FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# RAMPART-FL: Federated Learning for Intrusion Detection in the Edge through Reinforcement-based Multi-Criteria Participant Selection

**Lucas Ferreira de Sousa**




Master in Informatics and Computing Engineering

Supervisor: Daniel Castro Silva

Supervisor: Sinan Wannous

Supervisor: Isabel Praça

July 28, 2025

# RAMPART-FL: Federated Learning for Intrusion Detection in the Edge through Reinforcement-based Multi-Criteria Participant Selection

**Lucas Ferreira de Sousa**

Master in Informatics and Computing Engineering

Approved by:

President: Professor Rui Carlos Camacho de Sousa Ferreira da Silva
Examiner: Professor Pedro Ricardo Morais Inácio
Supervisor: Professor Daniel Augusto Gama de Castro Silva

July 28, 2025

# Resumo

O desempenho e a equidade dos sistemas de Aprendizagem Federada dependem da estratégia de seleção de clientes, um desafio para os Sistemas de Deteção de Intrusão de Redes em ambientes heterogéneos da Internet das Coisas/Edge, onde os dados dos clientes são frequentemente não independentes e distribuídos de forma idêntica e as capacidades dos dispositivos variam. Esta dissertação apresenta RAMPART-FL, um sistema em que um agente de Aprendizagem por Reforço escolhe dinamicamente clientes para otimizar objectivos conflitantes. O agente é orientado por uma função de recompensa multicritério concebida para maximizar o desempenho do modelo, promover a imparcialidade na seleção dos participantes, penalizando seleção excessiva de um conjunto de clientes, e garantir a eficiência de utiliação dos recursos, penalizando a seleção de dispositivos com alta taxa de utilização. Efectuamos uma análise comparativa do nosso agente multicritério em relação a estratégias que optimizam objectivos únicos como o desempenho, a imparcialidade ou os recursos. Usando um sistema de deteção de intrusão de redes baseado numa rede adversária generativa treinada no conjunto de dados ToN-IoT, os nossos resultados demonstram que, embora todas as estratégias convirjam para um pico semelhante de F1-Score, o agente multicritério fá-lo com maior eficiência. Este aprende efetivamente a equilibrar a qualidade do modelo, a imparcialidade na seleção dos participantes e os custos dos recursos, alcançando uma pontuação F1-Score de aproximadamente 0,82, comparável a um modelo centralizado de referência. Mostramos que o agente progride através de fases de aprendizagem distintas, desde a exploração inicial até à gestão estável e a longo prazo, demonstrando a sua adaptabilidade em ambientes de aprendizagem federados dinâmicos.

# Abstract

The performance and fairness of Federated Learning systems depend on the client selection strategy, a challenge for Network Intrusion Detection Systems in heterogeneous Internet of Things/Edge environments where client data is often Non-Independent and Identically Distributed and device capabilities vary. This dissertation introduces RAMPART-FL, a system where a Reinforcement Learning agent dynamically chooses clients to optimize competing objectives. The agent is guided by a multi-criteria reward function designed to maximize model performance, promote participant fairness by penalizing client starvation, and ensure resource efficiency by penalizing the selection of costly devices. We perform a comparative analysis of our multi-criteria agent against strategies that optimize for single objectives like performance, fairness, or resources. Using a Generative Adversarial Network-based Network Intrusion Detection System trained on the ToN-IoT dataset, our results demonstrate that while all strategies converge to a similar peak F1-Score, the multi-criteria agent does so with greater efficiency. It effectively learns to balance model quality, participant fairness, and resource costs, achieving an F1-Score of approximately 0.82, comparable to a centralized baseline. We show that the agent progresses through distinct learning phases, from initial exploration to stable, long-term management, demonstrating its adaptability in dynamic federated learning environments.

# Acknowledgments

First and foremost, I want to acknowledge the support and faith placed upon me by my family, in particular by my parents and grandfather, over these five years of university.
I also want to acknowledge and congratulate my friends and colleagues who undertook this journey alongside me and who are also finishing their master's after many years of work and effort.
I want to thank Dr. Matilde Carvalho from the Orientation and Integration Office for helping me learn to manage the added stress that the course and dissertation incited.
I want to thank my friend and colleague Flávio Vaz, who undertook with me a dissertation in the field of AI for Cybersecurity and with whom I've shared many meetings this last year.
Last but not least, I want to show particular appreciation for the constant supervision, guidance, and advice given to me by my supervisors Professor Daniel Castro Silva, Professor Isabel Praça, and Sinan Wannous throughout the making of this dissertation and article.

Lucas Sousa

*"We alone can accomplish little, but joined by others,*
*we may yet build a bonfire to carry us heavensward"*

Alphinaud Leveilleur,
Final Fantasy XIV

# Contents


**1 Introduction** **1**
- 1.1 Motivation and Problem . . . 2
- 1.2 Objectives, Research Questions, and Methodology . . . 2
- 1.3 Contributions . . . 3
- 1.4 Document Structure . . . 4

**2 Context** **6**
- 2.1 The Internet of Things and Edge Computing . . . 6
- 2.2 Intrusion Detection for the Internet of Things . . . 7
- 2.3 Federated Learning . . . 7
- 2.4 Reinforcement Learning . . . 8
- 2.5 Summary . . . 9

**3 State of the Art** **10**
- 3.1 Literature Review . . . 10
  - 3.1.1 Search Queries . . . 11
  - 3.1.2 Screening . . . 11
  - 3.1.3 Tagging of Relevant Results . . . 12
- 3.2 Fields of Application . . . 14
- 3.3 Defensive Mechanisms and Security Approaches . . . 16
  - 3.3.1 Byzantine Fault Tolerance and Adversarial Robustness . . . 16
  - 3.3.2 Privacy-Preserving Techniques and Encryption . . . 17
  - 3.3.3 Blockchain Integration and Trust Management . . . 19
- 3.4 Advanced Learning Paradigms and Model Architectures . . . 20
  - 3.4.1 Generative Models and Data Augmentation . . . 20
  - 3.4.2 Meta-Learning and Adaptive Classification Systems . . . 21
- 3.5 System Optimization and Resource Management . . . 22
  - 3.5.1 Communication Efficiency and Latency Reduction . . . 23
  - 3.5.2 Model Aggregation and Heterogeneous Data Management . . . 24
- 3.6 Anomaly Detection Learning Methods . . . 26
- 3.7 Participant Selection . . . 27
- 3.8 Summary . . . 29

**4 Proposed Solution** **31**
- 4.1 Conceptual Framework . . . 31
  - 4.1.1 Simulated Environment: Server and Clients . . . 31
  - 4.1.2 Data Composition for a Realistic IDS Scenario . . . 32
  - 4.1.3 Intelligent Participant Selection Strategy . . . 32

4.1.4 System Orchestration Cycle . . . 32
4.1.5 Multi-Criteria Optimization in an IDS Context . . . 33
4.2 Framework and Dataset Selection . . . 33
4.2.1 Federated Learning Framework Comparison . . . 34
4.2.2 Simulation Environment . . . 37
4.2.3 Intrusion Detection Dataset Analysis . . . 37
4.3 Summary . . . 39

**5 Implemented Solution: RAMPART-FL** **40**
5.1 System Architecture . . . 40
5.2 Data, Preprocessing and Non-IID Distribution . . . 40
5.2.1 Rationale for a Semi-Supervised Approach . . . 40
5.2.2 Dataset and Preprocessing . . . 41
5.2.3 Initial Dataset Splitting . . . 42
5.2.4 Non-IID Data Distribution to Clients . . . 42
5.3 Machine Learning Model: GAN for Anomaly Detection . . . 43
5.3.1 Client-Side Semi-Supervised GAN Training . . . 44
5.3.2 Model Architecture . . . 44
5.3.3 Training and Evaluation Process . . . 45
5.4 The RLClientSelectionStrategy . . . 46
5.4.1 Client Attribute Store . . . 47
5.4.2 Strategy Bootstrapping: The Initial Rounds . . . 47
5.5 The Reinforcement Learning Agent . . . 48
5.5.1 Markov Decision Process Formulation . . . 48
5.5.2 Action Selection and Policy . . . 49
5.5.3 Multi-Criteria Reward Function . . . 50
5.5.4 Q-Learning Update . . . 52
5.6 Summary . . . 52

**6 Experimental Evaluation and Results** **53**
6.1 Comparative Analysis of Reward Strategies . . . 54
6.1.1 Client Selection & Fairness . . . 54
6.1.2 Model Performance & Convergence . . . 55
6.1.3 Resource Efficiency . . . 56
6.1.4 Conclusion of the Comparative Analysis . . . 58
6.2 Analysis of the Learning Phases of the Multi-Criteria Agent . . . 58
6.2.1 Stage 1: Exploration and Instability (Rounds $\approx 3-40$) . . . 59
6.2.2 Stage 2: Exploitation and Rapid Improvement (Rounds $\approx 40-100$) . . . 60
6.2.3 Stage 3: Convergence and Management (Rounds $\approx 100-400$) . . . 61
6.3 Summary . . . 62

**7 Conclusion and Future Work** **63**
7.1 Summary of Findings . . . 63
7.2 Additional Contributions . . . 64
7.2.1 Experimental Dataset . . . 64
7.2.2 Open-Source Framework . . . 64
7.3 Revisiting the Research Questions . . . 65
7.4 Discussion and Future Work . . . 67

**Bibliography** **69**

**A Literature Review Queries** **79**
A.1 IEEE Xplore . . . . . 79
A.2 ACM Digital Library . . . . . 80
A.3 Science Direct . . . . . 80
A.4 DBLP . . . . . 80
A.5 Scopus . . . . . 80

# List of Figures


3.1 Comparison of search results. . . . . 12
3.2 Distribution of irrelevant search results. . . . . 13
3.3 Distribution of relevant search results over time. . . . . 13
3.4 Distribution of tags among the relevant search results. . . . . 14

4.1 Conceptual system orchestration cycle . . . . . 34

5.1 High-level system architecture of RAMPART-FL. . . . . 41
5.2 Data splitting and distribution diagram. . . . . 43
5.3 GAN structure. . . . . 44

6.1 Client selection comparison. . . . . 55
6.2 Global performance comparison. . . . . 56
6.3 Performance-Only Agent Reward Dynamics. . . . . 57
6.4 Resource usage comparison. . . . . 57
6.5 RL Learning Dynamics in Stage 1. . . . . 59
6.6 Distribution of model losses. . . . . 60
6.7 RL Learning Dynamics in Stage 2. . . . . 60

# List of Tables


3.1 Search results per database. . . . . 12
3.2 Gap Analysis: Byzantine Fault Tolerance . . . . . 17
3.3 Gap Analysis: Privacy-Preserving Techniques . . . . . 19
3.4 Gap Analysis: Blockchain Integration . . . . . 20
3.5 Gap Analysis: Generative Models . . . . . 21
3.6 Gap Analysis: Meta-Learning and Adaptive Systems . . . . . 22
3.7 Gap Analysis: Communication Efficiency . . . . . 24
3.8 Gap Analysis: Model Aggregation and Heterogeneous Data . . . . . 26
3.9 Gap Analysis: Anomaly Detection Learning Methods . . . . . 27
3.10 Gap Analysis: Participant Selection Methods . . . . . 29

4.1 Comparison of FL Frameworks . . . . . 36
4.2 Comparison of Intrusion Detection Datasets . . . . . 38

5.1 Client attributes store. . . . . 47

6.1 Centralized baseline model performance. . . . . 53

7.1 RAMPART-FL dataset columns . . . . . 65

# Abbreviations

| | |
|---|---|
| **ADMM** | Alternating Direction Method of Multipliers |
| **AI** | Artificial Intelligence |
| **APRIL** | Approximate Reconstruction and Poisoning via Iterative Learning |
| **BCE** | Binary Cross-Entropy |
| **CPU** | Central Processing Unit |
| **CVAE** | Conditional Variational Autoencoder |
| **DNS** | Domain Name Service |
| **EIDS** | Evolvable Intrusion Detection System |
| **FD2K** | Federated Deep Key Agreement |
| **FEDGAN-IDS** | Federated Generative Adversarial Network for Intrusion Detection Systems |
| **FEDGT** | Federated Learning Group Testing framework |
| **FEDMCCS** | Federated Multicriteria Client Selection |
| **FEDPCA** | Federated Principal Component Analysis |
| **FEDVB** | Federated Learning with Vertical Blocking aggregation |
| **FL** | Federated Learning |
| **FLDP** | Federated Learning with Differential Privacy |
| **FLUD** | Federated Learning Update Digest |
| **FFL** | Fragmented Federated Learning |
| **GAN** | Generative Adversarial Network |
| **GDPR** | General Data Protection Regulation |
| **GENAI** | Generative AI |
| **GPU** | Graphics Processing Unit |
| **HFL** | Hierarchical Federated Learning |
| **HIFGL** | Hierarchical Federated Graph Learning |
| **HIPAA** | Health Insurance Portability and Accountability Act |
| **HTTP** | Hypertext Transfer Protocol |
| **ICMI** | Initialization-Centric Model Initialization |
| **IDS** | Intrusion Detection System |
| **IID** | Independent and Identically Distributed |
| **IIoT** | Industrial Internet of Things |
| **IoMT** | Internet of Medical Things |
| **IoT** | Internet of Things |
| **LLM** | Large Language Model |
| **LOTTO** | Secure Participant Selection Against Adversarial Servers in Federated Learning |
| **LWE** | Learning with Errors (a cryptographic security technique) |
| **MDP** | Markov Decision Process |
| **MITFED** | Blockchain-based Federated Learning Framework for Secure Collaboration |

| | |
|---|---|
| **MITM** | Man-in-the-Middle (Attack) |
| **ML** | Machine Learning |
| **MLP** | multi-layer perceptrons |
| **MUB** | Model Update-Based Scheme |
| **RL** | Reinforcement Learning |
| **Non-IID** | Non-Independent and Identically Distributed |
| **NVFLARE** | NVIDIA Federated Learning Application Runtime Environment |
| **P2PK-SMOTE** | Peer-to-Peer Federated Learning with Synthetic Minority Over-Sampling Technique |
| **PCA** | Principal Component Analysis |
| **PHEC** | Privacy-Hardened Encryption Computing |
| **QSA** | Quondam Signature Algorithm |
| **RAM** | Random Access Memory |
| **SDN** | Software-Defined Networking |
| **SDVN** | Software-Defined Vehicular Network |
| **SFMD** | Semi-Supervised Federated Malicious Traffic Detection |
| **SNN** | Spiking Neural Network |
| **SMPC** | Secure Multiparty Computation |
| **SURFS** | Sustainable Intrusion Detection with Hierarchical Federated Spiking Neural Networks |
| **TEE** | Trusted Execution Environment |
| **TFF** | TensorFlow Federated |
| **TRUSTNETFL** | Trusted Client Aggregation in Federated Learning |
| **VANET** | Vehicular Ad Hoc Network |
| **XGBOOST** | eXtreme Gradient Boosting |
| **ZEKOC** | Zero-Knowledge Clustering for Federated Learning |

# Chapter 1

# Introduction

The rapid growth of Internet of Things (IoT) devices, combined with the widespread move toward edge computing, has completely transformed how we design network architectures. IoT ecosystems, encompassing everything from simple sensors to complex industrial machinery, generate vast data streams at the network periphery. Edge computing aims to process this data closer to its source, offering benefits in latency and bandwidth but also decentralizing security responsibilities (Ghimire & Rawat, 2022). This distributed nature, coupled with the often resource-constrained characteristics of IoT devices, presents a complex attack surface, challenging traditional cybersecurity measures (Ghimire & Rawat, 2022).

Network Intrusion Detection Systems (IDS) play a crucial role in spotting malicious activities. But traditional IDS approaches, whether they rely on signatures or anomaly detection, run into serious problems when you try to deploy them across these vast and varied IoT/edge environments. Centralized IDS architectures struggle with scalability, latency, and privacy concerns when handling data from a multitude of edge devices (Ghimire & Rawat, 2022). While advancements in Artificial Intelligence (AI) and Machine Learning (ML) have enhanced IDS capabilities, deploying sophisticated ML-based IDS effectively on resource-limited edge devices remains a hurdle (Ghimire & Rawat, 2022; Hamouda et al., 2024).

Federated Learning (FL) offers a promising solution to these challenges by allowing ML models to be trained collaboratively across distributed devices while keeping the raw data in place (Konečný et al., 2016). This approach naturally protects privacy by keeping data local and can help reduce communication bottlenecks. But implementing FL successfully is not trivial and brings its own set of complications. Edge environments are diverse, which creates significant statistical heterogeneity (Non-IID data) and system heterogeneity (varying device capabilities), which difficult model convergence (Ghimire & Rawat, 2022; Alazab et al., 2022). This dissertation argues that the method by which the clients are selected for participation in each training round is an underexplored component of FL, when compared to its other processes. Our research focuses on developing a smarter selection strategy that can make FL-based IDS systems more effective.

## 1.1 Motivation and Problem

As networks become more complex and cyberattacks become increasingly sophisticated, the need for efficient real-time intrusion detection is critical (Alazab et al., 2022). Traditional centralized approaches, which require sending data from edge devices to cloud-based servers for analysis, present substantial challenges in edge environments, including increased latency, privacy risks, and reliance on a single point of failure (Ghimire & Rawat, 2022).

FL, first introduced by Google (Konečný et al., 2016), offers a compelling alternative by decentralizing the training process. In the FL model, edge devices collaboratively train a shared global model without transmitting sensitive raw data. Instead, after local training on their data, each device sends only its model updates (model weights) to a central server. These updates are aggregated into an improved global model, which is then redistributed to the devices. This cyclical process allows all participants to benefit from collective training while preserving data privacy (Alazab et al., 2022; Hamouda et al., 2024).

Despite its advantages, the practical application of FL for IDS in IoT/edge settings is fraught with challenges. These include communication overhead during model update exchange (Ghimire & Rawat, 2022), and the considerable difficulty of achieving robust model convergence when faced with heterogeneous device capabilities and, critically, non-independent and identically distributed (Non-IID) data across clients. Furthermore, the limited computational resources on many edge devices can impede the training of sophisticated IDS models, and the FL process itself is not immune to security threats like model poisoning or inference attacks (Alazab et al., 2022; Hamouda et al., 2024).

The core problem addressed by this dissertation is the sub-optimality of current participant selection mechanisms within FL. Standard approaches often rely on random or simplistic heuristic-based client selection, which fails to adequately address the intertwined challenges of non-IID data and system heterogeneity. This can lead to inefficient training, slow convergence, biased global models, and unfair distribution of the workload (Abdulrahman et al., 2021). Therefore, there is a pressing need for an intelligent and adaptive participant selection strategy that can dynamically choose the most suitable clients for each training round to optimize the overall federated process.

This work also aligns with United Nations Sustainable Development Goal 9 by contributing to the development of resilient and secure digital infrastructure. By fostering innovation in cybersecurity for IoT and edge networks, it supports the foundation for sustainable industry, innovation, and infrastructure.

## 1.2 Objectives, Research Questions, and Methodology

To tackle the need for a smarter participant selection strategy, this dissertation introduces and tests RAMPART-FL, a framework that uses Reinforcement Learning (RL) to handle the complexities of federated environments. Our main goal is to research, build, and assess this adaptive strategy,

looking at how it can improve the FL process by finding the right balance between model performance, participant fairness, and resource efficiency. While the selection mechanism is designed to be model-agnostic, its efficacy is demonstrated by applying it to the federated training of an IDS model. The focus is not on the detection model itself, but on the principles of the RL-based selection mechanism. To guide this research, the following questions are posed:

**RQ1: What are the effective design principles for an RL agent's state representation and reward function in a federated IDS, and how do these design choices manifest in the agent's learning behavior over time?** This question focuses on the foundational design of the RL agent. It investigates the architecture of its state representation and its multi-criteria reward function. We want to understand how these design decisions control how the agent's policy changes over time, and whether they can successfully guide the learning process to develop a stable, effective strategy.

**RQ2: How does a multi-criteria, Reinforcement Learning-driven participant selection mechanism adapt its policy to successfully guide a federated IDS model to convergence in an environment characterized by high system heterogeneity and Non-IID data?** This question investigates the dynamic adaptation process of the agent within a challenging federated environment. The focus is on understanding how, given a fixed, highly heterogeneous setting, the agent's learned selection policy influences the training dynamics and contributes to achieving robust global model convergence. It also examines the agent's learning process to explain how its strategy evolves to manage the challenges of system and statistical heterogeneity.

**RQ3: How does a multi-criteria RL-based participant selection strategy compare against strategies driven by single-objective reward functions (i.e., performance-only, fairness-only, and resource-only) in a simulated federated IDS environment?** Building an intelligent agent for complex IoT/edge environments means dealing with competing goals. Here, we compare our multi-objective approach against simpler strategies that optimize for just one component at a time. Does balancing performance, fairness, and resource efficiency work better than focusing on performance alone? What about strategies that only care about fairness or resource usage? The hypothesis is that single-minded optimization often is not ideal; a performance-obsessed strategy might completely ignore certain clients, while a resource-focused one could produce a useless model. These comparisons help us understand which approach delivers practical FL systems.

## 1.3 Contributions

This dissertation makes the following contributions to the field of FL, particularly concerning its application to IDSs in IoT/edge environments:

1. **A Novel RL-Driven Participant Selection Strategy:** The primary scientific contribution is the design, implementation, and detailed specification of RAMPART-FL. This strategy

features an RL agent guided by a multi-criteria reward function and a tailored state representation to balance the competing objectives of model performance, participant fairness, and resource efficiency, showing that it can work as a practical alternative to centralized training.

2. **Empirically-Grounded Insights into Adaptive FL Dynamics:** The dissertation provides a detailed analysis of the agent's learning process, identifying key mechanisms within the reward structure that are critical for helping the model reach convergence. We show how performance incentives can speed up improvement, while mechanisms that encourage broader client participation help stabilize training and build stronger, more generalizable models from non-IID data.

3. **An Open-Source Framework and Dataset for Reproducible Research:** The primary practical contribution is the public release of the RAMPART-FL framework as open-source code, accompanied by a curated experimental dataset. The framework provides the necessary components to replicate the experiments and serves as a foundation for future research into RL-based participant selection. The dataset, capturing the round-by-round decisions and outcomes of the agent, offers a valuable resource for benchmarking other selection strategies.

By addressing these areas, this dissertation advances the understanding and application of intelligent participant selection in FL, providing a pathway towards more effective, efficient, and fair decentralized learning solutions for demanding real-world problems.

## 1.4 Document Structure

This dissertation is organized into the following chapters to systematically address the research objectives:

1. **Chapter 1: Introduction.** This chapter (the current one) provides the overall context, motivates the research by outlining the problem of participant selection in FL for Intrusion Detection in IoT/edge environments, defines the research questions, outlines the methodology, and states the expected contributions.

2. **Chapter 2: Context.** This chapter delves deeper into the foundational concepts. It covers IoT and edge computing security challenges, an overview of IDS and their role in cybersecurity, it explores the role of AI in intrusion detection, with emphasis on its applications and limitations in decentralized environments, the principles of FL, including its benefits and inherent challenges, and the fundamentals of RL as applied to optimizing FL processes.

3. **Chapter 3: State of the Art.** This chapter presents a comprehensive review of existing literature and foundational research. It will synthesize prior research on core FL aspects, including client/aggregation server dynamics, model update security, model aggregation techniques, challenges such as heterogeneous data, classification in FL, latency, and communication overhead. Building upon this foundation, the review will then focus more specifically on: (a) current approaches in FL for cybersecurity; (b) research into unsupervised and semi-supervised training methodologies for anomaly detection; (c) existing participant selection strategies in FL, encompassing statistical and heuristic methods; and (d) the application of RL to FL. The chapter will also survey advanced ML models for intrusion detection. This comprehensive review will establish the context for the approach of the dissertation, highlighting effective techniques and identifying the existing gap in research concerning adaptive, multi-criteria RL-based participant selection, which this work aims to address.

4. **Chapter 4: Proposed Solution.** This chapter outlines the conceptual framework for the RAMPART-FL system. It details the high-level architecture, the objectives for the intelligent selection strategy, the design of the simulated heterogeneous environment, the semi-supervised data composition, and the overall system orchestration cycle. It also justifies the selection of the core frameworks and datasets used.

5. **Chapter 5: Implemented Solution.** This chapter provides a detailed overview of the implementation of the system. It describes the overall architecture of RAMPART-FL, the data preparation and distribution pipeline, the specifics of the detection model, and details the core technical contribution: the design of the RL agent, its integration strategy, and its multi-criteria reward function.

6. **Chapter 6: Experimental Evaluation and Results.** This chapter covers the experimental setup, shows the results, and explains what they mean. It describes the specific experiments we ran to compare RAMPART-FL against single-objective methods, explains the evaluation metrics we used, and gives a thorough analysis of the findings concerning model performance, system efficiency, and fairness.

7. **Chapter 7: Research Contributions: Dataset and Code.** This chapter outlines the contributions made to the research community beyond the findings of this dissertation. It describes the curated experimental dataset and the developed open-source RAMPART-FL framework.

