (pp. 270-277). IEEE.

Kim, S., Yun, S., Lee, H., Gubri, M., Yoon, S., & Oh, S. J. (2023, July 4). *ProPILE: Probing Privacy Leakage in Large Language Models*. ArXiv.org. https://doi.org/10.48550/arXiv.2307.01881

Li, Y., Li, Y., Yan, Q., & Deng, R. H. (2019). Privacy leakage analysis in online social networks. Computers & Security, 87, 101551.

Liu, Q., Wang, G., & Wu, J. (2020). CPPM: A cloud-based privacy-preserving personal data management scheme. IEEE Transactions on Cloud Computing, 8(2), 472-483.

Liu, Y., Sarabi, A., Zhang, J., Naghizadeh, P., Karir, M., Liu, M., & Bailey, M. (2015). *Cloudy*

with a Chance of Breach: Forecasting Cyber Security Incidents This paper is included in the Proceedings of the 24th USENIX Security Symposium Cloudy with a Chance of Breach: Forecasting Cyber Security Incidents (pp. 1545–1562).

Liu, J., Wang, J. H., Rong, C., Xu, Y., Yu, T., & Wang, J. (2021). Fedpa: An adaptively partial model aggregation strategy in federated learning. Computer Networks, 199, 108468.

Lu, Y., Huang, X., Dai, Y., Maharjan, S., & Zhang, Y. (2020). Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. IEEE Transactions on Industrial Informatics, 16(6), 4177-4186.

Lu, C., Ma, W., Wang, R., Deng, S., & Wu, Y. (2023). Federated learning based on stratified sampling and regularization. Complex & Intelligent Systems, 9(2), 2081-2099.

Mccallister, E., Grance, T., & Scarfone, K. (2010). Special Publication 800-122 Guide to Protecting the Confidentiality of Personally Identifiable Information (PII) Recommendations of the National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-122.pdf

Mcmahan, H., Moore, E., Ramage, D., Hampson, S., & Agüera, B. (2017). *Communication-Efficient Learning of Deep Networks from Decentralized Data* (pp. 1273–1282).

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

Mimecast. (2024). Data Leakage Prevention. Mimecast. https://www.mimecast.com/content/data-leakage-prevention/

Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.
O'Reilly Media. (2015). *Building Microservices DESIGNING FINE-GRAINED SYSTEMS*.

Ozsahin, D. U., Mustapha, M. T., Mubarak, A. S., Ameen, Z. S., & Uzun, B. (2022, August). Impact of feature scaling on machine learning models for the diagnosis of diabetes. In 2022 International Conference on Artificial Intelligence in Everything (AIE) (pp. 87-94). IEEE.

Petronio, S. (2002). Boundaries of privacy: Dialectics of disclosure. SUNY Press.

Petronio, S. (2013). Brief status report on communication privacy management theory. Journal of Family Communication, 13(1), 6-14.

Prometheus. (2014). Configuration | Prometheus. Prometheus.io. https://prometheus.io/docs/prometheus/latest/configuration/configuration/

Qu, Y., Gao, L., Luan, T. H., Xiang, Y., Yu, S., Li, B., & Zheng, G. (2021). Decentralized privacy using blockchain-enabled federated learning in fog computing. IEEE Internet of Things Journal, 8(4), 2046-2059.

Rahman, M. M., Tabash, M. I., Salamzadeh, A., Abduli, S., & Rahaman, M. S. (2022). Sampling techniques (probability) for quantitative social science researchers: a conceptual guidelines with examples. Seeu Review, 17(1), 42-51.

Rezaei, S., & Liu, X. (2021). On the difficulty of defending self-supervised learning against model inversion. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 7602-7611)

Sanjalawe, Y. K., & Al-E'mari, S. R. (2023). Abnormal transactions detection in the ethereum network using semi-supervised generative adversarial networks. IEEE Access, 11, 98516-98531.

Shabtai, A., Elovici, Y., & Rokach, L. (2012). A survey of data leakage detection and prevention solutions. Springer Science & Business Media.

Shanmugam, L., Tillu, R., & Tomar, M. (2023). Federated learning architecture: Design, implementation, and challenges in distributed AI systems. Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online), 2(2), 371-384.

Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership inference attacks against machine learning models. In 2017 IEEE Symposium on Security and Privacy (SP) (pp. 3-18). IEEE.

Sun, T., Li, D., & Wang, B. (2022). Decentralized federated averaging. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(4), 4289-4301.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., & Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security (pp. 1-11).

Xie, D., & Li, D. (2025). Privacy-Preserving and Verifiable Personalized Federated Learning. Symmetry, 17(3), 361.

Xu, J., Glicksberg, B. S., Su, C., Walker, P., Bian, J., & Wang, F. (2019). Federated learning for healthcare informatics. Journal of Healthcare Informatics Research, 3(4), 414-434.

Xu, L., Huang, C., Wang, P., Wang, K., Wang, S., & Tian, Q. (2019). Efficient privacy- preserving learning for self-driving vehicles. IEEE Transactions on Intelligent Transportation Systems, 21(12), 5224-5236.

Xu, R., Baracaldo, N., Yi, Y., Anwar, A., & Ludwig, H. (2020). Hybrid Differential Privacy for Federated Learning. arXiv preprint arXiv:2010.02374.

Xu, X., Wang, H., Li, H., & Xu, L. (2019). A survey of privacy-preserving federated learning in the big data era. In 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS) (pp. 2339-2344). IEEE. https://doi.org/10.1109/HPCC/SmartCity/DSS.2019.00326

Wang, Y., Norcie, G., & Cranor, L. F. (2021). Privacy concerns and practices in online social networks: A qualitative study. Journal of the Association for Information Science and Technology, 72(7), 828-842.

Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated Machine Learning. *ACM Transactions on Intelligent Systems and Technology*, *10*(2), 1–19. https://doi.org/10.1145/3298981

Zerka, F., Barakat, S., Walsh, S., Bogowicz, M., Leijenaar, R. T., Jochems, A., ... & Lambin, P. (2020). Systematic review of privacy-preserving distributed machine learning from federated databases in health care. JCO clinical cancer informatics, 4, 184-200.

Zhao, B., Mopuri, K. R., & Bilen, H. (2020). iDLG: Improved deep leakage from gradients. arXiv preprint arXiv:2001.02610.

Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., & Gao, Y. (2021). A survey on federated learning.

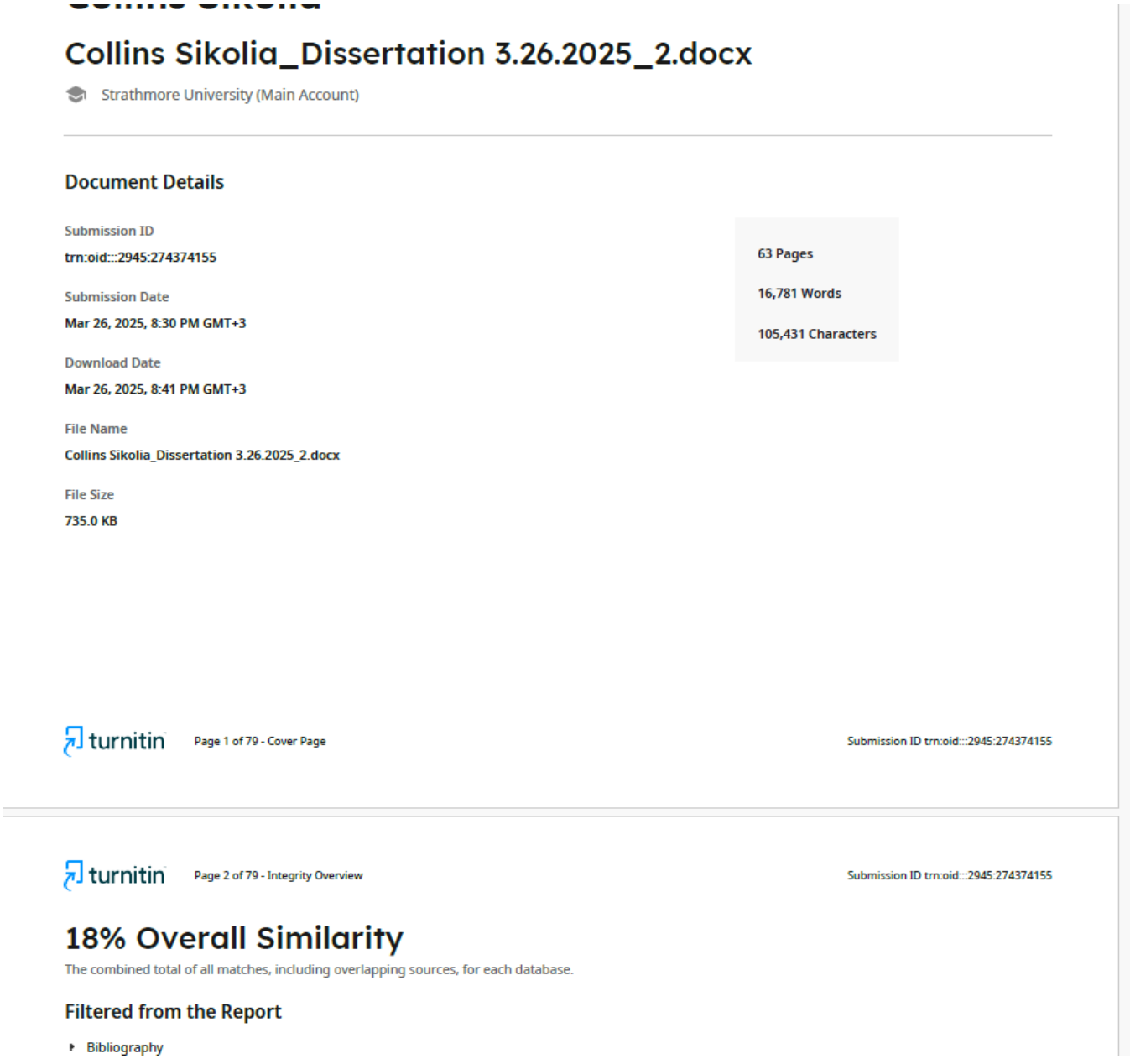
Knowledge-Based Systems, 216, 106775.

