الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

جامعة غرداية
University of Ghardaia

Registration No.
.../.../.../.../...

كلية العلوم والتكنولوجيا
Faculty of Science and Technology
قسم الرياضيات والإعلام الآلي
Department of Mathematics and Computer Science

Thesis submitted in partial fulfillment of the requirements for the degree of

Master

Domain: Mathematics and Computer Science,
Field: Computer Science
Specialty: Intelligent Systems for Knowledge Extraction

Topic

A Federated Learning Framework for Collaborative and Privacy-Preserving IoT Security Attack Prediction

Presented by:
ADDOUN Mohammed
FERTAS Mouad

Publicly defended on 22/06/2026

Before the jury composed of:

Ms. Brahim Nacera	MAA	University of Ghardaia	President
Mr. FEKAIR Mohamed El Amine	MCB	University of Ghardaia	Supervisor
Mr. Guerrbouz Tahar	MCB	University of Ghardaia	Examiner

Academic Year: 2025/2026

ABSTRACT

The rapid expansion of the Internet of Things (IoT) has multiplied the attack surface exposed to cyber threats. Traditional intrusion detection systems rely on collecting network data at a central location, which raises serious privacy, bandwidth, and regulatory concerns in IoT environments where devices are heterogeneous, resource-constrained, and geographically distributed. Federated Learning (FL) offers a promising alternative by enabling devices to collaboratively train a shared model without sharing their raw data. This thesis investigates how a federated learning-based framework can be designed to enable distributed IoT devices to collaboratively predict security attacks, while preserving data privacy and handling device heterogeneity. We propose FedShield, a federated learning framework for multi-class IoT attack prediction, built around a lightweight deep learning classifier, Federated Averaging, and a mask-based secure aggregation mechanism. The framework is implemented in Python using TensorFlow and the Flower library, and evaluated on the TON_IoT dataset partitioned across non-IID clients. The experimental results show that the framework reaches strong detection performance on a non-IID partition of the TON_IoT dataset, and that mask-based secure aggregation can be added without measurable utility cost. The work confirms that federated learning is a credible building block for privacy-preserving IoT intrusion detection.

Keywords: Internet of Things; IoT Security; Intrusion Detection System; Federated Learning; Privacy-Preserving Mechanisms; TON_IoT.

RÉSUMÉ

L'expansion rapide de l'Internet des Objets (IoT) a multiplié la surface d'attaque exposée aux menaces informatiques. Les systèmes traditionnels de détection d'intrusions reposent sur la centralisation des données réseau, ce qui soulève des problèmes sérieux de confidentialité, de bande passante et de conformité réglementaire dans les environnements IoT, où les dispositifs sont hétérogènes, contraints en ressources et répartis géographiquement. L'apprentissage fédéré (Federated Learning, FL) offre une alternative prometteuse en permettant aux dispositifs d'entraîner collaborativement un modèle partagé sans partager leurs données brutes. Ce mémoire étudie comment un cadre basé sur l'apprentissage fédéré peut être conçu pour permettre à des dispositifs IoT distribués de prédire collaborativement les attaques de sécurité, tout en préservant la confidentialité des données et en prenant en compte l'hétérogénéité des dispositifs. Nous proposons FedShield, un cadre d'apprentissage fédéré pour la prédiction multi-classes des attaques IoT, construit autour d'un classifieur d'apprentissage profond léger, de l'algorithme Federated Averaging et d'un mécanisme d'agrégation sécurisée basé sur le masquage. Le cadre est implémenté en Python à l'aide de TensorFlow et de la bibliothèque Flower, et évalué sur le jeu de données TON_IoT réparti entre des clients non-IID. Les résultats expérimentaux montrent que le cadre atteint de fortes performances de détection sur une partition non-IID de TON_IoT, et que l'agrégation sécurisée basée sur le masquage peut y être intégrée sans coût mesurable en utilité. Ce travail confirme que l'apprentissage fédéré constitue une brique crédible pour la détection d'intrusions IoT préservant la vie privée.

Mots-clés : Internet des Objets ; Sécurité IoT ; Système de détection d'intrusions ; Apprentissage fédéré ; Mécanismes de préservation de la confidentialité ; TON_IoT.