8. **Chapter 8: Conclusion and Future Work.** This final chapter summarizes the key findings of the dissertation, revisits the research questions, discusses the limitations of the current work, and proposes potential avenues for future research in adaptive participant selection for FL in cybersecurity and other domains.

These chapters work together to give a comprehensive look at the RAMPART-FL system, offering insights into the application of RL for optimizing FL in real-world-inspired scenarios.

# Chapter 2

# Context

This chapter provides the background information required to understand the topic of this dissertation. It begins by defining the IoT and its associated security challenges and establishing the environment in which this work is set. It then explains the role of IDSs in securing these environments and introduces FL as a privacy-preserving solution for training them. Finally, it outlines the fundamentals of RL, the mechanism used to optimize the FL process.

## 2.1 The Internet of Things and Edge Computing

The IoT refers to the vast and growing network of interconnected physical devices, embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the internet (Ghimire & Rawat, 2022). These devices range from everyday consumer gadgets—such as smart home assistants, wearable fitness trackers, and connected security cameras—to large-scale industrial and civic infrastructure, including sensors in manufacturing plants (Industrial IoT), connected vehicles, and smart city components for managing traffic or public utilities (Nower, 2019). This paradigm, along with edge computing, which involves processing data closer to its source rather than in a centralized cloud, has enabled a new generation of smart applications in domains like healthcare, transportation, and industrial automation (Industry 4.0) (Hamouda et al., 2024).

The growth of these devices has brought with it major privacy and security concerns. IoT devices constantly collect data, creating privacy concerns when it comes to personal and sensitive information. This data, such as location, and biometric data that can be used to monitor, profile, and manipulate people (Nower, 2019). Unauthorized access to this data can result in serious privacy breaches. On top of that, the distributed nature of IoT further complicates their security. Most IoT devices have limited resources, with restricted processing power, memory (RAM), and battery life, making it difficult to build strong security features directly into the devices. The sheer scale and diversity of these networks create a vast attack surface, rendering them vulnerable to a wide range of cyber threats that traditional, centralized security models struggle to address effectively due to issues of latency, scalability, and privacy (Ghimire & Rawat, 2022; Alazab

et al., 2022). Due to this problem, specialized security tools like IDSs, built specifically for IoT environments, are necessary.

## 2.2 Intrusion Detection for the Internet of Things

An IDS is a security technology that monitors network traffic by inspecting data packets for malicious actions or policy violations, acting as a critical line of defense in identifying and reporting threats (Ali et al., 2024). The two primary detection methodologies are signature-based and anomaly-based. Signature-based detection works by matching the content and metadata of network packets against a database of known attack patterns, or "signatures." This approach is highly effective for known threats but is inherently unable to identify novel or "zero-day" attacks. In contrast, anomaly-based detection first establishes a baseline model of normal network behavior and then flags any significant deviation from this baseline as a potential intrusion. This allows it to detect previously unseen threats, but can sometimes result in false positives if unusual but legitimate activity occurs (Ali et al., 2024).

While powerful, using these IDS methods in IoT environments is challenging. The application of these IDS methodologies in IoT environments presents significant challenges. The scale of device deployment and the inherent resource constraints of IoT hardware—including limited processing power, memory, and energy—complicate traditional monitoring approaches. Furthermore, conventional IDS solutions face difficulties with the high volume of encrypted traffic common in IoT networks and often fail to meet the low-latency requirements of real-time applications (Pinto et al., 2023). The advent of AI and ML has enhanced the capabilities of modern anomaly-based systems, which can identify complex patterns in large datasets to improve threat detection accuracy. Notably, certain ML techniques can perform traffic analysis on encrypted data by analyzing metadata and traffic flows without requiring payload inspection (Mohammad Lotfollahi and Mahdi Jafari Siavoshani and Ramin Shirali Hossein Zade and Mohammdsadegh Saberian, 2020). However, the centralized training of these sophisticated models remains impractical in the privacy-sensitive and resource-constrained context of IoT, creating a clear need for a decentralized training paradigm like FL (Ali et al., 2024).

## 2.3 Federated Learning

A foundational understanding of ML is essential for comprehending FL. In ML, a *model* is a mathematical function defined by a set of internal parameters, or *weights*, which maps input data (e.g., network packet features) to an output (e.g., a classification of malicious traffic). The process of *training* involves iteratively adjusting these weights to minimize a "loss" or "error" function, which quantifies the discrepancy between the model's predictions and the ground-truth labels in a dataset. The objective of training is to derive a set of weights that enables the model to generalize effectively, making accurate predictions on new, unseen data (Chopra & Khurana, 2023).

Traditionally, this training process has been centralized, requiring all training data to be collected on a single server or in a data center. However, as a response to the growing privacy and scalability challenges posed by decentralized data, a new FL paradigm was introduced by researchers at Google in 2017 (Konečný et al., 2016). FL is a ML approach that allows multiple devices or data sources to work together to train a model without having to move the raw data from where it lives. A central server coordinates the training rounds. During each round, the server sends out the current version of the global model to some of the participating clients. These clients then independently train the model on their local data and send back only the updated model weights, not the data itself, to the server. The server then aggregates these updates to produce an improved global model, which is used as the starting point for the next round (Kairouz et al., 2021). This iterative process allows the model to learn from a vast and diverse pool of data while preserving user privacy by keeping sensitive information localized.

FL still has challenges, namely statistical heterogeneity or the non-IID (independent and identically distributed) data problem. In conventional ML, it is often assumed that training data is IID, meaning that all data points are drawn from the same underlying distribution and are independent of each other, guaranteeing that a model trained on a sample of the data will generalize to the entire data population (Konečný et al., 2016). In FL, often this assumption does not hold; each client's local dataset is shaped by their unique usage patterns, environments, and demographics. Training on the complete, aggregated dataset would allow a model to learn from the overall data distribution. However, in FL, models are trained on small, non-IID portions of the data, and their resulting weights are then averaged. This averaging can allow for the scenario where a model update that is highly effective for one client's data distribution may be detrimental to another's. A simple average of such conflicting updates can result in a global model that performs poorly for all participants, a key distinction from centralized training, where the entire dataset's structure can be learned cohesively (Kairouz et al., 2021).

Further challenges arise from the operational mechanics of FL. The exchange of model updates between the server and a large client population can create significant communication overhead, a primary bottleneck in FL systems (Konečný et al., 2016). Another critical aspect is participant selection. In large-scale deployments, the selection of all clients in every training round is infeasible due to device availability and resource constraints. Consequently, a mechanism is required to select a subset of clients for each round. The selection strategy directly impacts model performance, convergence speed, and fairness. A naive random selection, for example, may lead to biased models or inefficient training if the chosen clients are not representative of the overall population or possess limited resources (Kairouz et al., 2021).

## 2.4 Reinforcement Learning

RL is a paradigm of ML where an agent learns to make decisions by interacting with an environment to achieve a specific goal. Unlike supervised learning, which relies on labeled data, or unsupervised learning, which seeks to find patterns in unlabeled data, RL agents learn from the

consequences of their actions, typically in the form of rewards or punishments. This learning process is iterative and allows the agent to develop a strategy or policy to maximize its cumulative reward over time (Sutton, 1992).

The key components that define an RL problem are as follows (Sutton, 1992):

- **Agent:** The learner or decision-maker that interacts with the environment.
- **Environment:** The external system with which the agent interacts, providing observations and rewards.
- **State ($s$):** A representation of the current situation or configuration of the environment as perceived by the agent.
- **Action ($a$):** A choice made by the agent that influences the environment's state.
- **Reward ($r$):** A scalar feedback signal from the environment that indicates the immediate desirability of an action taken in a given state. The agent's objective is to maximize the total reward it receives in the long run.
- **Policy ($\pi$):** The agent's strategy, which maps states to actions. It defines the agent's behavior at a given time.
- **Value Function ($V(s)$ or $Q(s,a)$):** A function that estimates the expected cumulative reward an agent can receive starting from a particular state (State-Value Function, $V(s)$) or a state-action pair (Action-Value Function, $Q(s,a)$) and following a specific policy.

RL is particularly well-suited for optimizing sequential decision-making problems in dynamic and complex environments where explicit programming of optimal behavior is challenging. Its ability to learn through trial and error makes it applicable to a wide range of tasks, including game playing, robotics, resource management, and, as explored in this dissertation, the optimization of processes within FL systems, such as participant selection (Sutton, 1992).

## 2.5 Summary

This chapter covered the key concepts that form the foundation of this dissertation. We started by looking at the IoT, exploring its network of connected devices and the security and privacy problems that come from its distributed and resource-limited nature. The chapter then examined IDSs as a main defense tool and the difficulties of using them in IoT settings. This brought us to FL, a distributed ML approach that allows collaborative model training without moving sensitive data to a central location. The fundamental process of FL, along with its key challenges, such as statistical heterogeneity (non-IID data), communication overhead, and participant selection, was discussed. We wrapped up by covering the basic principles of RL, explaining its main components like agents, states, actions, and rewards, and showing how it works as a useful tool for tackling sequential decision-making challenges. All of these topics set up the background needed for the research discussed in the following chapters.

# Chapter 3

# State of the Art

The state-of-the-art chapter begins by detailing the process of systematic literature review methodology used to identify, tag, and assess relevant academic papers. It also explores the latest developments in FL related to IDSs within edge and IoT environments. This section builds on the large body of literature covered in the previous chapter by focusing on novel approaches found in the field. It delves into key aspects of FL, such as model aggregation techniques, security, and the handling of heterogeneous data. Furthermore, it expands the review to two critical areas for this dissertation: the evolution of anomaly detection learning methods from unsupervised to semi-supervised paradigms, and the growing field of intelligent participant selection, which is central to the proposed solution.

***A note on the scope of this review: The following sections provide a broad overview of the state of the art in FL, including topics such as model security and aggregation techniques that are not the primary focus of the final contribution of this dissertation. This initial, wide-ranging research was conducted early in the project to gain a comprehensive understanding of the challenges in applying FL to intrusion detection. Although the research focus was later refined to concentrate specifically on the problem of participant selection, this broader review has been retained as it provides valuable context for the issues surrounding FL and reflects the foundational knowledge that informed the eventual direction of this work.***

## 3.1 Literature Review

This section details the search queries that were iteratively conducted and refined across five prominent databases: IEEE Xplore[1], ACM Digital Library[2], ScienceDirect[3], DBLP[4], and Scopus[5], to gather literature on intrusion detection, IoT, FL, cybersecurity, and AI. Using specific

[1] https://ieeexplore.ieee.org
[2] https://dl.acm.org
[3] https://www.sciencedirect.com
[4] https://dblp.org
[5] https://www.scopus.com

inclusion and exclusion criteria, papers were selected based on their relevance to these topics, ensuring a well-curated set of resources.

### 3.1.1 Search Queries

For each database, we started with a search query, which we iteratively refined until it yielded a satisfactory number of results. The approach for each query was the same, make several subqueries, separated by an *AND* operator. Each subquery pertained to one of the subjects of this dissertation, enumerating several keywords and synonyms separated by an *OR* operator.

Those keywords and synonyms are as follows:

- **In the subject of AI:** AI, Artificial Intelligence, ML, Machine Learning, Generative, GenAI, LLM, Large Language Model, Neural Network, and Transformer,
- **In the subject of Federated Learning:** Federated, Federated Learning, FL, and Collaborative Learning
- **In the subject of Network Security** Network Security, Network Traffic, Defense,
- **In the subject of Intrusion Detection:** Intrusion Detection, Intrusion Detection System, IDS, Threat Detection, Attack Detection, and Anomaly Detection.
- **In the Subject of the Internet of Things:** IoT, Internet of Things, and Edge.

Only papers as old as 2017 were selected, due to it being the inception date of transformer-based AI models.

The final query used for each database can be consulted in appendix A. Note that the queries for ScienceDirect and DBLP are much simpler and more general due to the limitations on query complexity and the number of logical operators.

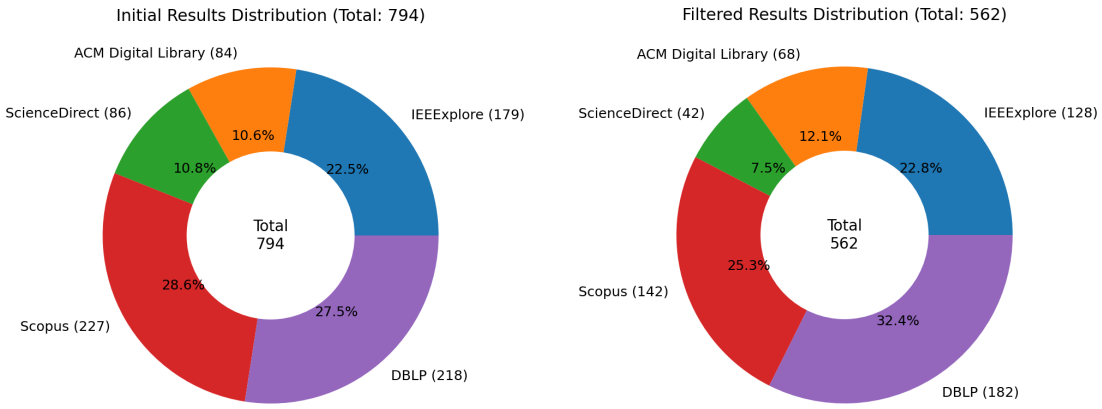
We were left with 794 results. Table 3.1 shows, for each database, the total and relevant number of results yielded and Figs. 3.1a and 3.1b show the distribution of results.

### 3.1.2 Screening

This stage entailed the careful filtering of the obtained results, based on exclusion criteria, and duplicate removal.

The exclusion criteria used and the number of results filtered out are listed below.

- **Results that focused on solutions that shared the device's data:** 9 results
- **Results that focused on a specific kind of attack, instead of focusing on general Intrusion Detection:** 35 results
- **Results that were out of the scope of the dissertation or otherwise unrelated to the topic:** 123 results




(a) The distribution of the results of the search queries.

(b) The distribution of relevant results obtained from the search queries.

Figure 3.1: Comparison of search query total and relevant results.

- **Conference proceedings:** 32 results

After excluding papers and removing duplicates, we were left with 488 relevant results.

The distribution of the irrelevant results can be found in Fig. 3.2.

As can be seen in Fig. 3.3, research on the topic of this dissertation has increased almost exponentially over the last few years, with emphasis on 2023 and 2024 at the time of writing.

Table 3.1: Search results from various databases with relevant result counts

| Database | Results | Relevant Results |
|---|---|---|
| IEEE Xplore | 179 | 128 |
| ACM Digital Library | 84 | 68 |
| ScienceDirect | 86 | 42 |
| DBLP | 218 | 182 |
| Scopus | 227 | 142 |

### 3.1.3 Tagging of Relevant Results

Due to the many relevant results, we decided to tag them according to the various topics they approached. The tagging criteria and the number of results with each tag are listed below:

- **Papers that focus on the application of FL in various contexts and fields:** 178 results
- **Papers that consider Blockchains as a viable solution to Federated Intrusion Detection:** 48 results
- **Papers that tackle the topic of heterogeneous data in IoT:** 23 results
- **Papers that present novel approaches to this dissertation's focused problems:** 158 results

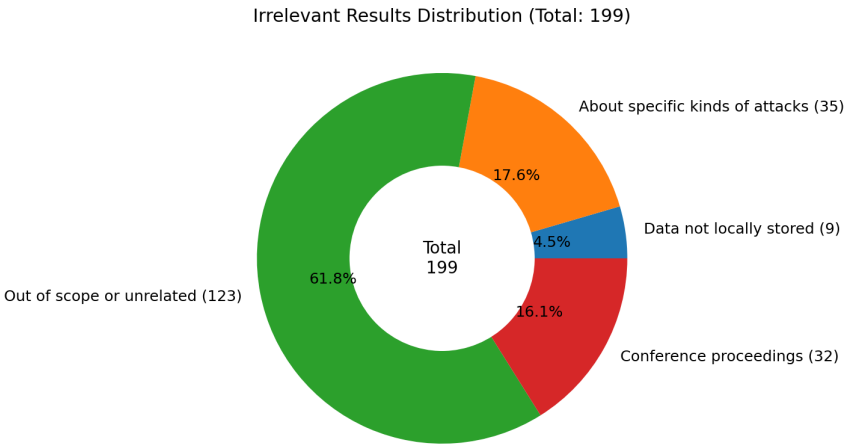



Figure 3.2: Distribution of irrelevant search results.

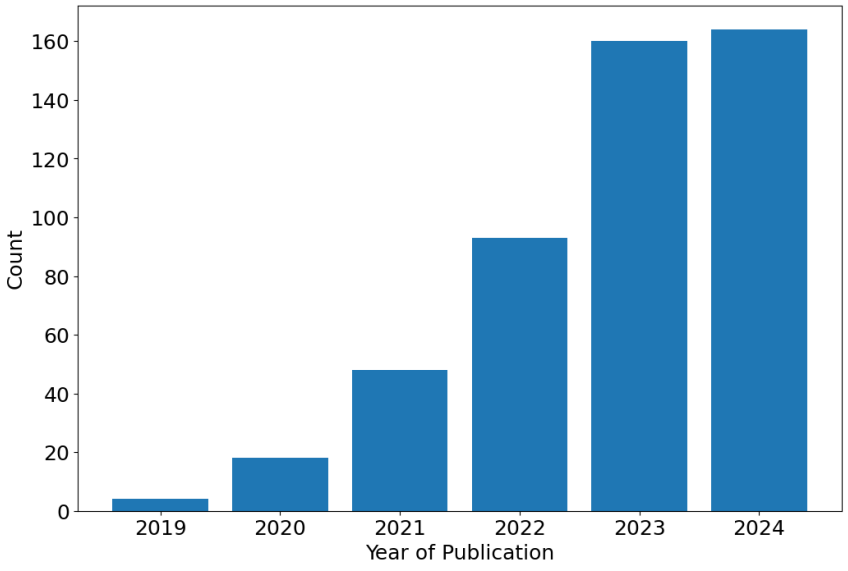



Figure 3.3: Distribution of relevant search results over time.

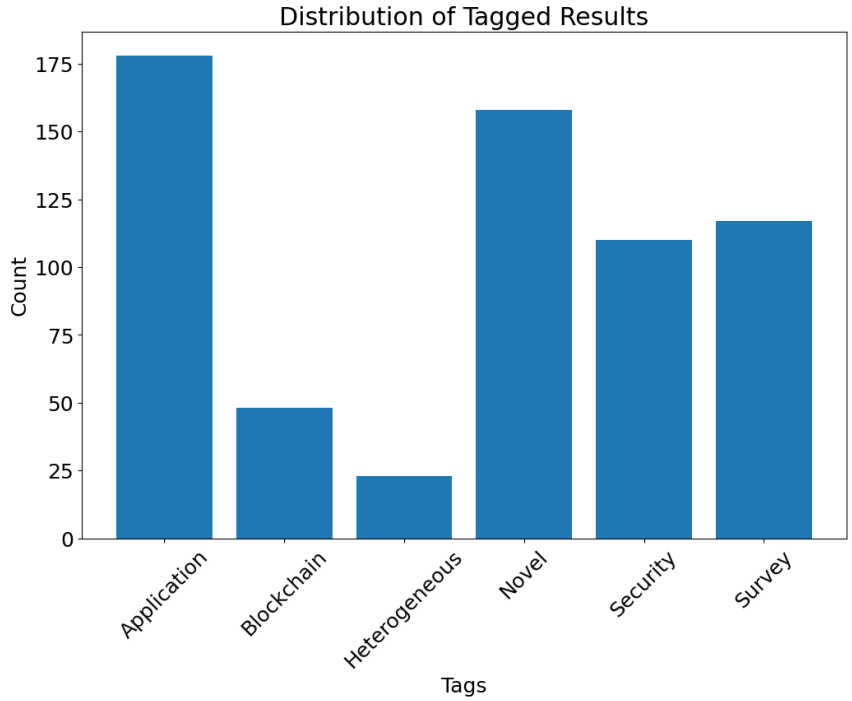



Figure 3.4: Distribution of tags among the relevant search results.

- **Papers that explore and tackle the unique security concerns of FL:** 110 results
- **Surveys, studies, and literature reviews on this dissertation's topics:** 117 results

Figure 3.4 shows the distribution of the tags among the relevant results.

## 3.2 Fields of Application

Based on the literature found, it is clear that privacy-preserving FL has been applied across various domains to address challenges such as data privacy, security, and operational efficiency. Below, the primary application areas are discussed with the benefits of FL highlighted.

**1. Cybersecurity in Industrial and IoT Networks** FL is widely adopted in industrial and IoT networks to enhance intrusion detection and mitigate cyber threats. By enabling privacy-preserving collaborative frameworks, FL reduces communication overhead and optimizes resource use. For example, Friha et al. (2022) propose FL-based intrusion detection for agricultural IoT, while Abdel-Basset et al. (2022) leverage blockchain to secure industrial edge networks. These approaches address the rising prevalence of cyberattacks while ensuring the confidentiality of sensitive industrial data.

**2. Healthcare and Internet of Medical Things (IoMT)** In healthcare, FL ensures sensitive patient data remains private while facilitating collaborative learning across distributed devices.

It strengthens IoMT cybersecurity, enabling real-time attack detection and safeguarding patient data. Zaabar et al. (2022) integrates FL with blockchain for secure IoMT data management, while Guembe et al. (2024) combines FL with blockchain technology to address both security and anomaly detection in healthcare environments. Other frameworks, such as those by Kolokotronis et al. (2022), enhance scalability and efficiency in heterogeneous device networks.

Beyond IoMT, one of the most significant applications of FL is in facilitating large-scale, multi-institutional research while adhering to strict patient data privacy regulations like the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA). Health centers, ranging from small clinics to national institutions, can act as participants, collaboratively training a global model without ever sharing raw patient data. By aggregating knowledge in the form of model updates rather than centralizing sensitive information, FL makes it possible to develop powerful diagnostic and prognostic models that would otherwise be unattainable due to privacy constraints and the logistical hurdles of data-sharing agreements (Sheller et al., 2020).

**3. Vehicular Networks and Smart Cities** Vehicular networks and smart cities use FL to secure traffic systems and optimize mobility. Cui et al. (2023) propose FL for intrusion detection in vehicular networks, while Arya et al. (2023) focus on anomaly detection in Vehicular Ad Hoc Networks (VANET). Liu et al. (2021) combine FL and blockchain to enhance security in vehicular edge computing. These solutions improve scalability and privacy in transportation systems.

**4. Energy Systems and Industrial Smart Grids** In energy systems, FL enhances security, detects anomalies, and optimizes resources. Pan et al. (2022) introduce an incentive-based FL approach for energy privacy, while the FeDiSa framework (Husnoo et al., 2023) addresses latency in federated power systems. Abdel-Basset et al. (2022) highlight the role of FL in privacy-preserving anomaly detection in smart grids, showcasing its potential in critical infrastructures.

**5. Software-Defined Networking (SDN) and Cloud Systems** FL in SDN and cloud systems focuses on secure traffic monitoring and anomaly detection. Wang et al. (2022b) use FL with autoencoders for SDN traffic anomaly detection, and Nguyen et al. (2025) integrate FL with deep reinforcement learning for collaborative monitoring.

**Summary** The adoption of privacy-preserving FL across these fields showcases its ability to address specific challenges while ensuring data security and privacy. FL enables effective collaborative learning without compromising sensitive information, making it a valuable tool in cybersecurity, healthcare, transportation, energy, and cloud systems. These applications underscore the growing role of FL in creating secure, scalable, and privacy-sensitive solutions for modern infrastructures.

## 3.3 Defensive Mechanisms and Security Approaches

FL introduces unique security challenges due to its decentralized nature, which relies on collaboration between clients and servers while maintaining data privacy. Unlike traditional centralized systems, FL must address vulnerabilities across multiple layers, including the clients, the aggregation server, and the model update process. These vulnerabilities expose FL systems to risks such as data poisoning, inference attacks, and tampering during communication.

### 3.3.1 Byzantine Fault Tolerance and Adversarial Robustness

The security of clients and aggregation servers in FL is critical for ensuring the integrity and robustness of the entire system. The aggregation server acts as the central coordinator, combining model updates from multiple clients, while the clients handle sensitive local data and contribute to the global model. This interaction creates vulnerabilities such as poisoning attacks, where malicious clients submit corrupted updates, and inference attacks, where adversaries attempt to extract private information from updates.

Lee et al. (2024) introduce a framework for Byzantine-robust FL through dummy contrastive aggregation. This technique uses synthetic dummy gradients to identify and exclude malicious updates during aggregation, enhancing resilience against Byzantine attacks. The approach employs triplet distance scoring to evaluate update quality and outperforms traditional methods in highly adversarial environments. While effective, this approach requires significant computational resources to generate and validate dummy gradients, potentially limiting its scalability in resource-constrained IoT networks.