Zhang, X., Liu, S., Chen, X., Wang, L., Gao, B., & Zhu, Q. (2020). Health information privacy concerns, antecedents, and information disclosure intention in online health communities.

Information & Management, 57(2), 103260.

Appendices

Appendix A: Similarity Report



Collins Sikolia_Dissertation 3.26.2025_2.docx

Strathmore University (Main Account)

Document Details

Submission ID
trn:oid:::2945:274374155

Submission Date
Mar 26, 2025, 8:30 PM GMT+3

Download Date
Mar 26, 2025, 8:41 PM GMT+3

File Name
Collins Sikolia_Dissertation 3.26.2025_2.docx

File Size
735.0 KB

63 Pages

16,781 Words

105,431 Characters

turnitin Page 1 of 79 - Cover Page Submission ID trn:oid:::2945:274374155

turnitin Page 2 of 79 - Integrity Overview Submission ID trn:oid:::2945:274374155

18% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Appendix B: Ethical Clearance Report

Strathmore UNIVERSITY

10th December 2024

Mr Sikolia Collins,
collins.sikolia@strathmore.edu

Dear Mr Sikolia,

RE: Privacy-Preserving Machine Learning Tool for Mitigating against Data Leakage in Microservices Architectures

This is to inform you that SU-ISERC has reviewed and **approved** your above SU-masters proposal. Your application reference number is **SU-ISERC2467/24.** The approval period is from **10th December 2024 to 9th December 2025.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used.
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 72 hours of notification.
iv. Any changes anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 72 hours.
v. Clearance for the export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

Mr Ambrose Rachier,
Chairperson; SU-ISERC

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu

Appendix C: NACCOSTI Report

REPUBLIC OF KENYA

NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Ref No: 842843

Date of Issue: 23/January/2025

RESEARCH LICENSE

This is to Certify that Mr.. Collins Sikolia of Strathmore University, has been licensed to conduct research as per the provision of the Science, Technology and Innovation Act, 2013 (Rev.2014) in Nairobi on the topic: Privacy-Preserving Machine Learning Tool for Mitigating Against Data Leakage in Microservices Architectures for the period ending : 23/January/2026.

License No: NACOSTI/P/25/415274

842843

Applicant Identification Number

Director General
NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Verification QR Code

NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.

See overleaf for conditions