الملخص

أدّى التوسّع السريع لإنترنت الأشياء (IoT) إلى تضاعف مساحة الهجوم المعرّضة للتهديدات السيبرانية. وتعتمد أنظمة كشف التسلّل التقليدية على جمع بيانات الشبكة في مكان مركزي، مما يطرح إشكاليات جدّية تتعلّق بالخصوصية وعرض النطاق الترددي والامتثال للأنظمة، خصوصًا في بيئات إنترنت الأشياء التي تتميّز بأجهزة غير متجانسة ومحدودة الموارد وموزّعة جغرافيًا. ويُقدّم التعلّم الموحّد (Federated Learning) بديلاً واعدًا يسمح للأجهزة بتدريب نموذج مشترك بصورة تعاونية دون مشاركة بياناتها الخام. تتناول هذه المذكرة كيفية تصميم إطار عمل قائم على التعلّم الموحّد يُمكِّن أجهزة إنترنت الأشياء الموزّعة من التنبّؤ التعاوني بهجمات الأمن، مع الحفاظ على خصوصية البيانات ومعالجة عدم تجانس الأجهزة. ونقترح إطار FedShield، وهو إطار للتعلّم الموحّد للتنبّؤ متعدّد الفئات بهجمات إنترنت الأشياء، مبنيٌّ على مصنّف خفيف للتعلّم العميق، وخوارزمية Federated Averaging، وآلية تجميع آمن قائمة على الإخفاء. وقد تمّ تنفيذ هذا الإطار بلغة Python باستخدام TensorFlow ومكتبة Flower، وتقييمه على مجموعة بيانات TON_IoT الموزّعة على عملاء بتوزيع غير متجانس. تُظهر النتائج التجريبية أن الإطار المقترح يحقّق أداءً قويًا في كشف الهجمات على توزيع غير متجانس لبيانات TON_IoT، وأنه يمكن دمج التجميع الآمن القائم على الإخفاء دون كلفة قابلة للقياس على الأداء. ويؤكّد هذا العمل أن التعلّم الموحّد يُمثّل لبنةً موثوقة لأنظمة كشف التسلّل في إنترنت الأشياء مع الحفاظ على الخصوصية.

الكلمات المفتاحية: إنترنت الأشياء؛ أمن إنترنت الأشياء؛ نظام كشف التسلّل؛ التعلّم الموحّد؛ آليّات الحفاظ على الخصوصية؛ TON_IoT.

DEDICATION

"To my beloved parents, whose unwavering love and sacrifices have been my greatest strength.

To my dear sisters, Lalla and Manel, for your endless encouragement and joy.

To my family, who always believed in me.

To my true friends, who stood by me through every challenge.

To my project partner, for our shared vision and teamwork.

To my teachers and colleagues from the SIEC 2025–2026 promotion, for your guidance and inspiration.

And to everyone who helped me along this journey, your support made it possible. Thank you from the bottom of my heart."

ADDOUN Mohammed

DEDICATION

I dedicate this thesis to my beloved parents for their endless love, sacrifices, prayers, and unconditional support throughout my journey.

To my brothers, thank you for your encouragement, motivation, and for always standing by my side.

To my family, I appreciate your support, understanding, and belief in me during every step of this journey.

Special thanks to my project partner for the teamwork, cooperation, and effort shared in completing this project successfully.

Finally, I would like to thank my teachers, mentors, and friends for their guidance, support, and encouragement, all of which played an important role in my growth and in the completion of this work.

FERTAS Mouad

ACKNOWLEDGMENTS

First and foremost, we would like to express our deepest gratitude and thanks to Almighty God for granting us the strength, patience, and perseverance needed to complete this work.

We would like to express our sincere gratitude to our supervisor, Dr. Fekair Mohamed El Amine, for his continuous guidance, valuable advice, and patient support throughout this research. His insightful remarks and constructive feedback played an essential role in shaping the direction and quality of this thesis.

We are also thankful to the members of the jury for the time and attention they have devoted to evaluating this work. Their comments and suggestions are a privilege that contributes to the improvement of this research.