Yan et al. (2024) propose Federated Robust Layer-wise Aggregation (FedRola), which leverages global model parameters and client updates independently through a unique layer-based aggregation with dynamic layer selection. This approach provides enhanced Byzantine robustness and improved aggregation efficiency through dynamic adaptation to client capabilities. However, the complex layer selection logic introduces computational overhead that may challenge deployment in edge environments.

Fan et al. (2024) explore the use of localized stepwise updates on the client side and element-wise anomaly detection on the server side to detect and exclude malicious or otherwise byzantine updates. This method focuses on granular detection mechanisms but requires careful tuning to balance detection accuracy with computational efficiency.

Ma et al. (2022) propose combining a Model Update-Based (MUB) scheme with the Initialization-Centric Model Initialization (ICMI) scheme to enhance both privacy and security. It proves particularly effective against Byzantine attacks, though the evaluation was limited to only two types: noise attacks and sign-flipping attacks, leaving questions about broader attack resistance.

Xhemrishi et al. (2025) introduce a novel group testing framework (FedGT) to enhance security and privacy in FL systems. FedGT utilizes group testing principles to efficiently identify and mitigate malicious actors while preserving the privacy of individual participants. This approach

balances security with computational efficiency, making it particularly suitable for large-scale deployments, though it requires careful tuning of group sizes to optimize detection rates without introducing significant overhead.

Li et al. (2023) apply game theory to model adversarial behaviors, enabling systems to proactively adjust defense mechanisms against evolving threats. This approach provides adaptive security but requires significant computational resources and may not scale well for large IoT networks.

Zhao et al. (2024) utilize Generative Adversarial Networks (GAN) to detect malicious global model updates and thwart reconstruction attacks. The approach provides effective detection of malicious updates and mitigation of reconstruction attacks while adapting to evolving threat patterns. The GAN-based detection mechanism offers robustness but requires careful training to avoid adversarial manipulation of the detection system itself.

Performance analysis reveals that dummy contrastive aggregation shows superior Byzantine resilience compared to traditional FedAvg, while layer-based aggregation provides better adaptation but a higher computational cost. Group testing offers scalable detection but requires parameter optimization for optimal performance.

The reviewed Byzantine fault tolerance approaches demonstrate varying trade-offs between security guarantees and computational efficiency. While methods like dummy contrastive aggregation provide strong theoretical guarantees, their practical deployment in resource-constrained environments remains challenging. The evolution toward more sophisticated detection mechanisms, such as layer-based and group testing approaches, shows promise but requires further optimization for edge deployment scenarios.

Table 3.2: Gap Analysis for Byzantine Fault Tolerance and Adversarial Robustness

| **Paper** | **Main Approach** | **Handles Noise Attacks** | **Handles Sign-Flipping** | **Lightweight for Edge** | **Scalable Detection** |
|---|---|---|---|---|---|
| Lee et al. (2024) | Dummy contrastive aggregation | ✓ | ✓ | | ✓ |
| Yan et al. (2024) | Layer-based aggregation | | ✓ | | ✓ |
| Fan et al. (2024) | Localized stepwise updates | ✓ | ✓ | ✓ | |
| Ma et al. (2022) | Model update-based scheme | ✓ | ✓ | | |
| Xhemrishi et al. (2025) | Group testing framework | ✓ | ✓ | ✓ | ✓ |
| Li et al. (2023) | Game-theoretic adversarial modeling | ✓ | ✓ | | |
| Zhao et al. (2024) | GAN-based update detection | ✓ | ✓ | | |

### 3.3.2 Privacy-Preserving Techniques and Encryption

In FL, model updates are pivotal as they represent the aggregated learning outcomes from individual clients and shape the performance of the global model. However, the distributed nature of FL introduces vulnerabilities to security threats such as adversarial manipulation, data poisoning, and inference attacks. These risks can compromise the reliability and accuracy of the global model, potentially jeopardizing the entire system.

Guo et al. (2024) combines Hierarchical Federated Graph Learning (HiFGL) framework with multi-layer encryption and real-time anomaly detection. Their approach uses graph-based representations to ensure update integrity and emphasizes cross-silo and cross-device collaboration

while mitigating risks associated with adversarial attacks on model updates. The framework offers enhanced data integrity and scalability, but the framework introduces significant communication overhead during update exchanges.

Bahar et al. (2024) utilizes hybrid encryption for Graph Neural Networks (GNNs), incorporating an enhanced encryption system for the updated weights of the clients. This approach proves particularly effective at preventing Man-in-the-Middle (MitM) attacks from intercepting weights and reconstructing data through reverse engineering. A similar approach could be beneficial for models different than GNNs.

Nguyen & Thai (2024) explores the use of homomorphic encryption for the private combination of local model updates and implements a Zero-Knowledge Proof protocol to shift attack detection from the server to the users. This approach provides strong privacy guarantees but could introduce resource limitation concerns for resource-limited edge devices.

Aljrees et al. (2023) introduces a sustainable encryption method designed specifically for IoT security within FL systems. It combines efficient encryption mechanisms with the Quondam Signature Algorithm (QSA) to provide robust end-to-end security for model updates, achieving communication costs of 192–220 bits. The framework ensures data privacy and integrity while maintaining energy efficiency, critical for IoT environments and battery-powered edge devices.

The FD2K scheme (Wang et al., 2023a) uses FL and multi-agent deep reinforcement learning to generate keys between legitimate users by exploiting common features of physical signal dynamics. This approach achieved considerable performance improvements in terms of key agreement and conciliation, randomness, and uniqueness, and shows promise for use in edge networks.

Li et al. (2025) proposes a framework that introduces update digests, compact representations of model updates, to detect and mitigate malicious contributions without requiring access to raw data, enabling efficient anomaly detection while preserving data privacy. The framework employs a voting-based defense mechanism in which participating nodes collaboratively evaluate and approve updates before aggregation, improving the robustness against poisoning attacks.

Sawada et al. (2024) proposes a method for employing random binary weights to obfuscate updates. By introducing stochasticity, the approach obfuscates gradients, preventing adversaries from reconstructing private data or injecting poisoned updates, being effective against Approximate Reconstruction and Poisoning via Iterative Learning (APRIL) attacks. This lightweight method improves security with minimal computational overhead, making it suitable for resource-constrained environments, and has the potential to be used with larger models.

Jebreel et al. (2024) introduces a Fragmented FL approach that prevents semi-honest third parties from mounting privacy attacks and effectively counters Model Poisoning attacks. It involves the participants randomly exchanging and mixing fragments of their updates before sending them to the aggregation server through the use of a lightweight protocol. It also incorporates a tailored Reputation-based Defense to build trust in participants and their mixed updates. The effects on the network bandwidth of these rounds of model update mixing should be explored, especially if it is to be used with larger models.

Performance analysis indicates that homomorphic encryption provides the strongest privacy guarantees but the highest computational cost, while hybrid encryption balances security and efficiency for specific architectures. Random binary weights offer lightweight protection with minimal overhead, making them suitable for resource-constrained deployments.

Privacy-preserving techniques in FL face the fundamental challenge of balancing protection strength with computational efficiency. While approaches like homomorphic encryption provide theoretical privacy guarantees, their practical deployment in edge environments remains limited by computational constraints. The trend toward lightweight, application-specific solutions shows promise for real-world deployment.

Table 3.3: Gap Analysis for Privacy-Preserving Techniques and Encryption

| Paper | Main Approach | Homomorphic Encryption | Zero-Knowledge Proofs | Lightweight for Edge | LLM Compatible |
|---|---|---|---|---|---|
| Guo et al. (2024) | Multi-layer encryption with graph learning | | | | |
| Bahar et al. (2024) | Hybrid encryption for GNNs | ✓ | | | |
| Nguyen & Thai (2024) | Homomorphic encryption with ZK proofs | ✓ | ✓ | | |
| Aljrees et al. (2023) | Sustainable encryption for IoT | ✓ | | ✓ | ✓ |
| Wang et al. (2023a) | FD2K scheme for key generation | | | ✓ | |
| Sawada et al. (2024) | Random binary weight obfuscation | | | ✓ | ✓ |
| Li et al. (2025) | Update digests with voting defense | | | ✓ | ✓ |
| Jebreel et al. (2024) | Fragmented FL with reputation defense | | | ✓ | ✓ |

### 3.3.3 Blockchain Integration and Trust Management

Blockchain technology offers immutable ledgers and decentralized trust mechanisms that address fundamental security challenges in FL. By providing tamper-proof record-keeping and automated validation through smart contracts, blockchain integration enhances transparency and accountability in FL systems.

Houda et al. (2023) introduces Multi-party computation in Trust-based Federated learning (MiTFeD), which integrates blockchain technology into FL to enhance security and transparency. MiTFeD leverages the immutable ledger of the blockchain to securely record model updates and facilitate tamper-proof model aggregation, using Ethereum smart contracts and Secure Multiparty Computation (SMPC) for secure aggregation. The approach ensures robust defense against adversarial manipulations but introduces latency and storage overhead that may limit scalability in resource-constrained environments.

Cui & Mei (2023) integrates FL with blockchain technology, using historical data for model update prediction and malicious node identification through consistency verification. The framework introduces an agent consensus mechanism to reduce model cross-validation costs and increase efficiency, though the added computational demands from blockchain operations may limit scalability in dynamic environments.

Xu et al. (2022) proposes an FL framework with authentication and verification using blockchain technology to protect data privacy and prevent poison attacks. Their method involves dynamic identity management, where only updates uploaded by authenticated local devices can be added to the blockchain, and local devices can verify the received global model.

Liu et al. (2022) proposes a comprehensive three-layer security model to defend against external and internal attacks. The first layer uses blockchain technology to authenticate trusted users, the second layer utilizes Trusted Execution Environment (TEE) technology for privacy protection, and the final layer adopts detection and location algorithms to defend against internal attackers.

Zaabar et al. (2022) introduces a secure framework that combines blockchain and FL for IoMT applications. The blockchain ensures immutable and verifiable model updates, while smart contracts automatically validate contributions. The system employs an optimized consensus mechanism to reduce resource usage while maintaining privacy and security.

Moudoud & Cherkaoui (2022) integrates blockchain with FL to enhance security and reduce communication in IoT systems. An Oracle blockchain ensures data reliability and privacy, while a resource allocation algorithm prioritizes devices with reliable behavior, minimizing communication. While effective, added computational demands from blockchain may limit scalability.

Performance analysis reveals that smart contract validation provides automation but introduces latency, while multi-layer approaches offer comprehensive security but increase complexity. Optimized consensus mechanisms reduce overhead but may compromise some security guarantees.

Blockchain integration in FL represents a significant advancement in trust management and transparency. However, the inherent trade-offs between security guarantees and system performance remain challenging. The evolution toward optimized consensus mechanisms and multi-layer architectures shows promise for practical deployment, though scalability concerns persist.

Table 3.4: Gap Analysis for Blockchain Integration and Trust Management

| Paper | Main Approach | Smart Contracts | Consensus Optimization | Multi-layer Security | Edge Compatible |
|---|---|---|---|---|---|
| Houda et al. (2023) | Blockchain with smart contracts | ✓ | | | |
| Cui & Mei (2023) | Blockchain with agent consensus | ✓ | ✓ | | |
| Xu et al. (2022) | Dynamic identity management blockchain | ✓ | | | ✓ |
| Liu et al. (2022) | Three-layer blockchain security | | | ✓ | |
| Zaabar et al. (2022) | Optimized consensus for IoMT | ✓ | ✓ | | ✓ |
| Moudoud & Cherkaoui (2022) | Oracle blockchain for IoT | ✓ | ✓ | | ✓ |

## 3.4 Advanced Learning Paradigms and Model Architectures

The evolution of ML architectures in federated settings has led to sophisticated approaches that address the unique challenges of distributed learning. These paradigms focus on leveraging generative models for data augmentation and implementing adaptive systems that can quickly respond to changing threat landscapes.

### 3.4.1 Generative Models and Data Augmentation

Generative models address critical challenges in FL, particularly data imbalance and security vulnerabilities. By generating synthetic data and detecting adversarial behavior, these approaches enhance both the robustness and effectiveness of distributed learning systems.

Tabassum et al. (2022) introduces a Federated GAN-based IDS (FEDGAN-IDS), combining GANs and FL for IoT intrusion detection, addressing data imbalance in federated settings and ensuring privacy preservation. The approach achieves 99% binary and 98% multi-class accuracy, outperforming traditional IDS methods across multiple datasets. However, GAN training complexity and computational overhead present challenges for resource-constrained environments.

Zhao et al. (2024) utilizes GANs to detect malicious global model updates and thwart reconstruction attacks. The approach provides effective detection of malicious updates and mitigation of reconstruction attacks while adapting to evolving threat patterns. The GAN-based detection mechanism offers robustness but requires careful training to avoid adversarial manipulation of the detection system itself.

Abdel-Basset et al. (2023) utilizes a semi-supervised generative network with a focus on rebalancing classes to improve the quality of generated minority (anomaly) samples. This approach makes the model more robust in imbalanced data scenarios, which is particularly relevant for cybersecurity applications where anomalous events are typically rare.

Chelli et al. (2023) introduces FEDGUARD, which employs Conditional Variational Autoencoders (CVAEs) for selective parameter aggregation, enhancing resilience against poisoning attacks. The framework operates without auxiliary datasets, making it adaptable, but may be potentially limited by synthetic data quality in heterogeneous settings.

Performance comparison shows that FEDGAN-IDS achieves the highest accuracy but requires substantial computational resources, while GAN-based detection provides adaptive security but faces training complexity. CVAE approaches offer flexibility but depend on synthetic data quality.

Generative models in FL have shown remarkable potential for addressing data imbalance and security challenges. However, the computational complexity of training generative models in distributed settings remains a significant barrier. The trend toward more efficient generative architectures and specialized applications shows promise for practical deployment.

Table 3.5: Gap Analysis for Generative Models and Data Augmentation

| Paper | Main Approach | Handles Data Imbalance | Synthetic Data Quality | Lightweight Training | Edge Compatible |
|---|---|---|---|---|---|
| Tabassum et al. (2022) | GANs with FL for IoT intrusion detection | ✓ | ✓ | | ✓ |
| Zhao et al. (2024) | GANs for malicious update detection | ✓ | ✓ | | |
| Abdel-Basset et al. (2023) | Semi-supervised generative networks | ✓ | ✓ | | |
| Chelli et al. (2023) | CVAE for selective aggregation | | ✓ | ✓ | |

### 3.4.2 Meta-Learning and Adaptive Classification Systems

Meta-learning approaches enable FL systems to quickly adapt to new threats and changing environments. These systems leverage prior learning experiences to improve performance on new tasks with minimal additional training, making them particularly valuable for cybersecurity applications where threat landscapes evolve rapidly.

Zukaib et al. (2024) proposes Meta-Fed IDS, combining meta-learning and FL to detect known and zero-day cyberattacks in IoMT environments. Meta-learning enables swift adaptation to new

threats, while FL preserves data privacy. Designed for fog-cloud environments, it improves accuracy and response time but requires high computational resources for meta-learning algorithms.

Jin et al. (2023) introduces an Evolvable Intrusion Detection System (EIDS) that tackles zero-day attacks using federated incremental learning. The system achieves 86% accuracy and a 90% reduction in update time but depends on careful parameter tuning for diverse IoT networks, limiting scalability across heterogeneous environments.

Tan et al. (2025) presents BackdoorIndicator, leveraging out-of-distribution data to detect malicious updates in FL. It achieves high detection rates with minimal task accuracy impact and offers a scalable solution for backdoor attacks but could benefit from additional refinement and integration with other defense mechanisms.

Guembe et al. (2024) introduces a Federated Bayesian Optimization XGBoost model for IoMT cyberattack detection. By combining FL and Bayesian optimization, the model achieves high accuracy while preserving data privacy, outperforming traditional methods in addressing evolving threats in healthcare environments. However, computational complexity and parameter tuning requirements present implementation challenges.

Performance analysis shows that meta-learning approaches provide the fastest adaptation but the highest computational cost, while incremental learning offers a good balance of adaptation and efficiency. Additionally, Out-of-distribution detection provides specialized backdoor protection with minimal overhead.

Meta-learning and adaptive systems represent the cutting edge of FL evolution, enabling systems to quickly adapt to new threats and changing environments. However, the computational requirements often exceed the capabilities of edge devices, creating a tension between adaptability and practical deployment.

Table 3.6: Gap Analysis for Meta-Learning and Adaptive Systems

| **Paper** | **Main Approach** | **Fast Adaptation** | **Zero-Day Detection** | **Lightweight for Edge** | **Scalable Training** |
|---|---|---|---|---|---|
| Zukaib et al. (2024) | Meta-learning with FL for IoMT | ✓ | ✓ | | |
| Jin et al. (2023) | Incremental learning for evolving threats | ✓ | ✓ | ✓ | |
| Tan et al. (2025) | OOD detection for backdoor attacks | ✓ | | ✓ | ✓ |
| Guembe et al. (2024) | Bayesian optimization for IoMT | ✓ | | | |

## 3.5 System Optimization and Resource Management

Efficient resource utilization and system optimization are crucial for the practical deployment of FL systems, particularly in resource-constrained environments. These approaches focus on reducing communication overhead, managing heterogeneous data distributions, and optimizing aggregation mechanisms to ensure scalable and efficient operation.

### 3.5.1 Communication Efficiency and Latency Reduction

Latency and communication overhead are critical challenges in FL, particularly in resource-constrained environments like IoT and edge computing. Efficient communication protocols and optimization techniques are necessary to address the high costs associated with data transmission, model synchronization, and the inherent delays in decentralized systems.

Houda et al. (2022) introduces FogFed, a low-latency fog-based framework to secure IoT applications using collaborative FL. The approach combines fog computing and FL to reduce communication delays, employing binary FL classifiers at the fog layer and multiclass classifiers in the cloud. FogFed achieves 99% accuracy and detection rates with significant latency reduction, though its dependence on fog infrastructure scalability could hinder application deployment.

Husnoo et al. (2023) proposes an FL-based Distributed IDS for power system Anomalies (FeDiSa), a semi-asynchronous FL framework for power system fault and cyberattack discrimination. It employs a buffer system and preset cut-off time for model aggregation to reduce delays and stragglers, achieving a 35% reduction in training time while maintaining 92.4% detection accuracy. The complexity of the approach may limit scalability in dynamic environments.

Aouedi & Piamrat (2024) introduces Sustainable Intrusion Detection with Hierarchical Federated Spiking Neural Networks (SURFS), integrating HFL and Spiking Neural Networks (SNNs) to reduce communication costs and energy usage in the Cloud-Edge-IoT continuum. Its decentralized HFL structure minimizes data transmission, achieving a 99% reduction in communication overhead, while SNNs ensure computational efficiency. It suits resource-constrained environments but depends heavily on specialized neural architectures, limiting broader application.

Stevens et al. (2022) presents FL with Differential Privacy (FLDP), combining differential privacy with secure aggregation to reduce communication overhead. By using Learning With Errors (LWE) for masking and Renyi Differential Privacy for guarantees, it achieves scalable, fast model updates while maintaining robust privacy, though the computational intensity may challenge deployment in resource-limited settings.

Abdulrahman et al. (2021) proposes Federated Multi-Criteria Client Selection (FedMCCS), optimizing client selection in IoT FL by evaluating Central Processing Unit (CPU), memory, energy, and time resources. This approach reduces communication rounds and minimizes discarded rounds while optimizing network traffic. Performance depends on accurate resource estimation, which could be limited in dynamic environments.

The performance analysis indicates that FogFed achieves the best latency reduction but requires infrastructure investment, while FeDiSa provides balanced performance with a 35% improvement in training time. SURFS offers the most significant communication reduction at 99% overhead reduction.

Communication efficiency approaches demonstrate substantial improvements in FL performance, with reductions ranging from 35% to 99% across different metrics. However, these improvements often come with trade-offs in terms of infrastructure requirements, computational

complexity, or security guarantees. The evolution toward hierarchical and fog-based architectures shows promise for practical deployment scenarios.

Table 3.7: Gap Analysis for Communication Efficiency and Latency Reduction

| Paper | Main Approach | Reduces Latency | Handles Stragglers | Infrastructure Independent | Scalable for LLMs |
|---|---|---|---|---|---|
| Houda et al. (2022) | Fog-based FL framework | ✓ | | | ✓ |
| Husnoo et al. (2023) | Semi-asynchronous FL with buffers | ✓ | ✓ | ✓ | |
| Aouedi & Piamrat (2024) | HFL with spiking neural networks | ✓ | | ✓ | ✓ |
| Stevens et al. (2022) | LWE-based secure aggregation | ✓ | | ✓ | ✓ |
| Abdulrahman et al. (2021) | Multi-criteria client selection | ✓ | | ✓ | ✓ |

### 3.5.2 Model Aggregation and Heterogeneous Data Management

Effective model aggregation is critical in FL, as it ensures collaborative training while preserving privacy. The diverse nature of data in IoT and industrial networks poses challenges to achieving high detection accuracy. Innovative aggregation strategies and techniques for handling heterogeneous data distributions are essential for robust performance across devices and networks.

Thein et al. (2024) introduces a mini-batch logit adjustment loss technique that enhances aggregation by addressing imbalances in data distributions. This approach improves the convergence of global models in scenarios with highly diverse data sources, ensuring consistent and reliable performance. However, it requires additional fine-tuning to handle extreme cases of data imbalance and avoid extreme client drift caused by imbalanced data distributions.

Houda et al. (2023) employs SMPC for secure aggregation, blockchain, and smart contracts for decentralized collaboration. While effective in maintaining privacy, blockchain integration adds latency and storage overhead, limiting its feasibility in resource-constrained environments.

Cui et al. (2023) leverages constrained multi-objective optimization for fairness in model aggregation across clients in software-defined vehicular networks. Optimizing prediction loss through a two-stage gradient method ensures fair contributions in FL while improving attack detection accuracy. The reliance on constrained optimization of the approach may introduce computational overhead in large-scale networks.

Willeke et al. (2023) uses Federated Averaging to aggregate updates, achieving 93% accuracy in malicious payload detection. It emphasizes raw data analysis over feature engineering but may require frequent updates to address evolving threats.

Yin et al. (2023) integrates FL to enhance model aggregation with a cloud-based loss function, achieving 96.5% accuracy while preserving privacy. Although efficient, the method's reliance on feature engineering and regularization complicates deployment in heterogeneous IoT networks.

de Cámara et al. (2023) introduces a clustered FL architecture that improves aggregation by grouping clients with similar updates, addressing non-Independent and Identically Distributed (non-IID) data issues. It accelerates convergence and enhances anomaly detection in IoT environments, though clustering overhead and dynamic client management present challenges.

Wang et al. (2023b) combines a multi-branch model with vertical blocking aggregation (FedVB) to improve detection accuracy and reduce communication overhead. The approach's branch-specific updates enhance privacy and efficiency but may face challenges in highly dynamic or non-IID environments.

Zhang et al. (2024) tackles heterogeneous data in Industrial Internet of Things (IIoT) environments using continual representation learning, dynamically adapting to non-stationary data distributions by extracting generalized features. This approach enhances anomaly detection while optimizing resource use, making it suitable for large-scale applications, though complex adaptation mechanisms introduce computational overhead.

Nguyen et al. (2024) proposes Federated Principal Component Analysis (FedPCA), employing Principal Component Analysis (PCA) with Alternating Directions Method of Multipliers (ADMM) to learn common representations from non-IID datasets. The approach achieves high detection rates with improved communication and memory efficiency on UNSW-NB15 and TON-IoT datasets, though PCA limitations may constrain feature space representation.

Wang et al. (2022a) introduces Semi-supervised Federated Model Distillation (SFMD) for personalized, semi-supervised model training, handling heterogeneous IoT data by tailoring global models to specific local distributions. This enhances detection accuracy while maintaining privacy, though distillation complexity and personalization overhead present implementation challenges.

Chen et al. (2021) propose Zero-Knowledge Clustering (ZeKoC). The method works by clustering client model updates on the server. The rationale is that updates from malicious clients will be statistical outliers, allowing them to be isolated from the clusters formed by benign client updates. The "zero-knowledge" component ensures this clustering is performed in a privacy-preserving manner, enabling the server to group the updates and identify these outliers without accessing the raw weight values themselves. This approach is shown to be effective at mitigating attacks in non-IID settings.

Abosata et al. (2022) presents Federated Transfer Learning for Customised Intrusion Detection (FTL-CID) for domain adaptation in IIoT heterogeneous networks. The approach improves cross-domain performance but requires domain similarity and introduces transfer learning complexity.

Performance analysis shows that continual learning provides the best adaptation but the highest computational cost, while PCA-based approaches offer a good efficiency-accuracy balance. Zero-knowledge methods provide the strongest privacy but complexity overhead.

The reviewed literature showcases diverse approaches to improving model aggregation in FL, focusing on privacy, security, and efficiency. Techniques like FedVB and clustered FL address communication overhead and non-IID data challenges, while frameworks like SFMD and ZeKoC enhance privacy through innovative aggregation methods. Challenges such as computational inefficiencies and trade-offs between privacy and accuracy remain prevalent, highlighting the need for further optimization.