Our sincere thanks go as well to the teaching staff of the Department of Mathematics and Computer Science at the University of Ghardaia, whose dedication and effort have accompanied our academic journey and provided us with the foundations on which this thesis is built.

Finally, we extend our heartfelt gratitude to our families for their unconditional support, patience, and encouragement, and to our friends and colleagues who, in many different ways, have contributed to the completion of this work.

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
AMQP	Advanced Message Queuing Protocol
ANN	Artificial Neural Network
API	Application Programming Interface
BLE	Bluetooth Low Energy
CNN	Convolutional Neural Network
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
CSV	Comma-Separated Values
DDoS	Distributed Denial-of-Service
DL	Deep Learning
DNN	Deep Neural Network
DNS	Domain Name System
DoS	Denial-of-Service
DP	Differential Privacy
DT	Decision Tree
FedAvg	Federated Averaging
FedNova	Federated Normalized Averaging
FedProx	Federated Proximal
FedSGD	Federated Stochastic Gradient Descent

FL	Federated Learning
FN	False Negative
FP	False Positive
FPR	False Positive Rate
FTL	Federated Transfer Learning
GDPR	General Data Protection Regulation
GRU	Gated Recurrent Unit
HE	Homomorphic Encryption
HFL	Horizontal Federated Learning
HIDS	Host-based Intrusion Detection System
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IID	Independent and Identically Distributed
IIoT	Industrial Internet of Things
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol version 6
k-NN	k-Nearest Neighbors
LoRaWAN	Long Range Wide Area Network
LPWAN	Low-Power Wide-Area Network
LSTM	Long Short-Term Memory
LTE-M	Long-Term Evolution for Machines
MITM	Man-in-the-Middle
ML	Machine Learning
MLP	Multi-Layer Perceptron
MQTT	Message Queuing Telemetry Transport

NaN	Not a Number
NB	Naive Bayes
NB-IoT	Narrowband Internet of Things
NIDS	Network-based Intrusion Detection System
OS	Operating System
REST	Representational State Transfer
ReLU	Rectified Linear Unit
RF	Random Forest
RFID	Radio-Frequency Identification
RNN	Recurrent Neural Network
RPL	Routing Protocol for Low-Power and Lossy Networks
SecAgg	Secure Aggregation
SMPC	Secure Multi-Party Computation
SQL	Structured Query Language
SSL/TLS	Secure Sockets Layer / Transport Layer Security
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TPR	True Positive Rate
VFL	Vertical Federated Learning
WSN	Wireless Sensor Network
XMPP	Extensible Messaging and Presence Protocol
XSS	Cross-Site Scripting

Introduction

Context

The Internet of Things (IoT) has become one of the most influential technological developments of the past two decades. Its scope extends far beyond traditional computing, connecting billions of physical objects, sensors, and embedded systems into a single global infrastructure [1]. These connected devices are now used in smart homes, healthcare, transportation, smart cities, and industrial automation, where they continuously generate and exchange large volumes of data. As a result, IoT has gradually transformed entire sectors of the economy and introduced new ways of monitoring, controlling, and automating physical environments.

However, this rapid expansion has also enlarged the attack surface. IoT devices are typically deployed in uncontrolled environments, have limited computational and memory resources, and are produced by many vendors with very different security practices. This combination makes them attractive and often easy targets for cyber-attacks. The Mirai botnet of 2016 [2], as well as many later incidents, showed how vulnerable IoT ecosystems can be exploited at scale to compromise services and infrastructures around the world. Protecting IoT systems against such threats has therefore become a major challenge in modern cybersecurity research.

Among the available defence mechanisms, intrusion detection systems (IDS) play a central role: they monitor traffic, detect malicious behaviour, and trigger appropriate responses. Over time, IDS approaches have evolved from rule-based and signature-based systems to data-driven systems that rely on machine learning (ML) and, more recently, deep learning (DL). These methods have shown strong performance on benchmark datasets, but they also share an important limitation: they usually require centralizing large amounts of network traffic on a single server or cloud platform. In an IoT context, this assumption is increasingly difficult to satisfy because of bandwidth and energy constraints and because the data generated by IoT devices often contains sensitive information about users and organizations.

Federated learning (FL) offers a promising alternative [3]. In a federated setting, multiple devices collaboratively train a shared model under the coordination of a central server while their raw data remain on the devices themselves. Only model updates are exchanged. This paradigm is especially attractive for IoT security because it fits the distributed nature of IoT environments and responds to the growing legal and ethical requirements related to

personal data protection. The present thesis is positioned at this intersection: it studies how federated learning can be designed and implemented to enable collaborative and privacy-preserving prediction of security attacks across distributed IoT devices.

Motivation

Several practical and scientific reasons motivate the work presented in this thesis. From a practical perspective, IoT devices have already become attractive entry points for attackers, and this trend is expected to intensify as new applications are deployed in critical sectors such as healthcare, transportation, and industry. Centralized intrusion detection approaches are not always able to follow this trajectory: collecting traffic from millions of devices and storing it in a single location is costly in bandwidth and storage, raises serious privacy concerns, and is increasingly constrained by data-protection regulations such as the European General Data Protection Regulation.

From a scientific perspective, federated learning has matured rapidly since its introduction and has generated a growing body of work dedicated to IoT security. As discussed in Chapter 2, several recent proposals show that federated learning can reach detection accuracies comparable to those of centralized models while keeping data on the devices. However, most existing works address only one dimension of the problem at a time. Some focus on accuracy on a specific dataset, others study communication efficiency, and only a few explicitly examine the trade-off between model utility and the formal privacy guarantees that protect what leaves each device. Moreover, the heterogeneity of IoT environments, where data distributions and device capabilities can vary significantly across clients, is rarely studied together with privacy.

These observations motivate the development of an integrated framework that combines federated learning with a privacy-preserving aggregation layer in a way that is realistic for IoT and that explicitly considers the heterogeneity of the data observed by different clients. Rather than introducing a new cryptographic primitive or a radically new model, this work focuses on bringing together known and validated components into a single coherent system and on evaluating its behaviour on a recent and realistic IoT security dataset.

Research Questions

The general research question that guides this thesis is the following:

How can a federated learning-based framework be designed to enable distributed IoT devices to collaboratively predict security attacks, while preserving data privacy through mask-based secure aggregation and handling device heterogeneity, without requiring the sharing of raw network traffic data?

From this main question, several more specific sub-questions are derived in order to structure the analytical and experimental parts of the thesis. The first sub-question concerns

the feasibility of federated learning itself in this context: to what extent can a Federated-Averaging-based configuration reach strong multi-class detection performance on a realistic IoT security dataset under non-IID client data? The second sub-question focuses on data heterogeneity, which is a defining characteristic of real IoT deployments: how does the framework behave under non-IID data distributions across clients, where each client observes a different mixture of normal and attack traffic, and to what extent does this heterogeneity affect the convergence and the per-class detection performance of the collaboratively trained model? The third sub-question concerns privacy: can a mask-based secure-aggregation layer be integrated into the federated pipeline without measurable loss in detection performance under the honest-but-curious server threat model adopted in this thesis? The fourth sub-question concerns the deployability of the framework on resource-constrained IoT devices: can a lightweight neural classifier achieve strong detection performance while keeping the computational and communication cost compatible with the constraints of typical IoT hardware? The fifth and final sub-question concerns reproducibility and realism: can the proposed pipeline be implemented under a production-grade federated learning framework, so that it remains compatible with realistic deployment scenarios beyond the simulation environment?

Objectives and Scope

The general objective of this thesis is to design, implement, and experimentally evaluate a federated learning framework dedicated to the collaborative and privacy-preserving prediction of security attacks in IoT environments. To reach this objective, the work pursues three more specific goals.

First, the thesis aims to consolidate the conceptual foundations of the problem by reviewing the main notions of IoT security, machine learning, deep learning, and federated learning, and by surveying the state of the art in IoT attack prediction. The purpose of this part is to position the contributions of the work within an existing scientific landscape and to identify the open challenges that motivate the proposed framework.

Second, the thesis aims to design a federated learning framework that is realistic for IoT, that supports multi-class attack prediction, and that integrates a mask-based secure-aggregation mechanism in a coherent way. The design choices are driven by the constraints of IoT environments, in particular the heterogeneity of clients, the distribution of data across devices, and the need to keep raw network traffic local.