Table 3.8: Gap Analysis for Model Aggregation and Heterogeneous Data Management

| Paper | Main Approach | Handles Non-IID Data | Personalized Training | Communication Efficient | Lightweight for Edge |
|---|---|---|---|---|---|
| Thein et al. (2024) | Mini-batch logit adjustment | ✓ | ✓ | | |
| Houda et al. (2023) | SMPC with blockchain aggregation | | | ✓ | |
| Cui et al. (2023) | Constrained multi-objective optimization | ✓ | | | |
| de Cámara et al. (2023) | Clustered FL for IoT | ✓ | | ✓ | ✓ |
| Wang et al. (2023b) | Multi-branch with vertical blocking | ✓ | | ✓ | |
| Zhang et al. (2024) | Continual representation learning | ✓ | ✓ | | |
| Nguyen et al. (2024) | FedPCA with ADMM | ✓ | | ✓ | ✓ |
| Wang et al. (2022a) | Semi-supervised model distillation | ✓ | ✓ | | |
| Chen et al. (2021) | Zero-knowledge clustering | ✓ | ✓ | | ✓ |
| Abosata et al. (2022) | Federated transfer learning | ✓ | ✓ | | ✓ |

## 3.6 Anomaly Detection Learning Methods

The choice of a ML model and learning paradigm is fundamental to the effectiveness of a federated IDS. While supervised learning offers high accuracy when large labeled datasets are available, such data is a rarity in real-world cybersecurity scenarios. Consequently, research has focused on unsupervised and semi-supervised methods that can learn from unlabeled or partially labeled data, which is far more common in edge environments.

### Unsupervised and Semi-Supervised Learning

Unsupervised learning methods aim to identify anomalies by learning the patterns of normal data without any labels, flagging deviations as potential intrusions. Semi-supervised learning offers a pragmatic middle ground, leveraging a small amount of labeled data to guide the learning process on a much larger pool of unlabeled data.

Aouedi et al. (2023) involves a hybrid model where clients perform unsupervised representation learning locally. In their work, clients train an Autoencoder on their local, unlabeled data to learn a compressed representation of normal behavior. The server then aggregates these models, adds fully-connected layers to the encoder, and performs supervised training on a small, publicly available labeled dataset. This division of labor allows the system to learn from private, unlabeled data at the edge while benefiting from centralized, supervised fine-tuning. The authors identify the use of RL for dynamic participant selection and optimization as a key direction for future research.

Hamidpour & Bushehrian (2023) presents a round-based approach that begins by training on a small, trusted set of benign samples. In subsequent rounds, the dataset is gradually expanded with new traffic samples, which are classified autonomously. This method mimics a continual refinement process, allowing the model to adapt to evolving data patterns and threats over time, which is a form of incremental, semi-supervised learning.

Wang et al. (2022a) proposes SFMD, a framework designed to assist clients that lack labeled data entirely. It enables collaborative training and allows each client to develop a personalized model tailored to its specific data distribution, addressing the challenge of data heterogeneity in IoT environments.

Abdel-Basset et al. (2023) utilizes a semi-supervised generative network with a focus on rebalancing classes to improve the quality of generated minority (anomaly) samples, making the model more robust in imbalanced data scenarios typical in cybersecurity applications.

Bekbulatova et al. (2023) adopts Deep Semi-supervised Anomaly Detection (Deep-SAD), a methodology that maps normal data points into a compact hypersphere in the latent space. Anomalies, being dissimilar, are mapped outside this sphere and are identified based on their distance from the center. This approach effectively uses the structure of the data itself to distinguish between normal and anomalous instances.

Chelli et al. (2023) employs CVAEs at the server to synthesize validation data. This server-side data generation can be viewed as a form of semi-supervised learning, where the server creates its own labeled data to validate client contributions, reducing reliance on pre-labeled datasets.

Performance analysis shows that hybrid autoencoder approaches provide a good balance of privacy and performance, while round-based learning offers adaptive capability with minimal labeled data. Deep-SAD provides effective structural anomaly detection with computational efficiency.

These approaches demonstrate a clear trend toward hybrid and semi-supervised models. They aim to overcome the scarcity of labeled data by combining unsupervised representation learning on clients with some form of supervised guidance, either from a small central dataset or through the structural assumptions of the model.

Table 3.9: Gap Analysis for Anomaly Detection Learning Methods

| Paper | Main Approach | Handles Data Imbalance | Personalized Models | No Labeled Data on Client | Lightweight for Edge | Mentions RL for Future Work |
|---|---|---|---|---|---|---|
| Abdel-Basset et al. (2023) | Semi-supervised generative network | ✓ | | | | |
| Wang et al. (2022a) | Semi-supervised FL for clients without labels | | ✓ | ✓ | | |
| Aouedi et al. (2023) | Client-side unsupervised AE, server-side supervised learning | | | ✓ | ✓ | ✓ |
| Chelli et al. (2023) | Server-side CVAE for synthetic validation data | | | ✓ | | |
| Hamidpour & Bushehrian (2023) | Round-based learning with evolving data | ✓ | | ✓ | ✓ | |
| Bekbulatova et al. (2023) | Deep-SAD for semi-supervised anomaly detection | | | | ✓ | |

## 3.7 Participant Selection

In FL, not all clients are equal. They may vary in data quality, data quantity, resource availability (CPU, memory, network), and reliability. Therefore, intelligently selecting which clients to participate in each training round is a critical research area. An effective participant selection strategy can significantly improve model convergence speed, final model accuracy, system fairness, and resource efficiency.

### Resource-Aware and Quality-Based Selection

Abdulrahman et al. (2021) proposes FedMCCS, a multi-criteria client selection model that considers the CPU, memory, energy, and time resources of the clients to predict their ability to complete a training round. The system groups clients into homogeneous strata based on metadata (like location) and then uses a greedy algorithm to solve a knapsack-like problem, maximizing the number of clients that can participate without exceeding a time threshold. This approach directly tackles the issue of client dropouts due to resource limitations.

Jiang et al. (2024) introduces Lotto, a system designed to provide secure participant selection even when the server itself is adversarial. Lotto uses verifiable randomness, allowing clients to autonomously determine if they should participate, thus preventing a malicious server from cherry-picking compromised clients. It offers both random and informed selection algorithms, ensuring that the proportion of dishonest participants in any round aligns with their base rate in the overall population, significantly enhancing security.

Chelli et al. (2023) focuses on mitigating poisoning attacks by detecting malicious peers. While its primary function is secure aggregation, its selection mechanism is integral. FEDGUARD uses a CVAE on the server to synthesize validation data. It then evaluates incoming client updates against this data and selectively aggregates only those that perform well, effectively deselecting or ignoring the contributions of malicious or underperforming clients.

Neto et al. (2024) implements FL with Score-Based Selection (FedSBS), a scoring system based on a variant of information gain, which considers both local and global loss to assess the contribution of the participants. It uses an epsilon-greedy approach for exploration and exploitation and incorporates a blocking mechanism based on a Boltzmann distribution to prevent the over-selection of certain clients, which also helps in identifying potentially malicious actors. A key innovation in FedSBS is the use of a global momentum term, calculated on the server and distributed to clients, to preserve knowledge from previous rounds and harmonize convergence across the network.

Chen et al. (2023) utilizes a dynamic clipping approach to identify and use a network of trusted users. This approach, however, depends on specific and often intense calculations to determine a reliable clipping threshold, particularly in vehicular network environments.

These methods highlight a trend towards more intelligent, criteria-based selection. Performance analysis shows that multi-criteria selection provides the best resource optimization, while cryptographic selection offers the strongest security guarantees. Score-based approaches, such as FedSBS, balance performance and security effectively using predefined heuristics to evaluate client contributions. While effective, the use of adaptive agents that can learn an optimal policy from experience, rather than relying on fixed heuristics, remains a promising area for research.

Table 3.10: Gap Analysis for Participant Selection Methods

| Paper | Main Approach | Considers Resources | Defends Against Malice | Score-Based Contribution | Handles Adversarial Server | Uses Reinforcement Learning |
|---|---|---|---|---|---|---|
| Abdulrahman et al. (2021) | Multi-criteria selection based on client resources (CPU, memory, etc.) | ✓ | | | | |
| Chelli et al. (2023) | Selective aggregation based on performance against synthetic data | | ✓ | ✓ | | |
| Jiang et al. (2024) | Secure selection with verifiable randomness to counter adversarial servers | | ✓ | | ✓ | |
| Neto et al. (2024) | Information gain-based scoring with epsilon-greedy selection and global momentum | | ✓ | ✓ | | |
| Chen et al. (2023) | Dynamic clipping with trusted user networks | | ✓ | | | |

## 3.8 Summary

The state of the art in FL for intrusion detection demonstrates significant advancements across multiple dimensions, with each contributing to a more robust, secure, and efficient distributed learning paradigm.

Defensive Mechanisms and Security Approaches have evolved from basic encryption to sophisticated multi-layered frameworks. Byzantine fault tolerance approaches show promising results, with dummy contrastive aggregation achieving superior resilience compared to traditional FedAvg, though computational overhead remains a concern. Privacy-preserving techniques demonstrate a clear trade-off between protection strength and computational efficiency, with lightweight solutions like random binary weights offering practical alternatives to intensive homomorphic encryption. Blockchain integration provides strong trust guarantees but faces scalability challenges, with optimized consensus mechanisms showing promise for practical deployment.

Advanced Learning Paradigms reveal the potential of generative models and adaptive systems, though computational requirements often exceed edge device capabilities. Generative approaches like FEDGAN-IDS achieve remarkable accuracy (99% binary, 98% multi-class) but require substantial resources. Meta-learning systems provide the fastest adaptation to new threats but face similar computational constraints. The evolution toward more efficient architectures and specialized applications shows promise for practical deployment in resource-constrained environments.

System Optimization and Resource Management approaches demonstrate substantial improvements in FL performance, with communication overhead reductions ranging from 35% to 99% across different metrics. However, these improvements often come with trade-offs in terms of infrastructure requirements, computational complexity, or security guarantees. The trend toward hierarchical and fog-based architectures shows promise for practical deployment scenarios, though heterogeneous data management remains challenging.

Anomaly Detection Learning Methods show a trend toward hybrid and semi-supervised models that overcome the scarcity of labeled data by combining unsupervised representation learning on clients with supervised guidance. These approaches effectively balance privacy with learning effectiveness, though personalization and generalization trade-offs remain challenging.

Participant Selection has evolved from random selection to sophisticated, multi-criteria optimization strategies. Resource-aware approaches like FedMCCS provide optimal resource utiliza-

tion, while security-focused methods like Lotto offer strong protection against adversarial scenarios. Score-based approaches provide balanced performance-security trade-offs, though RL integration remains an open research area.

In conclusion, while significant strides have been made, several research gaps persist. The literature shows that while heuristic-based, multi-criteria participant selection methods exist, the development of an adaptive agent that *learns* an optimal selection policy by balancing competing objectives through a unified reward signal remains an underexplored area. This dissertation aims to fill this gap by presenting a holistic system that integrates such an adaptive agent into a GAN-based, semi-supervised IDS, creating a more efficient, robust, and fair federated learning system for cybersecurity.

# Chapter 4

# Proposed Solution

This chapter describes our proposed solution for building an adaptive FL system. We start by laying out a conceptual framework that explains what the system is trying to accomplish and how its main pieces fit together. Then we get into the practical details, detailing our analysis of different frameworks, tools, and datasets to justify implementation decisions.

## 4.1 Conceptual Framework

The proposed solution is an FL network designed specifically for the task of network intrusion detection in a diverse and decentralized environment. The fundamental goal is to collaboratively train a robust, global intrusion detection model by leveraging data from a multitude of client devices, without requiring those clients to share their sensitive, raw network traffic data with a central server. The framework is designed to simulate and address the key challenges inherent to deploying such a system in a realistic IoT/edge environment. Our approach centers on a smart participant selection strategy that steers the learning process in the right direction.

### 4.1.1 Simulated Environment: Server and Clients

Our conceptual model uses a simulated setup with a central aggregation server and a diverse group of client devices. These clients represent the diverse landscape of IoT and edge nodes that would be found in a real-world deployment, from resource-constrained sensors to more powerful edge gateways. This heterogeneity is a core aspect of the simulation and manifests in two critical ways:

- **System Heterogeneity:** Clients are modeled with varying computational capabilities, including different processing power (CPU cores), memory, and network bandwidth. This ensures the system must contend with stragglers and devices that may not always be able to complete training rounds quickly, or at all.
- **Statistical Heterogeneity:** The local data held by each client is intentionally non-IID. This mirrors a real-world scenario where each device or local network observes a unique and biased slice of network activity. This effect is particularly pronounced in an IoT context,

where different classes of devices may generate fundamentally different types of network traffic and may rely on distinct communication protocols. An effective global model must therefore be built from these varied and specialized data distributions.

### 4.1.2 Data Composition for a Realistic IDS Scenario

To accurately simulate the challenges of intrusion detection, the data distributed across the clients is composed to mimic the scarcity of labeled data in operational networks. The federated dataset is primarily composed of:

1. **Unlabeled Normal Data:** A significant portion of the data consists of unlabeled, benign network traffic. This represents the day-to-day traffic of a network, which is impractical to label manually in its entirety.

2. **Labeled Anomalous Data:** A small set of labeled malicious traffic samples is distributed among the clients, representing known attack signatures or previously identified intrusions.

3. **Labeled Normal Data:** A small fraction of the normal data is also labeled, providing the model with a clear, trusted baseline for what constitutes benign behavior.

This semi-supervised data composition forces the learning process to rely on identifying patterns of normalcy from a mostly normal pool of unlabeled data.

### 4.1.3 Intelligent Participant Selection Strategy

One of the main components of our proposed solution is a smart, adaptive participant selection strategy, built to handle the challenges created by the heterogeneous and semi-supervised environment. Rather than relying on random or round-robin selection, a central RL agent is responsible for choosing which clients are best suited to participate in each round of federated training. The purpose of the agent is to learn an optimal policy that navigates the complex environment based on the outcomes of previous rounds.

### 4.1.4 System Orchestration Cycle

The proposed framework operates through a series of orchestrated FL rounds, managed by a central server component. The process begins with an initialization phase where the server creates the initial global model and the RL agent. Following initialization, the system enters a cyclical process for each round, as illustrated in Fig. 4.1:

1. **Client Selection:** The orchestration component of the server queries the RL agent, providing it with the current state of the network. The agent, using its learned policy, returns a list of clients chosen to participate in the training phase of the current round.

2. **Model Training:** The orchestrator instructs the *selected* clients to perform training on their local data.

3. **Model Aggregation:** Upon completion, the selected clients return their updated model weights to the server. The orchestrator aggregates these contributions to produce the new, improved global model for the current round.

4. **Global Model Evaluation:** The orchestrator then distributes the parameters of this new global model to *all* available clients in the network. Each client evaluates the model on its local validation data and reports its performance metrics back to the server.

5. **Reinforcement Learning Update:** Finally, the orchestrator gathers the evaluation metrics from all clients and computes a global reward signal that reflects the overall performance change. This reward, along with other contextual data from the round, is fed back into the RL agent, triggering its learning process to update its internal policy for the next round's selection of clients.

### 4.1.5 Multi-Criteria Optimization in an IDS Context

The objective of the RL agent is not to optimize a single metric, but to learn a policy that successfully navigates the complex trade-offs between three, often competing, goals, framed within the specific context of an IoT IDS:

1. **Maximize Intrusion Detection Performance:** The primary goal is to select clients whose data contributions are most likely to improve the effectiveness of the global intrusion detection model at identifying anomalies. The agent must learn to sample strategically from the pool of diverse clients to build a more robust and accurate global detector.

2. **Ensure Fairness and Model Generalization:** In the IDS context, fairness prevents the system from overburdening a few high-capability clients and ensures the global model learns from a wide variety of local network conditions. By promoting broad participation, the agent prevents the model from becoming biased towards the environments of a few clients, thus making it more generalizable and effective against a wider range of attacks.

3. **Promote Resource Efficiency on the Edge:** This goal is an important consideration for a system deployed on resource-constrained IoT or edge devices, many of which may be battery-powered and have limited computational resources. The agent must consider the computational cost of training on each client, learning to achieve its security objectives without unnecessarily draining the resources of the participating nodes.

By framing the selection problem as a multi-criteria optimization task, this conceptual framework aims to create a more robust, efficient, and equitable FL system for intrusion detection.

## 4.2 Framework and Dataset Selection

This section serves to analyze and compare several datasets that were used in the literature and select the one most appropriate to the approach of this dissertation.

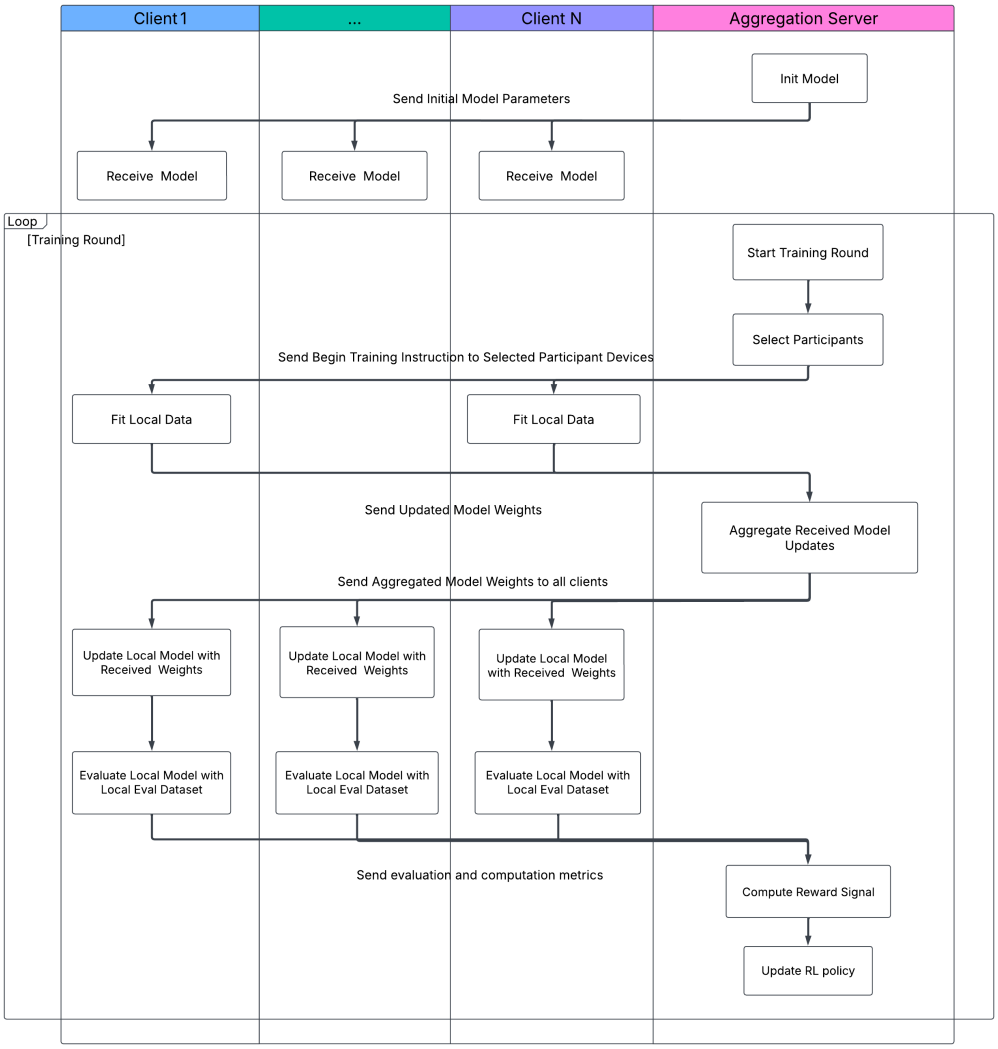



Figure 4.1: The conceptual feedback and learning loop of the system orchestration cycle, detailing the key steps of client selection, model training, aggregation, evaluation, and the RL agent's learning update.

### 4.2.1 Federated Learning Framework Comparison

This section examines various FL frameworks to identify the most appropriate option for our research objectives. The selection criteria focus on the framework's ability to support the implementation and testing of novel participant selection strategies. Our evaluation examines three key aspects: the ease of strategy customization, the quality of simulation capabilities, and the overall flexibility offered by each framework.

#### 4.2.1.1 TensorFlow Federated

TensorFlow Federated (TFF)[1] is an open-source framework for ML and other computations on decentralized data (Thapa et al., 2021). As a part of the TensorFlow ecosystem, it is powerful and well-supported for research. However, its core programming model, which relies on abstract type specifications and ‘tff.Computation‘ blocks, introduces a significant learning curve. While robust for defining novel federated algorithms from first principles, this structure can be less intuitive for implementing stateful strategies that require frequent and simple interaction with an environment's state, making it less agile for the rapid prototyping of selection mechanisms.

#### 4.2.1.2 NVIDIA FLARE

NVIDIA FLARE (NVIDIA Federated Learning Application Runtime Environment)[2] is a production-focused, enterprise-grade framework, originally designed for privacy-preserving AI in healthcare (Kaissis et al., 2020). It is built for scalability and security in real-world deployments, featuring a robust controller-worker architecture. While powerful, its focus on production workflows and security in regulated domains makes it less flexible for academic research that requires rapid, iterative changes to the core learning loop, such as experimenting with participant selection strategies.

#### 4.2.1.3 Flower

Flower[3] is an FL framework that uniquely prioritizes ease of use, flexibility, and extensibility, making it exceptionally well-suited for research (Beutel et al., 2022). Its design philosophy abstracts away the complexities of federated communication, presenting a simple yet powerful `Strategy` API as the primary point of interaction. This allows researchers to easily implement custom logic for client selection, configuration, and aggregation without needing to understand the internal implementation details of the framework.

Its key advantages for this dissertation are:

- **Beginner-Friendliness and Simplicity:** Flower is designed to be easy to learn. A basic federated system can run in a few lines of code, lowering the barrier to entry without sacrificing advanced capabilities.
- **Extensibility:** The modular `Strategy` API is the cornerstone of its flexibility, providing clear and simple hooks for implementing complex, stateful mechanisms like the selection agent at the core of this work.
- **Framework Agnostic:** It seamlessly integrates with multiple ML libraries, including PyTorch, TensorFlow, and Scikit-learn, allowing researchers to use the tools they are most familiar with.

[1] https://www.tensorflow.org/federated
[2] https://developer.nvidia.com/flare
[3] https://flower.ai/

- **Strong Community and Documentation:** Flower is supported by a growing and active community and offers vast documentation, numerous tutorials, and examples that cover a wide range of use cases, from basic to advanced.

This combination of simplicity and power makes Flower the ideal choice, enabling a focus on the research problem of participant selection rather than on boilerplate framework code.

#### 4.2.1.4 PySyft

PySyft[4] is built around the concept of a "Datasite". In this model, a data owner deploys a server, the Datasite, which holds and manages access to their private data. Instead of data being sent to external parties, data scientists submit their data processing code and queries to the Datasite. The Datasite then executes this code on the private data within its secure environment, subject to the data owner's policies, and returns only the permissible, non-sensitive results. While this architecture provides granular control for the data owner and can be used to simulate FL, its fundamental model is built around a centralized data custodian executing remote code. This differs from the dissertation's focus on a system of distributed clients and the dynamics of selecting them, a task for which the direct and flexible Strategy API of a dedicated FL framework is better suited.

#### 4.2.1.5 IBM Federated Learning

IBM Federated Learning (IBM FL)[5], described by Kaggie (2024), is another enterprise-grade solution designed for secure multi-party collaboration in production environments. It provides strong guarantees for security and data governance, making it suitable for regulated industries. However, similar to NVIDIA FLARE, its architecture is tailored for deployment rather than experimentation, making it less adaptable for academic research that requires the flexible and rapid customization of selection algorithms, and the IBM Cloud exclusivity makes it unsuitable for this work.

#### 4.2.1.6 Summary

Table 4.1: Comparison of FL Frameworks

| Framework | Ease of Setup | Strategy Customization | Flexibility |
|---|---|---|---|
| TFF | Moderate | Moderate | TensorFlow-based |
| NVFlare | Complex | Low | Optimized for NVIDIA |
| Flower | Easy | Very High | Multi-framework |
| PySyft | Complex | Moderate | Privacy-Focused |
| IBM FL | Moderate | Low | Limited Academic Flexibility |

Table 4.1 summarizes the comparison of the analyzed FL frameworks. Based on the need for a framework that provides maximum flexibility for implementing and testing a novel, stateful participant selection strategy, Flower emerges as the most suitable choice. Its clear and highly

[4] https://www.openmined.org/pysyft
[5] https://www.ibm.com/docs/en/watsonx/saas?topic=models-federated-learning

extensible `Strategy` API provides the ideal foundation for building the RL mechanism at the core of this work, distinguishing it from other frameworks whose primary focus lies elsewhere, be it production deployment, privacy, or a more rigid algorithmic structure.