Third, the thesis aims to validate the proposed framework empirically. The experimental study is conducted on the TON_IoT dataset [4], which is one of the most representative public benchmarks for IoT and Industrial IoT intrusion detection. The framework is evaluated under a non-IID client partition in a simulated federated setting, with particular attention to the effect of its mask-based secure-aggregation layer. The scope of the work is therefore deliberately focused on network-level attack prediction based on flow features, on simulated federations with a moderate number of clients, and on a single privacy-oriented aggregation mechanism. Aspects such as Byzantine robustness against malicious clients, fully crypto-

graphic secure-aggregation protocols, and real-world deployment on physical IoT devices are considered out of scope and discussed as perspectives for future work.

Contributions of the Thesis

The work presented in this thesis brings several contributions to the field of federated learning-based intrusion detection for the Internet of Things. These contributions span the conceptual, methodological, and experimental dimensions of the problem, and together they form a coherent answer to the research questions formulated above.

The first contribution is a structured review of the state of the art in IoT intrusion detection, organized along four progressive categories and synthesised through comparative tables and a taxonomy of federated learning-based intrusion detection systems. This review identifies the open challenges that motivate the rest of the work.

The second contribution is the design of a federated learning framework, named FedShield, dedicated to collaborative and privacy-preserving IoT attack prediction. The framework integrates a lightweight deep learning classifier, Federated Averaging, and a mask-based secure-aggregation mechanism in a way that explicitly takes into account the heterogeneity of client data and the resource constraints of IoT devices.

The third contribution is the implementation and the experimental validation of the framework on a realistic IoT security dataset. The framework is evaluated under a non-IID partitioning of the data across simulated clients, which makes it possible to assess its behaviour in a realistic federated setting for IoT intrusion detection.

The fourth contribution empirically verifies that the mask-based secure aggregation layer can be integrated into the federated pipeline without measurable loss in detection performance. This result directly addresses one of the main privacy challenges identified in the surveyed literature.

Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 1, Foundations of IoT Security and Federated Learning, introduces the theoretical foundations on which the work is built. It presents the Internet of Things and its security challenges, the main types of IoT attacks and intrusion detection systems, the fundamentals of machine learning and deep learning, and a detailed presentation of federated learning, including its architecture, types, aggregation algorithms, and privacy-preserving mechanisms.

Chapter 2, State of the Art, reviews the existing literature on IoT attack prediction. It first describes the main benchmark datasets and evaluation metrics, then organizes the literature into four progressive categories (classical, machine learning-based, deep learning-based,

and federated learning-based approaches) and concludes with a comparative discussion of the main open challenges.

Chapter 3, Proposed FedShield Framework, presents the conceptual and architectural design of FedShield. It describes the framework architecture, the local model, the federated training procedure, and the privacy-preserving mechanism before any experimental evaluation.

Chapter 4, Implementation and Experimental Evaluation, describes the implementation environment, the TON_IoT dataset and preprocessing pipeline, the experimental results, and the comparative discussion with related federated IDS studies.

The final chapter, General Conclusion, summarizes the main contributions of the thesis, discusses its limitations, and identifies several directions for future work, in particular the integration of more advanced aggregation algorithms, more sophisticated privacy-preserving protocols, and the deployment of the framework on real IoT hardware.

Chapter 1

Foundations of IoT Security and Federated Learning

1.1 Introduction

The rapid expansion of the Internet of Things (IoT) has transformed the way people, devices, and systems interact. Billions of connected objects, ranging from smart home appliances and wearable health monitors to industrial sensors and autonomous vehicles, now generate and exchange data continuously over global networks [1]. While this connectivity enables unprecedented automation and efficiency, it also introduces serious security concerns. IoT devices are often resource-constrained, deployed in uncontrolled environments, and exposed to a wide variety of cyber-attacks, which makes them a highly attractive target for malicious actors.

Traditional security solutions, such as signature-based intrusion detection systems, are no longer sufficient to cope with the scale, diversity, and evolving nature of IoT threats. Consequently, researchers have turned to machine learning (ML) and deep learning (DL) methods to build more adaptive and intelligent attack prediction systems [5]. However, these approaches typically rely on centralizing data at a single server or cloud platform, which raises privacy concerns and creates performance bottlenecks. To address these limitations, Federated Learning (FL) has recently emerged as a promising paradigm that enables multiple devices to collaboratively train a shared model without exchanging their raw data [3].

This chapter introduces the fundamental concepts that form the basis of our work. We first present an overview of the Internet of Things, its architecture, applications, and security challenges. We then describe the main categories of IoT attacks and the role of intrusion detection systems. The next section reviews the essential notions of machine learning and deep learning, with a focus on the techniques most relevant to security analytics. The chapter ends with a detailed presentation of federated learning, its architecture, types, aggregation algorithms, and privacy-preserving mechanisms.

1.2 Internet of Things (IoT)

1.2.1 Definition and Characteristics

The Internet of Things extends the reach of the Internet beyond traditional computing devices to include physical objects encountered in everyday life. The term was introduced in the context of supply-chain management through Radio-Frequency Identification (RFID) technology [6]. Since then, the concept has evolved substantially with the emergence of low-cost sensors, low-power communication protocols, and cloud computing platforms.

IoT is defined as a global infrastructure for the information society that enables advanced services by interconnecting physical and virtual things through interoperable information and communication technologies [7]. In practical terms, IoT refers to a network of connected devices capable of collecting, exchanging, and acting upon data with minimal human intervention [1].

The main characteristics that distinguish IoT from traditional networked systems are:

- Heterogeneity: IoT devices differ widely in terms of hardware capabilities, operating systems, communication protocols, and data formats.
- Large scale: The number of connected devices is expected to exceed tens of billions, generating massive volumes of data.
- Resource constraints: Many IoT devices have limited computational power, memory, battery life, and bandwidth.
- Dynamic environment: Devices frequently join, leave, or change location within the network.
- Context-awareness: IoT systems often rely on sensing the environment to provide personalized or adaptive services.

Figure 1 represents the main components of a generic IoT ecosystem and shows how connected devices, gateways, and cloud services interact. This visual overview helps connect the definition of IoT with the architectural layers discussed next.

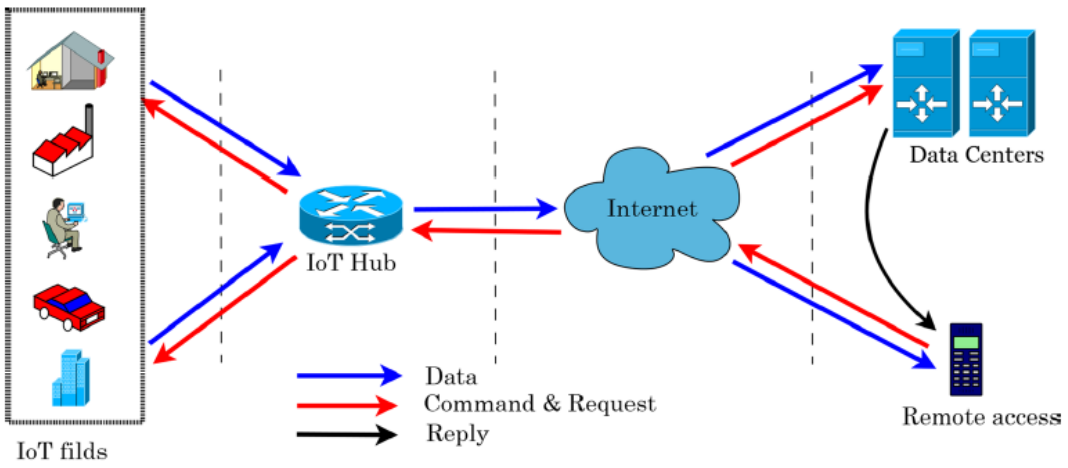


Figure 1. Conceptual IoT ecosystem (adapted from [8]).