### 4.2.2 Simulation Environment

To operationalize the conceptual framework, a realistic simulation environment is required. For this work, Docker[6] and Docker Compose[7] are used to create a virtual network of FL participants. This choice is strongly supported by the Flower framework itself, which provides dedicated tooling for Docker-based simulations. Flower offers base images for its core components, including the server and clients, as well as specialized components like `SuperNode` and `SuperLink` that manage network communication within the simulation.

By encapsulating the server and each client within their own Docker containers, this approach ensures a clean and reproducible environment for each component. Furthermore, it allows for the effective simulation of system heterogeneity by programmatically assigning different resource constraints (e.g., CPU core limits) to each client container via the Docker Compose configuration. This setup, facilitated by Flower's native support for Docker, allows for the creation of a realistic sandboxed environment that mimics a real-world deployment of diverse edge devices without requiring physical hardware.

### 4.2.3 Intrusion Detection Dataset Analysis

This section serves to analyze and compare several datasets that were used in the literature and select the one most appropriate to the approach of this dissertation.

#### 4.2.3.1 NSL-KDD

Tavallaee et al. (2009) analyzed NSL-KDD[8] as an improved version of the outdated KDD'99 dataset. It addresses class imbalance and redundancy issues, making it a frequently used benchmark dataset in FL research. However, its attack diversity is limited, making it less suitable for testing intrusion detection models against modern cyber threats.

#### 4.2.3.2 CICIDS2017

Sharafaldin et al. (2018) examined CICIDS2017[9] as a real-world network intrusion dataset containing multiple modern attack types. Due to its large size and complexity, preprocessing is required to extract meaningful features for FL applications. CICIDS2017 is highly relevant for evaluating FL-based IDS models, particularly for attack classification tasks.

[6] https://www.docker.com/
[7] https://docs.docker.com/compose/
[8] https://www.unb.ca/cic/datasets/nsl.html
[9] https://www.unb.ca/cic/datasets/ids-2017.html

#### 4.2.3.3 IoT-23

Garcia et al. (2020) introduced IoT-23[10] as a dataset specifically designed for intrusion detection in IoT environments. It provides labeled network traffic data collected from IoT devices, making it suitable for FL experiments targeting edge security. However, the relatively small dataset size of IoT-23 may necessitate augmentation techniques to improve generalization.

#### 4.2.3.4 TON-IoT

Alsaedi et al. (2020) presented TON-IoT[11] as a dataset containing telemetry and network traffic data from IoT environments. It includes a variety of modern attack types and captures heterogeneous data sources, making it ideal for evaluating FL-based security models. However, its complexity and large feature set increase the computational burden during preprocessing.

#### 4.2.3.5 UNSW-NB15

Moustafa & Slay (2015) developed UNSW-NB15[12] to include realistic network traffic scenarios and modern attack methodologies. The dataset is diverse and widely used for evaluating advanced intrusion detection models. However, preprocessing and feature engineering are required to extract relevant insights for FL applications.

#### 4.2.3.6 Summary

Table 4.2: Comparison of Intrusion Detection Datasets

| Dataset | Realism | Size | Attack Diversity | Relevance to Modern Threats |
|---|---|---|---|---|
| NSL-KDD | Low | Small | Limited | Low |
| CICIDS2017 | High | Large | Extensive | High |
| IoT-23 | Medium | Small | IoT-focused | High |
| TON-IoT | High | Large | Extensive | Very High |
| UNSW-NB15 | High | Medium | Comprehensive | High |

Table 4.2 highlights the main differences between the analyzed datasets. Based on this analysis, while datasets like IoT-23 and CICIDS2017 are strong candidates due to their focus on edge security and general intrusion detection scenarios, the TON-IoT dataset was ultimately chosen for the approach of this dissertation. Its extensive coverage of modern and realistic IoT attack types, combined with a large volume of telemetry data, provides a good foundation for training and evaluating a federated IDS. Nevertheless, IoT-23, CICIDS2017, and UNSW-NB15 remain highly relevant and would serve as excellent alternatives for future validation and comparative studies.

[10] https://www.stratosphereips.org/datasets-iot23
[11] https://research.unsw.edu.au/projects/toniot-datasets
[12] https://research.unsw.edu.au/projects/unsw-nb15-dataset/

## 4.3 Summary

This chapter has established the complete blueprint for the proposed solution, moving from a high-level conceptual design to the selection of specific, practical tools. It began by outlining a conceptual framework for an adaptive federated IDS, centered on an intelligent participant selection strategy designed to optimize performance, fairness, and efficiency. Following this, a systematic analysis of available technologies was conducted to choose the most appropriate components for realizing this vision.

Based on this evaluation, Flower was selected as the core FL framework for its flexibility and its extensible `Strategy` API, which proved ideal for implementing the custom RL agent. The simulation environment was constructed using Docker and Docker Compose, leveraging their capability to effectively model a network of heterogeneous clients. Finally, the ToN-IoT dataset was chosen for its richness and realism, providing a relevant data source for training and evaluating an IDS in a modern IoT context. Together, these carefully selected components provided the foundation upon which the implemented solution, detailed in the next chapter, was built.

# Chapter 5

# Implemented Solution: RAMPART-FL

This chapter presents the complete implementation of the RAMPART-FL system. It is divided into two main parts. The first part details the foundational components of the system, including its architecture, the data processing pipeline, the non-IID data distribution methodology, and the design of the GAN-based ML model used for anomaly detection. The second part delves into the core innovation: the design and implementation of the RL agent responsible for intelligent participant selection. By detailing these components, this chapter provides a blueprint of the implemented solution, bridging the gap between the high-level proposal and the empirical results.

## 5.1 System Architecture

The system follows a standard client-server architecture for FL, orchestrated by the Flower framework. Flower was chosen for its robust simulation capabilities and modular design, which allows for custom strategies and client/server logic. The entire simulation environment is built using Docker and Docker Compose to simulate a network of clients and a central server. This environment consists of multiple client containers, each configured with different CPU core capacities to simulate a network of heterogeneous edge devices. Communication between clients and the server is handled via gRPC, managed by Flower's SuperLink/SuperNode components for message routing in a simulated environment.

## 5.2 Data, Preprocessing and Non-IID Distribution

This section details the key decisions surrounding the data and the rationale behind them, to create a simulated environment that is representative of real-world intrusion detection in edge scenarios.

### 5.2.1 Rationale for a Semi-Supervised Approach

The choice of a semi-supervised learning paradigm is motivated by the practicalities of deploying an IDS in a real-world edge environment. In operational networks, labeled examples of malicious

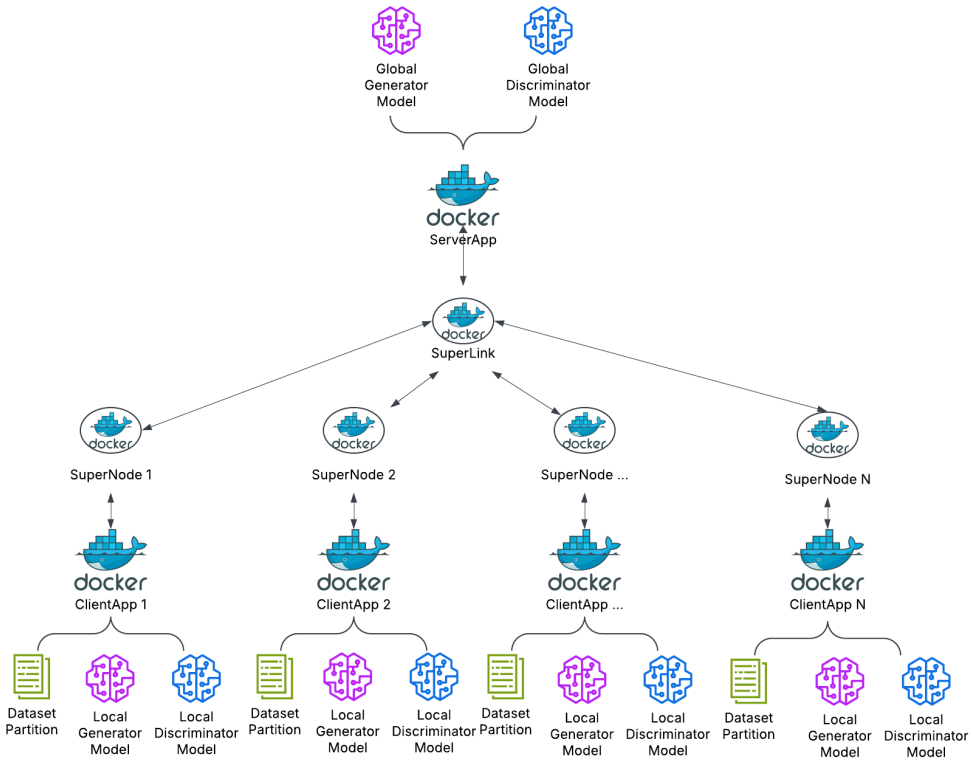



Figure 5.1: High-level system architecture of RAMPART-FL.

traffic are scarce, and manually labeling large amounts of benign traffic is impractical. Semi-supervised learning offers a solution by harnessing a large pool of unlabeled data to build a model of normalcy while using a small set of labeled samples to ground the model's understanding. This allows the model to distinguish between benign novelties and genuine threats with greater accuracy, mirroring a realistic operational scenario. This approach is consistent with recent trends in federated intrusion detection that leverage semi-supervised techniques to address the scarcity of labeled data (Aouedi et al., 2023; Wang et al., 2017). The data partitioning scheme is therefore designed to simulate this challenge.

### 5.2.2 Dataset and Preprocessing

The system utilizes the ToN-IoT dataset, a comprehensive dataset for network intrusion detection in IoT and IIoT environments. The raw data undergoes a multi-stage preprocessing pipeline before being used for federated training. First, a subset of features from the dataset are selected for model training, with the `label` column indicating normal (0) or anomalous (1) traffic. These features capture a range of network traffic characteristics, including basic connection metadata, data transfer volumes, and details from specific protocols like Domain Name Service (DNS) and Hypertext Transfer Protocol (HTTP). The data is then cleaned by replacing hyphen ('-') characters with 0, converting columns to a numeric type, and filling any resulting NaN values in both feature

and label columns with 0. Finally, all features are scaled to a [0, 1] range using `MinMaxScaler` to ensure that all features contribute equally to model training.

### 5.2.3 Initial Dataset Splitting

The raw ToN-IoT dataset is first processed to create a deliberately imbalanced dataset that reflects real-world network conditions.

1. **Inducing Imbalance:** A key characteristic of real-world network traffic is significant class imbalance; benign instances are abundant, while malicious instances are rare. Furthermore, the labeled anomalous data available for training an IDS is often scarce, typically sourced from specific red-teaming exercises or public benchmarks rather than from the vast stream of operational data. To accurately reflect this scenario, the anomalous instances from the original ToN-IoT dataset were deliberately down-sampled, retaining only one-third of the instances for each unique attack type. This ensured the final training dataset was heavily skewed toward normal data. While alternative methods for creating imbalance exist, such as distributing the full set of anomalies across a larger pool of clients or augmenting the dataset with additional normal traffic, these were not pursued. The decision to down-sample was a pragmatic one, dictated by the constraints of using a finite source dataset and the computational limitations on the number of clients that could be simulated in the experimental environment.

2. **Creation of a Labeled Pool:** From this new imbalanced dataset, a small fraction of the "normal" traffic samples, along with all the selected "anomalous" traffic samples, are separated into a "labeled pool." In a real-world application, this labeled data could be sourced from internal red-teaming exercises, publicly available intrusion detection datasets, or traffic samples manually classified by domain experts.

3. **Creation of Test and Validation Sets:** This labeled pool is then split to create per-client test and validation sets, used in the fit and validation rounds, respectively.

4. **Creation of the Unlabeled Client Pool:** The vast majority of "normal" traffic samples, which were not part of the labeled pool, form a large, unlabeled pool of data. This pool, along with a small fraction of the labeled data, is distributed among the clients.

### 5.2.4 Non-IID Data Distribution to Clients

To simulate a heterogeneous edge environment, the combined client data (the large unlabeled pool and a fraction of the labeled pool) is distributed among clients using a non-IID strategy. A Dirichlet distribution, controlled by the `DIRICHLET_ALPHA` hyperparameter (set to 0.3 for this work), is used to partition the data, ensuring that no data samples are shared between clients. A low alpha value results in high data heterogeneity, with clients receiving very different data distributions, while a high alpha value leads to a more IID-like distribution. To prevent clients

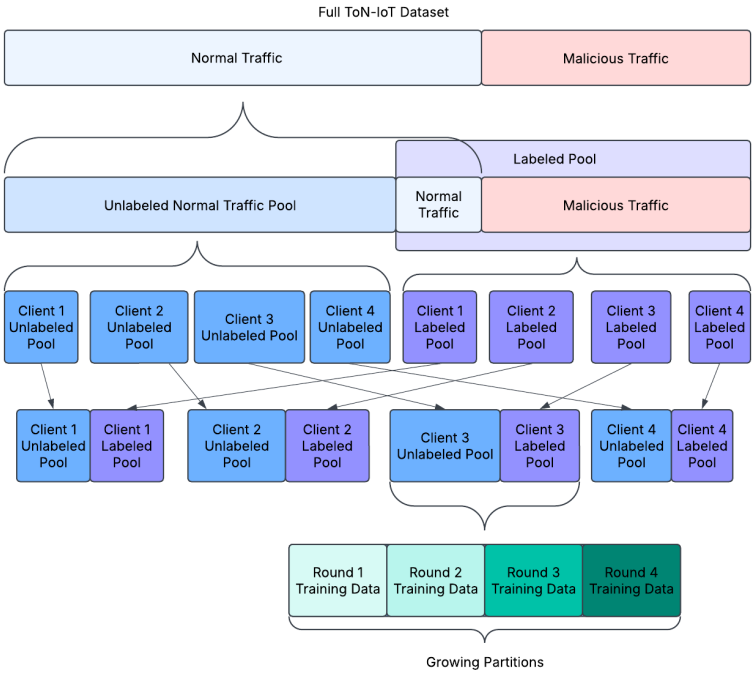



Figure 5.2: Data splitting and distribution diagram (Not to scale).

from being starved of data, especially in such a highly heterogeneous configuration, a minimum of 100 samples is guaranteed for the training and local validation sets of each client. This is a crucial step to ensure that all clients have sufficient data to participate meaningfully in both training and evaluation throughout the simulation. While this minimum guarantee is not perfectly reflective of a real-world scenario, it was a necessary constraint to support a simulation with a significant number of participants under a highly skewed distribution, given the finite size of the dataset. The remaining samples are then distributed according to the Dirichlet distribution.

Figure 5.2 shows a high-level view of the entire data splitting and distribution process.

## 5.3 Machine Learning Model: GAN for Anomaly Detection

The core of the IDS is a GAN, an architecture chosen for its proven effectiveness in semi-supervised anomaly detection. This approach is well-suited to the IDS domain, where labeled anomalous data is typically scarce (Tabassum et al., 2022). The GAN framework consists of two neural networks, a Generator (G) and a Discriminator (D), which are trained in opposition to one another. The role of the Generator is to learn the underlying distribution of normal data and produce synthetic samples, while the role of the Discriminator is to distinguish these synthetic samples from real data. This adversarial competition forces the Discriminator to develop a highly refined understanding of what constitutes normal data, making it an effective tool for identifying any samples that deviate from this learned norm as anomalous.

Figure 5.3 illustrates the typical structure and architecture of a GAN.

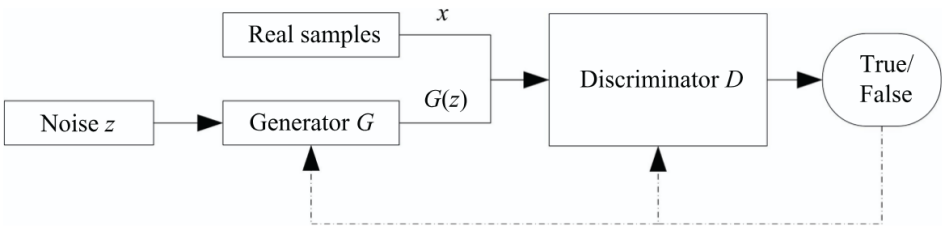



Figure 5.3: Computation procedure and structure of GAN. (Wang et al., 2017)

### 5.3.1 Client-Side Semi-Supervised GAN Training

The semi-supervised training of the GAN is implemented on the client side during each round's `fit` operation. First, clients take an incrementally larger slice of their total assigned data pool for the current round, simulating a real-world scenario where more data becomes available over time. From this slice, they construct a training batch that leverages all three available types of data:

1. **Real Normal Data:** These are samples known to be benign. The Discriminator is trained to assign a high probability (close to 1) to these samples.
2. **Real Anomalous Data:** These are the few labeled malicious samples available. The Discriminator is trained to assign a low probability (close to 0) to these, explicitly teaching it what an anomaly looks like.
3. **Fake Data:** These samples are produced by the Generator. The Discriminator is trained to assign a low probability (close to 0) to these as well.

This training dynamic creates a powerful feedback loop. The Generator, which never sees anomalous data, constantly strives to create more realistic "normal" data to fool the Discriminator. This, in turn, forces the Discriminator to become increasingly adept at identifying the subtle characteristics that define the boundary between normal and not normal (either anomalous or fake). The advantage of using the fake data is that it provides a rich source of "negative" examples that are very close to the decision boundary, compelling the Discriminator to learn a much more precise model of normalcy than if it were only trained on real examples.

### 5.3.2 Model Architecture

Both the Generator and Discriminator are implemented as multi-layer perceptrons (MLP) with fully-connected layers.

- **Generator Architecture:**

  - **Input:** A latent vector $z$ of dimension 100.
  - **Hidden Layers:** Two fully-connected hidden layers with LeakyReLU activation functions.
  - **Output Layer:** A final fully-connected layer with a Sigmoid activation function to output a vector of size 17, matching the number of input features.

- **Discriminator Architecture:**
  - **Input:** A data sample $x$ with 17 features.
  - **Hidden Layers:** Two fully-connected hidden layers with LeakyReLU activation functions.
  - **Output Layer:** A final fully-connected layer with a Sigmoid activation function, producing a single scalar value representing the probability that the input is real normal data.

### 5.3.3 Training and Evaluation Process

- **Training:** The GAN is trained using separate Adam optimizers for the Generator and Discriminator. The loss function is Binary Cross-Entropy (BCE). During the semi-supervised training, the Discriminator learns to output a high probability for real normal samples and a low probability for both real anomalous samples and fake samples from the Generator. The Generator, in turn, is trained to produce fake samples that fool the Discriminator into assigning them a high probability.
- **Evaluation:** For anomaly detection, only the trained **Discriminator** is used. The anomaly score for a given data sample $x$ is calculated as $1 - D(x)$. A low output from the Discriminator (indicating it believes the sample is not normal) results in a high anomaly score. A dynamic threshold is then applied to these scores to classify samples as either normal or anomalous.

## 5.4 The RLClientSelectionStrategy

The `RLClientSelectionStrategy` coordinates the entire selection process. It acts as a connector between the RL agent's decisions and the rest of the framework. It simplifies the FL environment and provides an interface for the agent to observe the environment's state and take actions. In each round, the strategy is responsible for gathering the current state of the system, querying the agent for a selection decision, executing that decision by dispatching tasks to the chosen clients, and finally, translating the outcome of the round into a reward signal to guide the agent's learning.

The operation of the strategy in each round follows a defined sequence. First, the strategy instructs the selected clients to perform local training. After training, the selected clients return their updated local model parameters to the strategy. The strategy then aggregates these parameters, using Federated Averaging, to produce the updated global model for the current round. Next, during the evaluation phase, the parameters of this new global model are distributed to all available clients. Each client evaluates the model on its local validation data and returns the resulting metrics. Finally, the strategy aggregates these evaluation metrics to calculate the global reward signal, which is then passed to the RL agent to trigger its learning update.

From a technical standpoint, this is implemented as a custom Flower `Strategy` that extends the framework's standard `FedAvg` strategy, allowing it to inherit the basic aggregation logic while overriding the core client selection mechanism. Its primary responsibilities are:

- **State Management:** The strategy maintains a comprehensive, live dictionary of client attributes (`CLIENT_ATTRIBUTES`). After each fit and evaluation round, it intercepts the metrics returned from all participating clients (e.g., 'g_loss', 'd_loss', 'f1_score', 'avg_anomaly_score', 'cpu_cores') and updates this central store. This ensures the RL agent always has the most recent information available for decision-making.

- **Interaction with the RL Agent:** At the beginning of each selection phase, the strategy provides the RL agent with the necessary context, including the list of currently available clients, the current server round, and the performance of the global model from the previous round. It then calls the agent's `choose_actions_softmax` method to get the list of clients to select.

- **Triggering the Learning Cycle:** After a full federated round (fit and evaluate) is complete, the strategy calculates the global reward based on the aggregated evaluation metrics. It then invokes the agent's `learn_from_round_outcome` method, passing the calculated reward and other contextual information, which triggers the agent to update its Q-table.

- **Handling GAN Model Parameters:** The strategy is designed to handle the parameters of both the Generator and the Discriminator. It correctly packages the parameters from both global models to be sent to clients and correctly unpacks the updated parameters returned.

### 5.4.1 Client Attribute Store

For the RL agent to make informed decisions, it requires a continuous stream of data about the state and performance of each client in the federated network. This data collection is managed by the `RLClientSelectionStrategy`, which maintains a stateful Python dictionary, `CLIENT_ATTRIBUTES`, serving as a live, in-memory database of client history. After every `fit` and `evaluate` round, the strategy intercepts the metrics returned by each participating client and updates its corresponding entry in this store. This provides the RL agent with immediate access to the up-to-date information required to construct the state representation for each client. Table 5.1 details the key metrics collected for each client.

Table 5.1: Structure of the `CLIENT_ATTRIBUTES` store for each client.

| Attribute Key | Type | Description | Updated By |
|---|---|---|---|
| `partition_id` | str | Original integer ID of the client | Client |
| `cores` | int | Number of CPU cores reported by the client | Client |
| `last_fit_round` | int | Server round of the client's last fit participation | Strategy |
| `fit_rounds_participated` | List[int] | List of server rounds the client participated in fit | Strategy |
| `fit_g_loss_history` | List[float] | History of Generator loss from client fits | Strategy |
| `fit_d_loss_history` | List[float] | History of Discriminator loss from client fits | Strategy |
| `eval_f1_score_history` | List[float] | History of F1-Score from client evaluations | Strategy |
| `eval_avg_anomaly_score_hist` | List[float] | History of average anomaly score from evaluations | Strategy |
| `eval_precision_history` | List[float] | History of precision from client evaluations | Strategy |
| `eval_recall_history` | List[float] | History of recall from client evaluations | Strategy |

### 5.4.2 Strategy Bootstrapping: The Initial Rounds

The system does not begin with the RL agent making decisions immediately. A three-stage bootstrapping process is used to ensure a stable and informed start.

#### 5.4.2.1 Round 1: The Ghost Round

The simulation begins with a "ghost" round. In this initial round, no model training (`fit`) occurs. Instead, an `evaluate` instruction is sent to all available clients. This serves two critical purposes: first, it acts as a discovery and registration phase, allowing the server to populate its `CLIENT_ATTRIBUTES` store with the unique identifiers and hardware capabilities (e.g., CPU cores) of every client in the pool. The metrics gathered in this round provide the foundational data for the state representation of the agent.

#### 5.4.2.2 Round 2: Resource-Driven Selection

In the second round, we use a straightforward, resource-based selection approach that temporarily skips the RL agent's decision-making. Since we collected resource information in Round 1, the strategy now picks a particular group of clients based on their reported CPU cores. The objective is to ensure a successful and informative second round by selecting clients that are known to be capable. This round is primarily a `fit` round, providing the first set of training-related metrics (e.g., training loss) and creating the first meaningful update to the global model. This deliberate, heuristic-based selection jump-starts the learning process before the agent has accumulated enough experience to make any decisions.

#### 5.4.2.3 Round 3 and Beyond: RL-Agent Control

From the third round onward, the `RLAgent` assumes full control of the client selection process. With initial data on client attributes and performance from the first two rounds, the agent begins making decisions based on its Q-learning policy. The softmax temperature is at its highest at the start of this phase, promoting exploration to help the agent learn the value of selecting different types of clients. As the simulation progresses, the temperature decays and the agent gradually shifts from exploration to exploiting the knowledge captured in its Q-table, guided by the multi-criteria reward function. This three-stage initiation ensures a smooth and stable transition from a cold start to a fully operational, intelligent selection strategy.

## 5.5 The Reinforcement Learning Agent

The `RLAgent` component contains the decision-making logic for participant selection. It implements a Q-learning algorithm to learn a policy for selecting clients. The application of RL to client selection has been explored in prior work, for instance, using Q-Learning to manage clients with heterogeneous resources (Xu et al., 2023). Our work employs a Q-learning framework, chosen for its suitability to this problem. Its model-free nature is advantageous as it does not require an explicit model of the complex FL system's dynamics. Furthermore, its simplicity and stability in environments with discrete state spaces, such as the one designed for this work, make it a practical choice.

### 5.5.1 Markov Decision Process Formulation

The problem of selecting an optimal group of clients can be modeled as a Markov Decision Process (MDP), which provides a mathematical framework for decision-making. The key components of our MDP are defined as follows:

- **Agent:** The `RLAgent` residing on the FL server. It observes the environment and makes selection decisions.

- **Environment:** The entire FL system, including the pool of all clients, their respective data, their hardware capabilities, and the communication network connecting them to the server.
- **State ($s \in S$):** The state is not a single vector describing the whole system. Instead, at each round, the agent observes a set of individual state vectors, one for each available client. This allows the agent to evaluate each client's potential contribution independently. A client's state vector is a tuple of the following discretized features:
  1. **Average Local F1-Score:** The client's average F1-Score over its last $N$ evaluation participations.
  2. **Recent Participation:** The number of times the client was selected for training in the last $M$ rounds.
  3. **CPU Core Count:** The client's reported number of available CPU cores (binned into low, medium, and high).
  4. **Previous Global F1-Score:** The aggregated global F1-Score from the immediately preceding round.
  5. **Maximum Global F1-Score:** The all-time-high global F1-Score achieved so far in the simulation.

  This state representation allows the agent to consider the historical performance, recent selection frequency, and hardware capabilities of the clients, all within the context of the global model's current and best-ever performance.
- **Action ($a \in A$):** In each round, the agent performs a single, compound action. This action is the decision of **whether to select each client or not**, resulting in the selection of a subset of clients $\{c_1, c_2, ..., c_k\}$ from the pool of all available participants for the training round.
- **Reward ($R$):** The reward is the feedback signal the agent receives after an action is taken. The reward, $R_t$, is calculated at the end of the round based on the performance of the newly aggregated global model. It is a global signal shared by all clients that were selected in that round's action.

### 5.5.2 Action Selection and Policy

The agent's policy, $\pi$, defines its strategy for choosing an action (i.e., selecting a subset of clients) based on the observed states. A **softmax (Boltzmann) exploration** policy is used because it provides a balance between exploration and exploitation by probabilistically selecting clients based on their learned values. The process is as follows:

1. For each available client, the agent calculates its current state tuple, $s$, and retrieves the corresponding Q-value, $Q(s)$, from its Q-table.

2. These Q-values are converted into a probability distribution using the softmax function shown in Eq. 5.1, where $\tau$ is the temperature parameter.

$$P(\text{select}_i) = \frac{e^{Q(s_i)/\tau}}{\sum_j e^{Q(s_j)/\tau}} \tag{5.1}$$

3. The temperature $\tau$ controls the **exploration-exploitation trade-off**. It starts high to encourage exploration (trying novel client combinations) and decays over time, shifting the agent towards exploitation (picking clients with high known Q-values).

4. Finally, instead of static sampling, a dynamic probability threshold is used for selection. All clients whose selection probability surpasses this value are chosen. This allows the number of participants to adapt each round, enabling the agent to select a larger group when it has high confidence in many clients and a smaller one when it is more uncertain.

### 5.5.3 Multi-Criteria Reward Function

The reward function is central to guiding the learning process of the agent. It is a composite signal, calculated by the 'RLClientSelectionStrategy' after each evaluation round, designed to encapsulate the multiple, often competing, objectives of an effective FL system. The total reward $R$ is formulated as shown in Eq. 5.2. The various weighting factors used within these components were set through manual experimentation over multiple simulation runs. Each component is detailed below.

$$R = R_{\text{performance}} - P_{\text{fairness}} - C_{\text{resource}} \tag{5.2}$$

#### 5.5.3.1 Performance Reward ($R_{\text{performance}}$)

This is the primary driver of the agent's behavior, incentivizing the selection of clients that contribute to a more effective global model. It is calculated based on the global evaluation metrics achieved in the current round ($t$), as shown in Eq. 5.3.

$$R_{\text{performance}} = (w_{f1} \cdot \Delta F1_t) + (w_p \cdot \Delta P_t) + (w_r \cdot \Delta R_t) + R_{\text{jackpot}} \tag{5.3}$$

- $\Delta F1_t, \Delta P_t, \Delta R_t$ represent the change in the global F1-Score, precision, and recall, respectively, compared to the previous round ($t-1$). This focus on the *improvement* encourages consistent progress rather than rewarding stagnant high performance.

- $w_{f1}, w_p, w_r$ are weighting factors that allow for prioritizing different aspects of model performance. The weights used, selected via experimentation, were $w_{f1} = 0.8$, $w_p = 0.25$, $w_r = 0.25$.

- $R_{\text{jackpot}}$ is a dynamic bonus awarded only if the F1-Score in the current round, $F1_t$, surpasses the maximum F1-Score ever recorded. This reward is scaled by the magnitude of the

improvement over the previous record, providing a proportional incentive to break performance plateaus. This component is zero if no new record is achieved.

#### 5.5.3.2 Fairness Penalty ($P_{\text{fairness}}$)

This component ensures that the agent does not develop a strong bias towards a small subset of high-performing clients, a phenomenon known as "client starvation". Because the state representation is on a per-client basis, it is natural to implement the fairness mechanism as a penalty for over-selecting individual clients. This promotes broader participation, which can lead to a more generalized and robust global model. The penalty is calculated as shown in Eq. 5.4.

$$P_{\text{fairness}} = f_{penalty} \cdot \sum_{c \in C_{selected}} \mathbb{I}(p_c > p_{thresh}) \tag{5.4}$$

- $C_{selected}$ is the set of clients selected in the current round.
- For each selected client $c$, $p_c$ is the number of times it has participated in training within a recent `recency_window` of $M$ rounds.
- $p_{thresh}$ is a participation threshold (e.g., 50% of the recency window).
- $\mathbb{I}(\cdot)$ is an indicator function that is 1 if the condition is true and 0 otherwise.
- A penalty ($f_{penalty} = 0.01$) is incurred for each "over-selected" client in the chosen set, directly discouraging the agent from creating an elite group of participants.

#### 5.5.3.3 Resource Cost ($C_{\text{resource}}$)

This component adds resource efficiency considerations to how the agent makes decisions. It stops the agent from always picking the most powerful clients without it being necessary. CPU utilization was chosen over other metrics like memory, as it was found to be more dependent on the intrinsic hardware capabilities of the clients as opposed to their data quantity. The cost is calculated based on the clients' actual CPU utilization percentages, as shown in Eq. 5.5.

$$C_{\text{resource}} = c_{factor} \cdot \sum_{c \in C_{selected}} \text{utilization}_c \tag{5.5}$$

- $\text{utilization}_c$ is the CPU utilization percentage reported by client $c$.
- A cost factor ($c_{factor} = 0.02$) is multiplied by the total utilization, adding a penalty for choosing computationally expensive clients.

It is important to clarify that the specific weights and factors used in the reward function components ($w_{f1}, w_p, w_r, f_{penalty}, c_{factor}$) are user-defined hyperparameters. The values presented were determined through extensive manual experimentation over many simulation iterations and are particular to the specific characteristics of this work's setup. They are not presumed to be

universally optimal; a different simulation environment—with a different number of clients, other hardware or data characteristics, or a different underlying machine learning model—would likely require a re-tuning of these hyperparameters to achieve the best performance.

### 5.5.4 Q-Learning Update

After receiving the total reward for the round, the agent updates the Q-values for all clients that were selected. The standard Q-learning update rule, shown in Eq. 5.6, is applied.

$$Q(s) \leftarrow Q(s) + \alpha \left( R + \gamma \max_{a'} Q(s', a') - Q(s) \right) \tag{5.6}$$

Where:

- $s$ is the client's state when it was selected.
- $R$ is the total global reward received.
- $\alpha$ is the learning rate.
- $\gamma$ is the discount factor.
- $s'$ is the client's predicted state for the next round.
- $\max_{a'} Q(s', a')$ is the estimated maximum future Q-value from the next state.

This rule updates the value estimate $Q(s)$ by scaling the temporal difference error with the learning rate $\alpha$. The error is the difference between the target value ($R + \gamma \max_{a'} Q(s', a')$) and the current value $Q(s)$. It allows the agent to learn the long-term value of selecting a client in a particular state, based on the immediate reward and the potential for future rewards.

## 5.6 Summary

This chapter has provided a comprehensive overview of the implementation of RAMPART-FL. It began by detailing the architecture of the system, built upon Flower and Docker, and the meticulous data preprocessing pipeline designed to simulate a realistic, non-IID environment for intrusion detection. The design and semi-supervised training of the client-side GAN for anomaly detection were also presented. The chapter then transitioned to the core intelligent component: the RL agent. We detailed the formulation of the problem as a Markov Decision Process, the agent's state representation, its action selection policy, and the multi-criteria reward function designed to balance performance, fairness, and resource efficiency. Together, these sections provide a complete technical blueprint of the RAMPART-FL system, establishing the foundation for the experimental results and analysis presented in the following chapter.

# Chapter 6

# Experimental Evaluation and Results

This chapter details the experimental methodology, presents the results of the simulation runs, and provides a discussion of their implications. The primary objective of these experiments is to answer the research questions outlined in Chapter 1 by evaluating different reward structures. The analysis is presented in two parts: first, a comparative study is conducted to assess the performance of different strategies. Second, a detailed analysis of the multi-criteria agent's learning process is presented, examining the emergent behavioral phases during its training. This comprehensive evaluation serves to validate the effectiveness of the proposed multi-criteria participant selection methodology.

To ground the performance of the federated strategies, a centralized baseline GAN model was trained on the entire training dataset. The performance of this baseline, detailed in Table 6.1, served as a benchmark for the FL experiments. Given the imbalanced nature of the ToN-IoT Intrusion Detection dataset, where anomalous samples are rare, the F1-Score was used as the primary metric for evaluation throughout this chapter, as it provides a measure of a model's performance by considering both precision and recall.

Table 6.1: Performance Metrics of the Centralized Baseline Model.

| **Metric** | *F1-Score* | *Accuracy* | *Precision* | *Recall* |
|---|---|---|---|---|
| **Value** | 0.8310 | 0.7562 | 0.9990 | 0.7113 |

This chapter presents an analysis of the performance and behavior of the RAMPART-FL system. It begins with a comparative analysis of four distinct RL agent reward strategies, measuring the multi-criteria reward approach against single-objective reward options. It then analyzes the behavior of the `Multi-Criteria` agent, dissecting its learning process into three distinct phases and using a suite of diagnostic plots to provide an evidence-based explanation for its adaptation and convergence.

## 6.1 Comparative Analysis of Reward Strategies

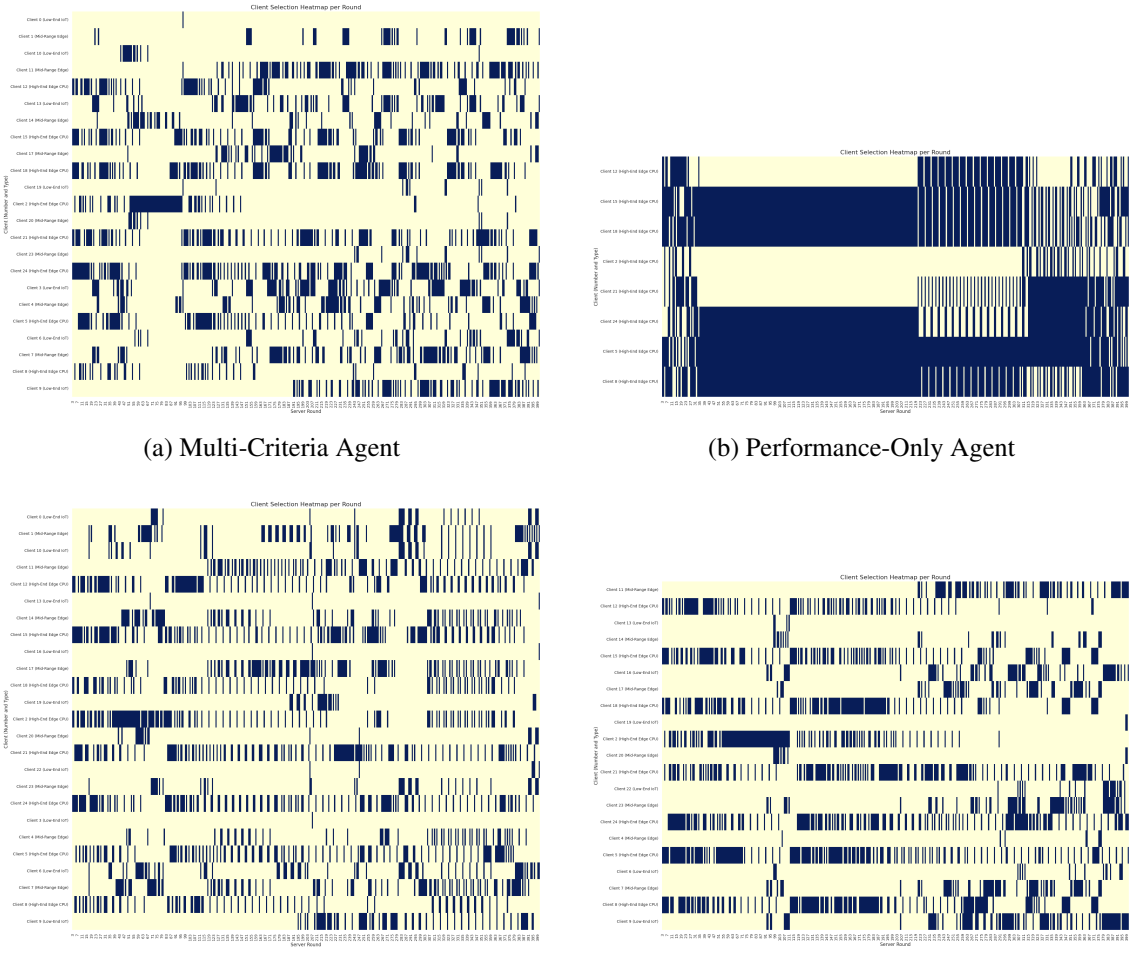
To evaluate the effectiveness of the multi-criteria reward function, a comparative analysis was conducted between four agent configurations over 400 rounds with 25 clients under high data heterogeneity (`Dirichlet Alpha = 0.3`). The strategies were:

1. **`Multi-Criteria`**: The primary proposed agent, balancing performance, fairness, and resource cost.
2. **`Performance-Only`**: An agent rewarded solely for increasing the global model performance metrics.
3. **`Fairness-Only`**: An agent rewarded solely for selecting a diverse set of clients, penalizing frequent re-selection.
4. **`Resource-Only`**: An agent rewarded solely for minimizing resource utilization.

### 6.1.1 Client Selection & Fairness

The experiment revealed distinct differences in selection behavior, as visualized in the heatmaps in Fig. 6.1. The `Performance-Only` agent exhibited significant selection bias, resulting in client starvation for most of the experiment, before switching up its selection approach in the later rounds. The other three agents ensured diverse participation but with different underlying strategies. The `Fairness-Only` agent adopted a cyclical pattern, enforcing diversity by rotating through clients. This strategy creates a selection heatmap that looks like a barcode since participation gets spread out systematically to avoid picking the same clients in quick succession. The `Resource-Only` agent demonstrated a clear learning process: it began by selecting high-cost clients but, after learning from the resource penalties, shifted to consistently selecting low and medium-resource clients. Finally, the `Multi-Criteria` agent displayed a more complex strategy, learning to balance objectives by favoring high-performing clients when beneficial, while also rotating in others to maintain fairness. It is noteworthy, however, that even with these more diverse strategies, some clients were still selected infrequently across all three agents, indicating that perfect fairness remains a challenge.

A cost-benefit analysis distinguishes the adaptive behavior of the `Multi-Criteria` agent from the cyclical behavior of the `Fairness-Only` agent. The `Fairness-Only` agent, seeking only to minimize participation counts, rotates clients frequently. It might drop a high-performing client simply because its participation count is high. In contrast, the `Multi-Criteria` agent learned a more adaptive strategy. It would tolerate the rising fairness "cost" of re-selecting a valuable client if the expected performance "benefit" was high enough, leading to longer selection streaks. Conversely, it learned to sideline clients that yielded poor performance, as they represent a high cost for no benefit. This strategic thinking, absent in the fairness-only approach, is an indicator of the adaptability of the multi-criteria agent.




(a) Multi-Criteria Agent

(b) Performance-Only Agent

(c) Fairness-Only Agent

(d) Resource-Only Agent

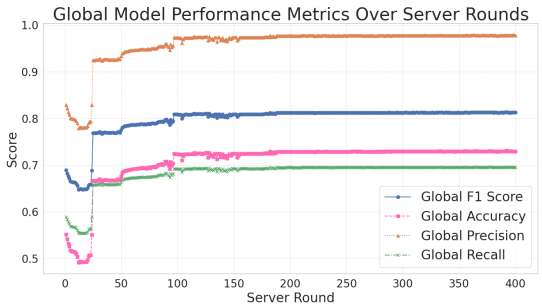
Figure 6.1: Comparative client selection heatmaps for each reward strategy. The `Performance-Only` agent (b) shows selection bias, while the other three demonstrate broader, more equitable participation.

### 6.1.2 Model Performance & Convergence

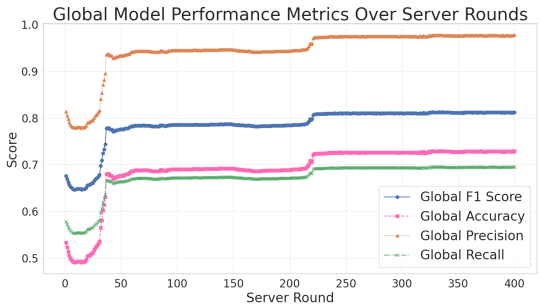
As illustrated by the performance metrics in Fig. 6.2, all strategies managed to get the federated system close to what the centralized baseline achieved with less than a 2% gap in F1-Scores. This tells us the federated approach works well overall. However, the different strategies took different approaches to get there. The `Performance-Only` agent struggled at first. It kept picking the same subset of clients, not being exposed to enough variety in the data, which made the initial learning slow. On the contrary, both the `Multi-Criteria` and `Fairness-Only` approaches were quicker to reach their plateau. They both selected more clients to participate, giving the system access to more diverse data early on. The `Resource-Only` agent fell somewhere in between, not as fast as the ones with a fairness component, but still better than the performance-focused approach.

The late convergence of the `Performance-Only` agent warrants a specific explanation. For a

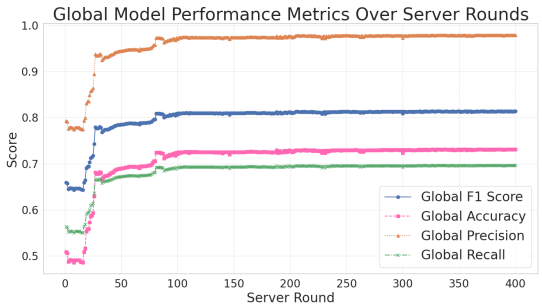
significant portion of the experiment, the agent was stuck in a local optimum, repeatedly selecting a sub-optimal subset of clients, as shown in its selection heatmap (Fig. 6.1b). This led to the performance plateau visible for over 200 rounds (Fig. 6.2b). The trigger for its escape from this plateau is visible in its reward dynamics, shown in Fig. 6.3. Following an extended period where the agent received zero performance reward, there was a small, isolated positive reward that showed up around round 350. This reward was enough to push the agent's learned strategy in a different direction. The change was magnified because of how the softmax selection policy works. This small shift was enough for the agent to try a new combination of clients. This new subset proved to be more effective, leading to the rapid performance increase that brought the agent's final F1-Score in line with the other strategies. This event highlights how even a purely exploitative agent can escape local optima due to minor environmental fluctuations and the inherent exploratory component of its policy.



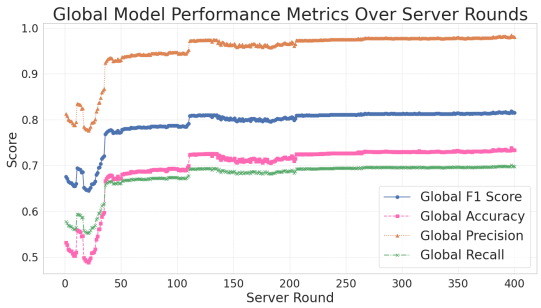

(a) Multi-Criteria Agent




(b) Performance-Only Agent




(c) Fairness-Only Agent




(d) Resource-Only Agent

Figure 6.2: Comparative evolution of global performance metrics. While all strategies reach a similar peak F1-Score, the `Performance-Only` agent (b) converges at a slower rate. Legend: F1-Score, Precision, Recall, Accuracy.

### 6.1.3 Resource Efficiency

The agents also exhibited distinct resource usage profiles, as shown in Fig. 6.4. The `Performance-Only` agent was consistently the most resource-intensive. In contrast, the `Resource-Only` agent's usage mirrored its selection heatmap: it started high but learned to lower its consumption over time, ending as the most efficient strategy. The `Fairness-Only` agent's usage oscillated between

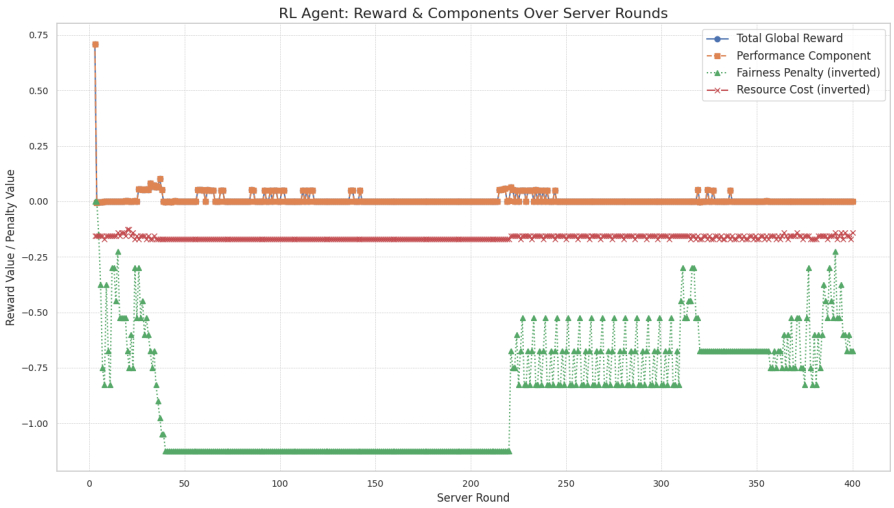
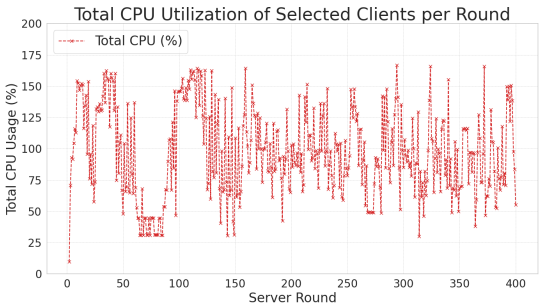
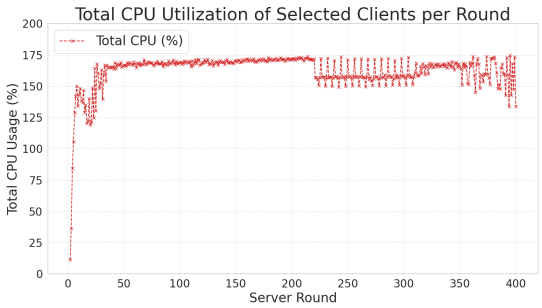



Figure 6.3: The reward dynamics for the `Performance-Only` agent. The total reward is driven entirely by the performance component. A small, isolated reward spike is visible around round 350 after a long period of zero reward.

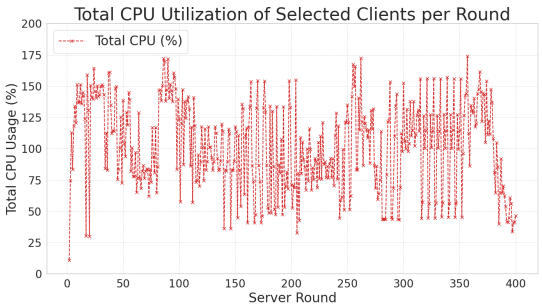
high and low-cost clients, reflecting its cyclical selection pattern. The `Multi-Criteria` agent achieved a balance, managing to be more consistent by picking clients "in the middle" more often, and while it had peaks, it avoided the regular oscillations of the fairness-only approach.



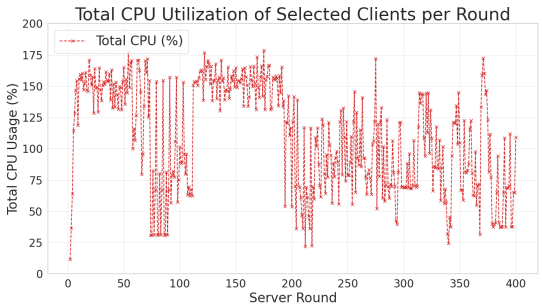

(a) Multi-Criteria Agent




(b) Performance-Only Agent




(c) Fairness-Only Agent




(d) Resource-Only Agent

Figure 6.4: Comparative resource usage, measured as the sum of client CPU utilization percentages per round. Each agent exhibits a distinct resource consumption profile reflecting its underlying selection strategy.

An emergent behavior was observed in the `Resource-Only` agent. While its sole objective was to minimize cost, it did not exclusively select low-resource clients from the start. Instead, its learning process unfolded in distinct stages:

1. **Initial Exploration of High-Cost States:** Driven by an optimistic initialization that makes high-resource clients seem promising until tried, the agent first selected these clients.
2. **Learning from Penalties:** It received negative rewards for these choices, learning their high cost. This incentivized the exploration of unknown mid-range and then low-range clients.
3. **Converged Exploitation:** After mapping the cost landscape, the agent entered a pure exploitation phase in the later rounds (300+). By this point, it had learned that low-resource clients yielded the lowest penalty (i.e., the highest reward) and began selecting them almost exclusively.

This observed shift provides evidence of the Q-learning process navigating the environment to optimize its objective.

### 6.1.4 Conclusion of the Comparative Analysis

The comparative analysis confirms that the `Performance-Only` strategy is sub-optimal. Its selection bias leads to an unfair, resource-intensive, and inefficient training process, resulting in the slowest convergence. The results indicate that incorporating fairness and resource considerations is important for a robust FL system. All three alternative strategies guided the model to a final F1-Score of approximately 0.82. However, they achieved this through different means. The `Resource-Only` agent becomes the most resource-efficient, but its convergence speed is middling. The `Fairness-Only` agent achieves rapid convergence but does so with a cyclical selection pattern that leads to inconsistent resource usage.

The `Multi-Criteria` agent emerges as an effective strategy. It matches the convergence rate of the `Fairness-Only` agent while maintaining more stable resource usage. Its ability to make strategic trade-offs, unlike the patterns of the `Fairness-Only` agent, demonstrates a more adaptive approach. This ability to weigh competing objectives supports the benefits of a balanced reward signal and establishes the `Multi-Criteria` agent as a well-rounded approach.

## 6.2 Analysis of the Learning Phases of the Multi-Criteria Agent

The following analysis dissects a simulation run that used the `Multi-Criteria` agent. This run revealed a three-phase learning process. By synthesizing insights from a range of diagnostic plots, it is possible to ascertain an understanding of how the RL agent perceived the state of the FL system, adapted its strategy, and optimized its objectives.

### 6.2.1 Stage 1: Exploration and Instability (Rounds $\approx 3-40$)

The first phase was characterized by a counterintuitive dip in model performance, which was later understood to be a necessary learning period.

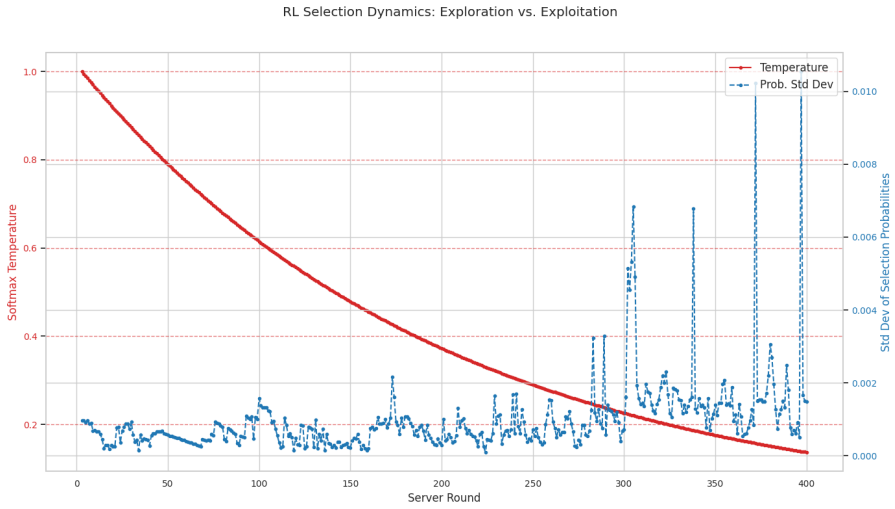



Figure 6.5: RL learning selection dynamics show high temperature and low probability variance, indicating exploration early on.

#### The Agent's Behavior

As shown in Fig. 6.5, Stage 1 was defined by the **exploration** policy of the RL agent. The softmax temperature was at its highest, and the standard deviation of selection probabilities was at its lowest. This confirms that the agent had no pre-existing knowledge and was making random client selections to gather information about the environment.

#### The Consequence for the GAN

This random selection had a destabilizing effect on the federated GAN, which was compounded by the "growing window" data mechanism, where clients possess less data in these early stages. The combination of random selection across non-IID clients and limited initial training data led to high variance among local models—a phenomenon shown by the wide distribution of client losses seen in Fig. 6.6—which in turn destabilized the central aggregated model and explains the dip in the global F1-Score (Fig. 6.2a).

The wide variance for both G-Loss and D-Loss in the first $\approx 30-40$ rounds (Fig. 6.6) shows that the randomly selected clients, with their non-IID data, were learning in divergent ways. This disagreement among local models destabilized the central, aggregated global model when their weights were averaged.

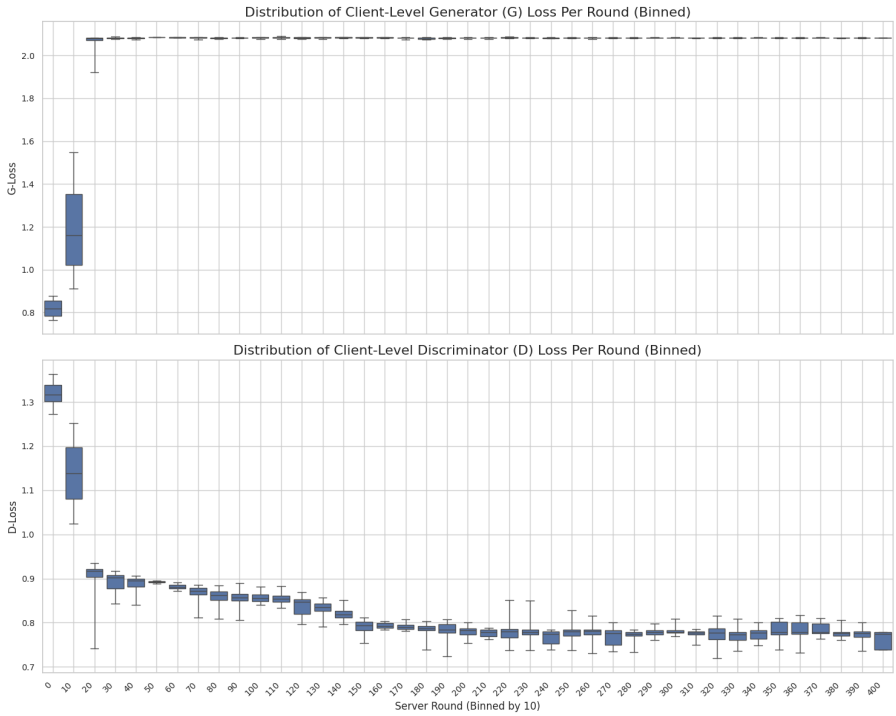


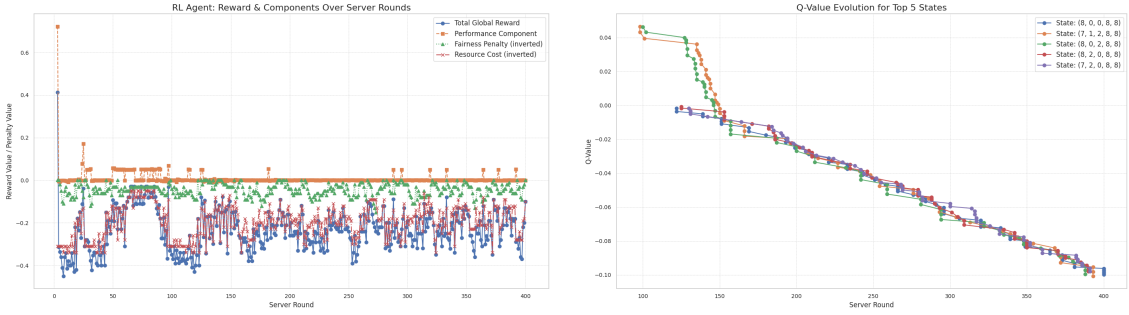

Figure 6.6: Distribution of local Generator (G) and Discriminator (D) losses. The large boxes in early rounds show high variance among client models.

### The Outcome

The unstable global GAN, a result of the agent's exploration, led to the observed dip in the global F1-Score (Fig. 6.2a). The agent temporarily sacrificed performance to build a foundational understanding of the client pool.

## 6.2.2 Stage 2: Exploitation and Rapid Improvement (Rounds $\approx 40 - 100$)

In this phase, the agent began to apply its knowledge, leading to an increase in performance.




(a) RL rewards and components. Legend: Total Reward, Performance, Fairness (inv.), Resource (inv.).

(b) Q-values for the most visited states begin to stabilize.

Figure 6.7: RL learning dynamics in Stage 2. The increase in total reward (a, blue line) was driven by performance rewards. This updated the Q-values, which began to converge (b).

#### The Agent's Behavior

The agent's temperature decayed, causing it to shift from exploration to **exploitation**. It now used its Q-table, which had been updated based on the rewards from Stage 1, to select clients it predicted would be valuable.

#### Evidence of Learning

The learning process of the agent is shown in the reward dynamics. This stage was marked by an increase in the total global reward (Fig. 6.7a), driven by the "jackpot" rewards the agent received for achieving new all-time-high F1-Scores. As shown in Fig. 6.7b, the Q-values for the most frequently visited states began to converge, indicating the agent was learning their long-term value.

#### The Outcome

By selecting more synergistic client combinations, the variance in local training decreased (Fig. 6.6 shows tightening boxplots). This stabilized the global GAN, and the global F1-Score (Fig. 6.2a) began its ascent, reaching its peak performance before round 100.

### 6.2.3 Stage 3: Convergence and Management (Rounds $\approx 100 - 400$)

In the final stage, the system reached a state of equilibrium, and the agent's task shifted from optimization to stable management.

#### The Agent's Behavior

At this point, the global model had hit its performance ceiling (Fig. 6.2a). Performance-based rewards started becoming scarce (Fig. 6.7a). This shift meant that the other parts of the reward function, specifically the fairness and resource cost penalties, began playing a bigger role in how the agent made its decisions. The focus of the agent shifted to keeping performance high while staying within the fairness and resource limits. Looking at Fig. 6.1a, we can see that even though the agent had its preferred clients, it still made sure to include others in the rotation to prevent the same subset of clients from being selected over and over. This is due to the fairness penalty. In the same way, Fig. 6.4a demonstrates that the agent learned not to always select the most powerful (and expensive) clients, which helped keep overall resource consumption reasonable.

#### The Outcome

The system converged to a stable state. The RL agent navigated the trade-offs, creating a high-performing global model while adhering to the operational constraints of fairness and resource efficiency.

## 6.3 Summary

The RAMPART-FL system demonstrated that a multi-criteria RL agent is a viable solution for participant selection in a complex FL environment. The analysis revealed a learning process where the agent navigated the trade-offs between performance maximization, fairness, and resource management. The comparative analysis showed that while single-objective agents can perform well in their dedicated dimension, they create imbalances in others. The multi-criteria agent, in contrast, learned an adaptive strategy, achieving high performance and efficiency while maintaining fairness. It learned to overcome the initial instability caused by its exploration and the constraints of the environment, adapted its strategy as the global model matured, and converged to a stable, high-performance state. This work provides evidence that such adaptive agents are a component for building robust and fair FL systems for real-world applications.

# Chapter 7

# Conclusion and Future Work

This dissertation embarked on an investigation into one of the key challenges of FL: participant selection. Set against the backdrop of IDSs for heterogeneous IoT and edge environments, the central problem was identified not as a shortcoming of the IDS models themselves, but as the sub-optimal nature of how clients are chosen to collaboratively train them. The research was motivated by the observation that conventional selection strategies do not navigate the multi-objective landscape of real-world FL deployments, where model performance, client fairness, and resource efficiency are often in tension.

To address this, the RAMPART-FL system was proposed, designed, and evaluated. At its heart is an RL agent tasked with learning an adaptive selection policy. This work has detailed the journey from the formulation of this problem as a Markov Decision Process, through the design of the system and the RL agent, to an empirical evaluation of its performance. This final chapter showcases this research's discoveries, returns to the questions that guided our work, acknowledges this work's shortcomings, and proposes promising directions for future investigation.

## 7.1 Summary of Findings

Through this dissertation research, we have achieved several important findings and contributions:

1. **A Novel RL-Driven Participant Selection Strategy:** The primary contribution is the design and implementation of RAMPART-FL, a novel strategy for FL that uses a multi-criteria RL agent to orchestrate participant selection.

2. **Effectiveness of Multi-Criteria Optimization:** The comparative analysis demonstrated that the `Multi-Criteria` agent, which balances performance, fairness, and resource costs, provides an effective solution. It achieved model convergence at a rate comparable to a `Fairness-Only` strategy but with stable resource usage and a trade-off-aware selection policy. This finding shows how important it is to take a comprehensive approach when optimizing these systems.

3. **Insight into Agent Learning Dynamics:** When we analyzed how the `Multi-Criteria` agent behaved, we found a three-phase learning process: an initial, destabilizing exploration phase; a rapid exploitation and improvement phase; and a final, stable management phase. This analysis provides empirical evidence of the capacity of the agent to adapt to and navigate a non-IID federated environment.

## 7.2 Additional Contributions

An important contribution of this work is the creation and public release of a detailed experimental dataset, alongside the open sourcing of the RAMPART-FL framework code. These artifacts were created to provide valuable resources for the research community, allowing for the replication of our results and serving as a foundation for future work.

### 7.2.1 Experimental Dataset

The dataset was generated from the 400-round simulation run with 25 clients using the `Multi-Criteria` agent strategy. Its purpose is to provide a resource for researchers to further investigate, develop, and benchmark RL-based participant selection strategies in FL environments.

The complete dataset is available on Hugging Face[1].

The dataset is provided as a single CSV file where each row represents a specific event that occurred during the FL simulation. The data is structured to capture the complete learning cycle of the RL agent, including the state representation of each client before selection, the action taken by the agent, and the resulting rewards and performance metrics. This allows for an analysis of the agent's behavior and provides the necessary components (State, Action, Reward) for training and evaluating other RL agents.

While the internal logging captured a wide range of metrics for our analysis, the final published dataset has been curated to include the fields most relevant for the task of participant selection. Table 7.1 describes the key columns available in the dataset.

### 7.2.2 Open-Source Framework

To further promote transparency and encourage the extension of this work, the source code for the RAMPART-FL framework has been made publicly available. The repository contains the necessary components to set up the simulation environment and run experiments with different configurations.

The framework is available on GitHub[2].

This curated dataset and open-source code provide a foundation for researchers to replicate these findings and build upon this work.

[1] `https://huggingface.co/datasets/LCS-FS/RAMPART-FL`
[2] `https://github.com/LCS-FS/RAMPART-FL`

Table 7.1: Description of Key Columns in the RAMPART-FL Experimental Dataset.

| Column Name | Description |
|---|---|
| `server_round` | The server round in which the event was logged. |
| `event_type` | The type of event, such as `selection_info`, `learning_update`, or `global_eval_metrics`. |
| `client_cid` | The unique identifier for the client the row pertains to. Is null for round-wide global metrics. |
| `s_client_state_tuple` | The state vector representing the client's condition at the moment of selection. This is the primary input for the RL agent's decision. |
| `s_was_selected` | The action taken by the agent for this client (1 if selected for training, 0 otherwise). |
| `l_reward_for_action` | The total global reward calculated for the set of actions taken in the round. This is the main feedback signal for the RL agent. |
| `gm_eval_f1` | The aggregated global F1-Score of the model after the round's evaluation phase. |
| `gm_eval_precision` | The aggregated global precision of the model after the round. |
| `gm_eval_recall` | The aggregated global recall of the model after the round. |
| `c_eval_f1` | The individual client's F1-Score from its local evaluation. This is a key component of the client's state. |
| `c_eval_cores` | The number of CPU cores reported by the client. This is a key component of the client's state. |
| `c_fit_time_seconds` | The time in seconds the client took to complete its local training (fit) task. |

## 7.3 Revisiting the Research Questions

This dissertation was guided by three research questions, which can now be answered based on the experimental results.

**RQ1: What are the effective design principles for an RL agent's state representation and reward function in a federated IDS, and how do these design choices manifest in the agent's learning behavior over time?** This research identified two key design principles for an effective RL agent. First, a state representation that combines local client data (performance, participation) with global system context (model progress) is sufficient to inform a learning policy. Second, a multi-criteria reward function that uses distinct components for performance, fairness, and resource cost is crucial for balancing competing objectives. These design choices were shown to be directly manifest in the learning behavior of the agent. The agent's policy, governed by a decaying temperature parameter, produced an initial exploratory phase. The performance "jackpot" component of the reward function drove a subsequent phase of rapid exploitation and improvement. Finally, the fairness and resource penalties became the dominant guide in the third phase of stable,

long-term management, demonstrating a clear link between the agent's design and its emergent adaptive strategy.

**RQ2: How does a multi-criteria, RL-driven participant selection mechanism adapt its policy to successfully guide a federated IDS model to convergence in an environment characterized by high system heterogeneity and Non-IID data?** The research demonstrated that the RL agent's adaptive policy was key to guiding the model to convergence in the heterogeneous environment. The agent's strategy evolved through distinct phases. In the beginning, the necessary exploration across the non-IID clients caused temporary instability, but it allowed for the understanding of the environment. As the agent learned from the reward signal, it transitioned to an exploitation phase, where it selected more effective combinations of clients. This shift brought stability to the training process and led to rapid performance gains. During this process, the policy was shaped by the multi-criteria reward. The fairness penalty encouraged the agent to draw from a diverse set of clients, preventing overfitting to a small group and ensuring the model learned from a representative sample of different data distributions. At the same time, the resource cost penalty incentivised the agent to balance performance gains against how much computational power the high-end clients would consume, which helped make efficient use of the network's shared resources. This learning approach, with its different phases and guided by the complex reward signal, shaped the behavior that led to successful handling of the interconnected challenges of data and system heterogeneity.

**RQ3: How does a multi-criteria RL-based participant selection strategy compare against strategies driven by single-objective reward functions (i.e., performance-only, fairness-only, and resource-only) in a simulated federated IDS environment?** The comparative study provided a clear answer regarding the trade-offs between reward philosophies. The `Performance-Only` strategy proved ineffective, leading to an unfair and resource-intensive system that converged slowly. The single-objective `Fairness-Only` and `Resource-Only` agents, while effective in their respective dimensions, created imbalances in others. In terms of achieving rapid model convergence and ensuring broad client participation, both the `Multi-Criteria` and `Fairness-Only` agents demonstrated superior and very similar results. However, the `Multi-Criteria` agent distinguished itself by demonstrating the capability of making strategic trade-offs, such as tolerating a higher fairness "cost" to re-select a high-performing client, a nuanced behavior absent in the more mechanical, cyclical pattern of the `Fairness-Only` agent. This supports the central hypothesis that a holistic, multi-criteria approach is necessary for building a robust federated system. Distinguishing the finer strategic advantages between these two high-performing agents would require experimentation under more complex environmental conditions, which remains an avenue for future work.

## 7.4 Discussion and Future Work

This dissertation has explored the challenge of participant selection in FL, demonstrating that a multi-criteria RL agent is a solution for creating adaptive, efficient, and fair decentralized learning systems. This work has shown that an adaptive federated strategy can achieve a final F1-Score of approximately 82%, with a performance gap of approximately 1% when compared to the 83% achieved by a centralized model with full data access. This result is significant, as it supports the federated approach by demonstrating that the benefits of the paradigm, enhanced data privacy, reduced network latency, and improved scalability, can be realized with a minor sacrifice in final model quality. The findings support the application of this approach in sensitive domains like intrusion detection, where preserving data privacy is a requirement.

Furthermore, this research supports the use of RL as a tool for the participant selection task itself. The RL agent demonstrated the capacity to adapt its strategy over time, progressing from a phase of broad exploration to a phase of targeted exploitation that improved model performance. The multi-criteria agent learned to navigate the trade-offs between performance, fairness, and resource cost, a balancing act that is difficult to encode with fixed rules. This was evidenced by its ability to achieve convergence by promoting diverse participation while still making trade-off-aware decisions about when to re-select high-value clients. This adaptive, learning-based approach not only optimized the final model but also the federated process, supporting RL as a mechanism for managing the operational dynamics of FL systems.

However, it is important to acknowledge several limitations of this study. First, the findings are based on a single benchmark dataset, ToN-IoT. While representative for IDS, validating the RAMPART-FL strategy across a wider range of datasets is necessary to confirm its generalizability. Second, our simulation of Non-IID data relies on a Dirichlet distribution, which is a synthetic proxy for real-world data distributions. This is compounded by the fact that the experiments were conducted within a Docker-based simulation, which does not fully capture the unpredictability of real-world network conditions. Furthermore, the performance of the RL agent is sensitive to its hyperparameters, which would require careful tuning for different scenarios. Finally, our current work assumes a cooperative and honest client environment and does not implement defenses against malicious actors.

These limitations suggest several avenues for future work. A critical next step is to test the strategy's resilience against adversarial scenarios, such as the presence of Byzantine and malicious clients attempting to poison and destabilize the global model, and to investigate whether the agent can learn to avoid and isolate such participants. Future work should also aim to evaluate the system on naturally partitioned, real-world data. Other valuable directions include conducting larger-scale simulations to test scalability, deploying the system on a physical testbed, and enriching the agent's state representation and reward function with additional metrics to further improve its adaptability.

In conclusion, this research provides evidence that intelligent agents are a crucial component for building robust and fair FL systems. The insights, methodologies, and publicly available dataset contributed by this work are intended to support future research in this emerging area, to

develop intelligent IDSs capable of operating effectively within the constrained and decentralized nature of IoT and Edge environments.

# Bibliography


Abdel-Basset, M., Moustafa, N., & Hawash, H. (2022). Privacy-Preserved Cyberattack Detection in Industrial Edge of Things (IEoT): A Blockchain-Orchestrated Federated Learning Approach. *IEEE Transactions on Industrial Informatics*, 18(11), 7920–7934. https://doi.org/10.1109/tii.2022.3167663

Abdel-Basset, M., Moustafa, N., & Hawash, H. (2023). Privacy-Preserved Generative Network for Trustworthy Anomaly Detection in Smart Grids: A Federated Semisupervised Approach. *IEEE Transactions on Industrial Informatics*, 19(1), 995–1005. https://doi.org/10.1109/tii.2022.3165869

Abdulrahman, S., Tout, H., Mourad, A., & Talhi, C. (2021). FedMCCS: Multicriteria Client Selection Model for Optimal IoT Federated Learning. *IEEE Internet of Things Journal*, 8(6), 4723–4735. https://doi.org/10.1109/jiot.2020.3028742

Abosata, N., Al-Rubaye, S., & Inalhan, G. (2022). Customised Intrusion Detection for an Industrial IoT Heterogeneous Network Based on Machine Learning Algorithms Called FTL-CID. *Sensors*, 23(1), 321. https://doi.org/10.3390/s23010321

Alazab, M., RM, S. P., M, P., Maddikunta, P. K. R., Gadekallu, T. R., & Pham, Q.-V. (2022). Federated Learning for Cybersecurity: Concepts, Challenges, and Future Directions. *IEEE Transactions on Industrial Informatics*, 18(5), 3501–3509. https://doi.org/10.1109/tii.2021.3119038

Ali, A. H., Charfeddine, M., Ammar, B., Hamed, B. B., Albalwy, F., Alqarafi, A., & Hussain, A. (2024). Unveiling machine learning strategies and considerations in intrusion detection systems: a comprehensive survey. *Frontiers in Computer Science*, 6. https://doi.org/10.3389/fcomp.2024.1387354

Aljrees, T., Kumar, A., Singh, K. U., & Singh, T. (2023). Enhancing IoT Security through a Green and Sustainable Federated Learning Platform: Leveraging Efficient Encryption and the Quondam Signature Algorithm. *Sensors*, 23(19), 8090. https://doi.org/10.3390/s23198090

Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., & Anwar, A. (2020). TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access*, 8, 165130–165150. https://doi.org/10.1109/ACCESS.2020.3022862

Aouedi, O. & Piamrat, K. (2024). SURFS: Sustainable IntrUsion Detection with HieraRchical Federated Spiking Neural Networks. *In Proceedings of the IEEE International Conference on Communications (ICC 2024), Denver, CO, USA, 9-13 June 2024*, 2173–2178. https://doi.org/10.1109/icc51166.2024.10622560

Aouedi, O., Piamrat, K., Muller, G., & Singh, K. (2023). Federated Semisupervised Learning for Attack Detection in Industrial Internet of Things. *IEEE Transactions on Industrial Informatics*, 19(1), 286–295. https://doi.org/10.1109/tii.2022.3156642

Arya, M., Sastry, H., Dewangan, B. K., Rahmani, M. K. I., Bhatia, S., Muzaffar, A. W., & Bivi, M. A. (2023). Intruder Detection in VANET Data Streams Using Federated Learning for Smart City Environments. *Electronics*, 12(4), 894. https://doi.org/10.3390/electronics12040894

Bahar, A. A. M., Ferrahi, K. S., Messai, M.-L., Seba, H., & Amrouche, K. (2024). FedHE-Graph: Federated Learning with Hybrid Encryption on Graph Neural Networks for Advanced Persistent Threat Detection. *In Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES '24), Vienna, Austria, July 30 - August 2, 2024*, 1–10. https://doi.org/10.1145/3664476.3670466

Bekbulatova, V., Morichetta, A., & Dustdar, S. (2023). FL-SERENADE: Federated Learning for SEmi-supeRvisEd Network Anomaly DEtection. A Case Study. *In Proceedings of the 2023 IEEE International Conf on Dependable, Autonomic and Secure Computing, International Conf on Pervasive Intelligence and Computing, International Conf on Cloud and Big Data Computing, International Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Abu Dhabi, United Arab Emirates, 13-17 November 2023*, 1072–1079. https://doi.org/10.1109/dasc/picom/cbdcom/cy59711.2023.10361504

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K. H., Parcollet, T., de Gusmão, P. P. B., & Lane, N. D. (2022). *Flower: A Friendly Federated Learning Research Framework*. https://doi.org/10.48550/arXiv.2007.14390

Chelli, M., Prigent, C., Schubotz, R., Costan, A., Antoniu, G., Cudennec, L., & Slusallek, P. (2023). FedGuard: Selective Parameter Aggregation for Poisoning Attack Mitigation in Federated Learning. *In Proceedings of the 2023 IEEE International Conference on Cluster Computing (CLUSTER), Santa Fe, NM, USA, Oct 31 - Nov 3, 2023*, 72–81. https://doi.org/10.1109/cluster52292.2023.00014

Chen, D., Wang, K., Sundar, A. P., & Li, F. (2023). TrustNetFL: Enhancing Federated Learning with Trusted Client Aggregation for Improved Security. *In Proceedings of the 24th International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23), Washington DC, USA, October 23-26, 2023*, 534–539. https://doi.org/10.1145/3565287.3617635

Chen, Z., Tian, P., Liao, W., & Yu, W. (2021). Zero Knowledge Clustering Based Adversarial Mitigation in Heterogeneous Federated Learning. *IEEE Transactions on Network Science and Engineering*, 8(2), 1070–1083. https://doi.org/10.1109/tnse.2020.3002796

Chopra, D. & Khurana, R. (2023). *Introduction to Machine Learning with Python*. Bentham Science Publishers. https://doi.org/10.2174/9789815124422123010 1

Cui, B. & Mei, T. (2023). ABFL: A Blockchain-enabled Robust Framework for Secure and Trustworthy Federated Learning. *In Proceedings of the Annual Computer Security Applications Conference (ACSAC '23), Austin*
*San Antonio, TX, USA, December 4-8, 2023*, 636–646. https://doi.org/10.1145/3627106.3627121

Cui, J., Sun, H., Zhong, H., Zhang, J., Wei, L., Bolodurina, I., & He, D. (2023). Collaborative Intrusion Detection System for SDVN: A Fairness Federated Deep Learning Approach. *IEEE Transactions on Parallel and Distributed Systems*, 34(9), 2512–2528. https://doi.org/10.1109/tpds.2023.3290650

de Cámara, X. S., Flores, J. L., Arellano, C., Urbieta, A., & Zurutuza, U. (2023). Clustered federated learning architecture for network anomaly detection in large scale heterogeneous IoT networks. *Computers & Security*, 131, 103299. https://doi.org/10.1016/j.cose.2023.103299

Fan, S., Shi, H., Wang, C., Ma, R., & Wang, X. (2024). RFL-LSU: A Robust Federated Learning Approach with Localized Stepwise Updates. *ACM Transactions on Internet Technology*, 24(4), 1–26. https://doi.org/10.1145/3690822

Friha, O., Ferrag, M. A., Shu, L., Maglaras, L., Choo, K.-K. R., & Nafaa, M. (2022). FELIDS: Federated learning-based intrusion detection system for agricultural Internet of Things. *Journal of Parallel and Distributed Computing*, 165, 17–31. https://doi.org/10.1016/j.jpdc.2022.03.003

Garcia, S., Parmisano, A., & Erquiaga, M. J. (2020). *Iot-23: A labeled dataset with malicious and benign iot network traffic*. Dataset published on Zenodo. https://doi.org/10.5281/zenodo.4743745. Report type: Dataset

Ghimire, B. & Rawat, D. B. (2022). Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things. *IEEE Internet of Things Journal*, 9(11), 8229–8249. https://doi.org/10.1109/jiot.2022.3150363

Guembe, B., Misra, S., & Azeta, A. (2024). Federated bayesian optimization xgboost model for cyberattack detection in internet of medical things. *Journal of Parallel and Distributed Computing*, 193, 104964. https://doi.org/10.1016/j.jpdc.2024.104964

Guo, Z., Yao, D., Yang, Q., & Liu, H. (2024). HiFGL: A Hierarchical Framework for Cross-silo Cross-device Federated Graph Learning. *In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), Barcelona, Spain, August 25-29, 2024*, 968–979. https://doi.org/10.1145/3637528.3671660

Hamidpour, H. & Bushehrian, O. (2023). A Round-Based Network Attack Detection Model Using Auto-encoder In IoT-Edge Computing. *In Proceedings of the 2023 7th International Conference on Internet of Things and Applications (IoT), Isfahan, Iran, May 24-25, 2023*, 1–6. https://doi.org/10.1109/iot60973.2023.10365347

Hamouda, D., Ferrag, M. A., Benhamida, N., Seridi, H., & Ghanem, M. C. (2024). Revolutionizing intrusion detection in industrial IoT with distributed learning and deep generative techniques. *Internet of Things*, 26, 101149. https://doi.org/10.1016/j.iot.2024.101149

Houda, Z. A. E., Hafid, A. S., & Khoukhi, L. (2023). MiTFed: A Privacy Preserving Collaborative Network Attack Mitigation Framework Based on Federated Learning Using SDN and Blockchain. *IEEE Transactions on Network Science and Engineering*, 10(4), 1985–2001. https://doi.org/10.1109/tnse.2023.3237367

Houda, Z. A. E., Khoukhi, L., & Brik, B. (2022). A Low-Latency Fog-based Framework to secure IoT Applications using Collaborative Federated Learning. *In Proceedings of the 2022 IEEE 47th Conference on Local Computer Networks (LCN), Sydney, Australia, 26-29 September 2022*, 343–346. https://doi.org/10.1109/lcn53696.2022.9843315

Husnoo, M. A., Anwar, A., Reda, H. T., Hosseinzadeh, N., Islam, S. N., Mahmood, A. N., & Doss, R. (2023). FeDiSa: A Semi-asynchronous Federated Learning Framework for Power System Fault and Cyberattack Discrimination. *In Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS 2023), Hoboken, NJ, USA, May 17-20, 2023*, 1–6. https://doi.org/10.1109/infocomwkshps57453.2023.10226030

Jebreel, N. M., Domingo-Ferrer, J., Blanco-Justicia, A., & Sánchez, D. (2024). Enhanced Security and Privacy via Fragmented Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(5), 6703–6717. https://doi.org/10.1109/tnnls.2022.3212627

Jiang, Z., Ye, P., He, S., Wang, W., Chen, R., & Li, B. (2024). Lotto: Secure Participant Selection against Adversarial Servers in Federated Learning. *In Proceedings of 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, August 14-16, 2024*, 343–360. https://www.usenix.org/conference/usenixsecurity24/presentation/jiang-zhifeng

Jin, D., Chen, S., He, H., Jiang, X., Cheng, S., & Yang, J. (2023). Federated Incremental Learning based Evolvable Intrusion Detection System for Zero-Day Attacks. *IEEE Network*, 37(1), 125–132. https://doi.org/10.1109/mnet.018.2200349

Kaggie, J. D. (2024). Addressing machine learning challenges with microcomputing and federated learning. *The Lancet Digital Health*, 6(2), e88–e89. https://doi.org/10.1016/s2589-7500(23)00266-2

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., & Zhao, S. (2021). Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*, 14(1–2), 1–210. 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. https://doi.org/10.1038/s42256-020-0186-1

Kolokotronis, N., Dareioti, M., Shiaeles, S., & Bellini, E. (2022). An Intelligent Platform for Threat Assessment and Cyber-Attack Mitigation in IoMT Ecosystems. *In Proceedings of the 2022 IEEE Globecom Workshops (GC Wkshps), Rio de Janeiro, Brazil, 4-8 December 2022*, 541–546. https://doi.org/10.1109/gcwkshps56602.2022.10008548

Konečný, J., McMahan, H. B., Ramage, D., & Richtárik, P. (2016). *Federated optimization: Distributed machine learning for on-device intelligence*. https://doi.org/10.48550/arXiv.1610.02527

Lee, Y., Park, S., & Kang, J. (2024). Security-Preserving Federated Learning via Byzantine-Sensitive Triplet Distance. *In Proceedings of the 2024 IEEE International Symposium on Biomedical Imaging (ISBI), Athens, Greece, May 27-30, 2024*, volume abs/2210.16519, 1–5. https://doi.org/10.1109/isbi56570.2024.10635545

Li, W., Fan, K., Zhang, J., Li, H., Lim, W. Y. B., & Yang, Q. (2025). Enhancing Security and Privacy in Federated Learning Using Low-Dimensional Update Representation and Proximity-Based Defense. *IEEE Transactions on Knowledge and Data Engineering*, 37(6), 3372–3385. https://doi.org/10.1109/tkde.2025.3539717

Li, X., Qu, Z., Zhao, S., Tang, B., Lu, Z., & Liu, Y. (2023). LoMar: A Local Defense Against Poisoning Attack on Federated Learning. *IEEE Transactions on Dependable and Secure Computing*, 20(1), 437–450. https://doi.org/10.1109/tdsc.2021.3135422

Liu, H., Zhang, S., Zhang, P., Zhou, X., Shao, X., Pu, G., & Zhang, Y. (2021). Blockchain and Federated Learning for Collaborative Intrusion Detection in Vehicular Edge Computing. *IEEE Transactions on Vehicular Technology*, 70(6), 6073–6084. https://doi.org/10.1109/tvt.2021.3076780

Liu, J., Wang, Z., Yang, Q., & Wu, S. X. (2022). A Three-layer Security Assurance Model for a Decentralized Federated Learning System. *In Proceedings of the 2022 31st Wireless and Optical Communications Conference (WOCC), Newark, NJ, USA, April 28-29, 2022*, 85–90. https://doi.org/10.1109/wocc55104.2022.9880591

Ma, X., Sun, H., Hu, R. Q., & Qian, Y. (2022). A New Implementation of Federated Learning for Privacy and Security Enhancement. *In Proceedings of the 2022 IEEE Global Communications Conference (GLOBECOM 2022), Rio de Janeiro, Brazil, 4-8 December 2022*, 4885–4890. https://doi.org/10.1109/globecom48099.2022.10001614

Mohammad Lotfollahi and Mahdi Jafari Siavoshani and Ramin Shirali Hossein Zade and Mohammdsadegh Saberian (2020). Deep packet: a novel approach for encrypted traffic classification using deep learning. *Software: Practice and Experience*, 24. https://doi.org/10.1007/s00500-019-04030-2

Moudoud, H. & Cherkaoui, S. (2022). Toward Secure and Private Federated Learning for IoT using Blockchain. *In Proceedings of the 2022 IEEE Global Communications Conference (GLOBECOM 2022), Rio de Janeiro, Brazil, 4-8 December 2022*, 4316–4321. https://doi.org/10.1109/globecom48099.2022.10000623

Moustafa, N. & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10-12 November 2015*, 1–6. https://doi.org/10.1109/milcis.2015.7348942

Neto, H. N. C., Hribar, J., Dusparic, I., Fernandes, N. C., & Mattos, D. M. (2024). FedSBS: Federated-Learning participant-selection method for Intrusion Detection Systems. *Computer Networks*, 244, 110351. https://doi.org/10.1016/j.comnet.2024.110351

Nguyen, Q. H., Hore, S., Shah, A., Le, T., & Bastian, N. D. (2025). FedNIDS: A Federated Learning Framework for Packet-Based Network Intrusion Detection System. *Digital Threats: Research and Practice*, 6(1), 1–23. https://doi.org/10.1145/3696012

Nguyen, T. & Thai, M. T. (2024). Preserving Privacy and Security in Federated Learning. *IEEE/ACM Transactions on Networking*, 32(1), 833–843. https://doi.org/10.1109/tnet.2023.3302016

Nguyen, T.-A., Le, L. T., Nguyen, T. D., Bao, W., Seneviratne, S., Hong, C. S., & Tran, N. H. (2024). Federated PCA on Grassmann Manifold for IoT Anomaly Detection. *IEEE/ACM Transactions on Networking*, 32(5), 4456–4471. https://doi.org/10.1109/tnet.2024.3423780

Nower, N. (2019). Privacy in IoT: Expectations, Causes of Concerns, and Reasons for Concern Mitigation. *International Journal of Computer Applications*, 177(25), 37–42. https://doi.org/10.5120/ijca2019919718

Pan, Q., Wu, J., Bashir, A. K., Li, J., Yang, W., & Al-Otaibi, Y. D. (2022). Joint Protection of Energy Security and Information Privacy for Energy Harvesting: An Incentive Federated Learning Approach. *IEEE Transactions on Industrial Informatics*, 18(5), 3473–3483. https://doi.org/10.1109/tii.2021.3105492

Pinto, A., Herrera, L.-C., Donoso, Y., & Gutierrez, J. A. (2023). Survey on Intrusion Detection Systems Based on Machine Learning Techniques for the Protection of Critical Infrastructure. *Sensors*, 23(5), 2415. https://doi.org/10.3390/s23052415

Sawada, H., Imaizumi, S., & Kiya, H. (2024). Enhancing Security Using Random Binary Weights in Privacy-Preserving Federated Learning. *In Proceedings of the 2024 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Taipei, Taiwan, 4-7 December 2024*, volume abs/2409.19988, 1–6. https://doi.org/10.1109/apsipaasc63619.2025.10848632

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), Funchal, Madeira, Portugal, January 22-24, 2018*. https://doi.org/10.5220/0006639801080116

Sheller, M. J., Edwards, B., Reina, G. A., Martin, J., Pati, S., Kotrotsou, A., Milchenko, M., Xu, W., Marcus, D., Colen, R. R., & Bakas, S. (2020). Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, 10(1), 12598. https://doi.org/10.1038/s41598-020-69250-1

Stevens, T., Skalka, C., Vincent, C., Ring, J., Clark, S., & Near, J. (2022). Efficient Differentially Private Secure Aggregation for Federated Learning via Hardness of Learning with Errors. *In Proceedings of 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, USA, August 10-12, 2022*, 1379–1395. https://www.usenix.org/conference/usenixsecurity22/presentation/stevens

Sutton, R. S. (1992). *Introduction: The Challenge of Reinforcement Learning. In R. S. Sutton (Ed.),* Reinforcement Learning, 1–3. Springer ScienceBusiness Media. https://doi.org/10.1007/978-1-4615-3618-5_1

Tabassum, A., Erbad, A., Lebda, W., Mohamed, A., & Guizani, M. (2022). FEDGAN-IDS: Privacy-preserving IDS using GAN and Federated Learning. *Computer Communications*, 192, 299–310. https://doi.org/10.1016/j.comcom.2022.06.015

Tan, Q., Li, Y., & Shin, B.-S. (2025). Defending against Backdoor Attacks in Federated Learning by Using Differential Privacy and OOD Data Attributes. *Computer Modeling in Engineering & Sciences*, 143(2), 2417–2428. https://doi.org/10.32604/cmes.2025.063811

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), Ottawa, ON, Canada, 8-10 July 2009*, 1–6. https://doi.org/10.1109/cisda.2009.5356528

Thapa, C., Chamikara, M. A. P., & Camtepe, S. A. (2021). *Advancements of Federated Learning Towards Privacy Preservation: From Federated Learning to Split Learning. In M. H. Rehman & M. M. Gaber (Eds.),* Federated Learning Systems: Towards Next-Generation AI, 79–109. Springer International Publishing. https://doi.org/10.1007/978-3-030-70604-3_4

Thein, T. T., Shiraishi, Y., & Morii, M. (2024). Personalized federated learning-based intrusion detection system: Poisoning attack and defense. *Future Generation Computer Systems*, 153, 182–192. https://doi.org/10.1016/j.future.2023.10.005

Wang, K., Gou, C., Duan, Y., Lin, Y., Zheng, X., & Wang, F.-Y. (2017). Generative adversarial networks: introduction and outlook. *IEEE/CAA Journal of Automatica Sinica*, 4(4), 588–598. https://doi.org/10.1109/jas.2017.7510583

Wang, L., Wei, Z., & Guo, W. (2023a). Securing IoT Communication Using Physical Sensor Data — Graph Layer Security with Federated Multi-agent Deep Reinforcement Learning. *In Proceedings of the 2023 8th International Conference on Signal and Image Processing (ICSIP), Suzhou, China, July 7-9, 2023*, volume abs/2302.12592, 860–865. https://doi.org/10.1109/icsip57908.2023.10271026

Wang, W., Wang, S., Bai, D., Zhao, C., Peng, L., & Chen, Z. (2022a). SFMD: A Semi-Supervised Federated Malicious Traffic Detection Approach in IoT. *In Proceedings of the 2022 IEEE International Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Abu Dhabi, UAE, Dec 19-21, 2022*, 774–781. https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom57177.2022.00104

Wang, Y., Zheng, W., Liu, Z., Wang, J., Shi, H., Gu, M., & Di, Y. (2023b). A Federated Network Intrusion Detection System with Multi-Branch Network and Vertical Blocking Aggregation. *Electronics*, 12(19), 4049. https://doi.org/10.3390/electronics12194049

Wang, Z., Wang, P., & Sun, Z. (2022b). SDN traffic anomaly detection method based on convolutional autoencoder and federated learning. *In Proceedings of the 2022 IEEE Global Communications Conference (GLOBECOM 2022), Rio de Janeiro, Brazil, 4-8 December 2022*, 4154–4160. https://doi.org/10.1109/globecom48099.2022.10001438

Willeke, M., Bierbrauer, D., & Bastian, N. D. (2023). Data-efficient, federated learning for raw network traffic detection. *In Proceedings of the Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications V (SPIE), Orlando, FL, USA, 30 April - 4 May 2023*, volume 12538, 31. https://doi.org/10.1117/12.2663092

Xhemrishi, M., Östman, J., Wachter-Zeh, A., & Amat, A. G. i. (2025). FedGT: Identification of Malicious Clients in Federated Learning With Secure Aggregation. *IEEE Transactions on Information Forensics and Security*, 20, 2577–2592. https://doi.org/10.1109/TIFS.2025.3539964

Xu, T., Liu, Y., Ma, Z., Huang, Y., & Liu, P. (2023). A DQN-Based Multi-Objective Participant Selection for Efficient Federated Learning. *Future Internet*, 15(6). https://doi.org/10.3390/fi15060209

Xu, Y., Su, M., Hou, J., & Nie, C. (2022). Federated Learning Scheme with Dual Security of Identity Authentication and Verification. *In Proceedings of the 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China, August 9-11, 2022*, 695–702. https://doi.org/10.1109/trustcom56396.2022.00099

Yan, G., Wang, H., Yuan, X., & Li, J. (2024). FedRoLA: Robust Federated Learning Against Model Poisoning via Layer-based Aggregation. *In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), Barcelona, Spain, August 25-29, 2024*, 3667–3678. https://doi.org/10.1145/3637528.3671906

Yin, J., Shi, Y., Deng, W., Yin, C., Wang, T., Song, Y., Li, T., & Li, Y. (2023). Internet of Things Intrusion Detection System Based on Convolutional Neural Network. *Computers, Materials & Continua*, 75(1), 2119–2135. https://doi.org/10.32604/cmc.2023.035077

Zaabar, B., Cheikhrouhou, O., & Abid, M. (2022). Intrusion Detection System for IoMT through Blockchain-based Federated Learning. *In Proceedings of the 2022 15th International Conference on Security of Information and Networks (SIN), Doha, Qatar, December 12-14, 2022*, 01–08. https://doi.org/10.1109/sin56466.2022.9970536

Zhang, Z., Zhang, Y., Li, H., Liu, S., Chen, W., Zhang, Z., & Tang, L. (2024). Federated continual representation learning for evolutionary distributed intrusion detection in Industrial Internet of Things. *Engineering Applications of Artificial Intelligence*, 135, 108826. https://doi.org/10.1016/j.engappai.2024.108826

Zhao, H., Liu, L., Fan, F., Zhang, H., & Ma, Y. (2024). An Adaptive Federated Learning Intrusion Detection System Based on Generative Adversarial Networks under the Internet of Things. *In Proceedings of the 2024 3rd Asia Conference on Algorithms, Computing and Machine Learning (CACML '24), Shanghai, China, April 26-28, 2024*, 1–6. https://doi.org/10.1145/3654823.3654824

Zukaib, U., Cui, X., Zheng, C., Liang, D., & Din, S. U. (2024). Meta-Fed IDS: Meta-learning and Federated learning based fog-cloud approach to detect known and zero-day cyber attacks in IoMT networks. *Journal of Parallel and Distributed Computing*, 192, 104934. https://doi.org/10.1016/j.jpdc.2024.104934

# Appendix A

# Literature Review Queries

The queries used to obtain the papers reviewed in the State of the Art chapter, during the Literature Review, are found below. Note that the queries for ScienceDirect and DBLP are much simpler and more general due to the limitations on query complexity and the number of logical operators.

## A.1 IEEE Xplore

```
("AI" OR "ML" OR "GenAI" OR "Artificial Intelligence"
OR "Machine Learning" OR "Generative" OR "LLM*"
OR "Large Language Model" OR "Transformer*"
OR "neural network*")
AND
("Abstract":"Federated" OR "Abstract":"Federated Learning"
OR "Abstract":"FL" OR "Abstract":"collaborative learning")
AND
("Network Security" OR "Network Traffic"
OR "threat detection" OR "attack detection" OR "defense")
AND
("Intrusion Detection" OR "Intrusion Detection System"
OR "IDS" OR "Anomaly Detection")
AND
("*IoT" OR "Internet of Things" OR "Edge")
```

## A.2 ACM Digital Library

```
[[All: "ai"] OR [All: "ml"] OR [All: "genai"] OR [All: "artificial intelligence"]
    OR [All: "machine learning"] OR [All: "generative"] OR [All: "llm*"] OR [All: "
    large language model"] OR [All: "transformer*"] OR [All: "neural network*"]]
AND
[[All: "network security"] OR [All: "network traffic"] OR [All: "threat detection"]
     OR [All: "attack detection"] OR [All: "defense"]]
AND
[[All: "intrusion detection"] OR [All: "intrusion detection system"] OR [All: "ids"
    ] OR [All: "anomaly detection"]]
AND
[[All: "*iot"] OR [All: "internet of things"] OR [All: "edge"]]
AND
[[Abstract: "federated"] OR [Abstract: "federated learning"] OR [Abstract: "fl"] OR
     [Abstract: "collaborative learning"]]
```

## A.3 Science Direct

```
(Artificial Intelligence) AND (Federated Learning OR FL OR collaborative learning)
    AND (Cybersecurity) AND (Intrusion Detection) AND (Internet of Things)
```

## A.4 DBLP

```
(federated learning  fl  collaborative learning) (security)
```

## A.5 Scopus

```
TITLE-ABS-KEY ( ( "AI" OR "ML" OR "GenAI" OR "Artificial Intelligence" OR "Machine
    Learning" OR "Generative" OR "LLM*" OR "Large Language Model" OR "Transformer*"
     OR "neural network*" ) AND ( "Federated" OR "Federated Learning" OR "FL" OR "
    collaborative learning" ) AND ( "Network Security" OR "Network Traffic" OR "
    threat detection" OR "attack detection" OR "defense" ) AND ( "Intrusion
    Detection" OR "Intrusion Detection System" OR "IDS" OR "Anomaly Detection" )
    AND ( "*IoT" OR "Internet of Things" OR "Edge" ) ) AND ( LIMIT-TO ( SUBJAREA ,
    "COMP" ) )
```