1.2.2 IoT Architecture

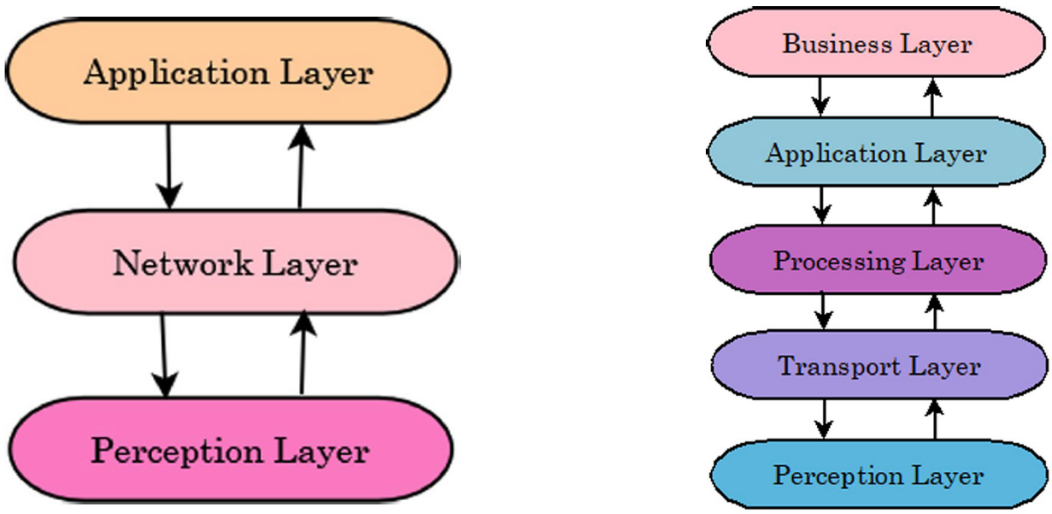
Although no single reference architecture is universally adopted, most IoT systems are described through a layered model. The most common representations include three, four, or five layers [9]. We present here the widely used three-layer architecture and briefly mention its extensions.

Perception Layer (Sensing Layer). This is the lowest layer, responsible for interacting with the physical world. It comprises sensors, actuators, RFID tags, and other data acquisition devices. The main function of this layer is to collect data about the environment (such as temperature, humidity, motion, or location) and to perform basic actions through actuators.

Network Layer. This layer handles the transmission of data between the perception layer and upper layers. It relies on a variety of wired and wireless communication technologies, including Wi-Fi, Bluetooth, Zigbee, LoRa, 4G/5G, and the Internet. The network layer is also responsible for routing, addressing, and ensuring reliable data delivery.

Application Layer. This is the topmost layer, where data is processed, analyzed, and delivered to end users through specific applications and services. Examples include smart home platforms, healthcare monitoring systems, and industrial dashboards.

More detailed architectures add two intermediate layers: the middleware layer, which manages device discovery, data aggregation, and service composition, and the business layer, which handles the overall management of the IoT system, including business models and user privacy [9]. As illustrated in Figure 2, the five-layer model refines the basic three-layer view by separating processing and management functions more explicitly.



(a) Three-layer IoT architecture. (b) Five-layer IoT architecture.

Figure 2. Three-layer and five-layer IoT architectures (adapted from [8]).

1.2.3 IoT Application Domains

IoT has penetrated nearly every sector of modern life. Among its most prominent application domains, we mention:

- Smart homes: Connected thermostats, lighting systems, security cameras, and voice assistants improve comfort, energy efficiency, and home security.
- Healthcare: Wearable devices and remote monitoring systems enable continuous tracking of vital signs, early detection of health problems, and personalized treatment.
- Smart cities: IoT is used to optimize traffic flow, manage waste collection, monitor air quality, and enhance public safety.
- Industrial IoT (IIoT): Sensors and actuators deployed in factories enable predictive maintenance, real-time monitoring, and automation of production lines.
- Agriculture: Smart farming uses soil sensors, drones, and weather stations to optimize irrigation, fertilization, and crop yield.
- Transportation: Connected vehicles, traffic sensors, and logistics tracking systems improve mobility, safety, and supply chain efficiency.

1.2.4 IoT Communication Protocols

Communication protocols form the backbone of any IoT system, as they define how devices exchange data with each other and with gateways or cloud servers. Because of the diversity

of IoT applications and the strict resource constraints of end devices, no single protocol fits all scenarios. Instead, IoT ecosystems rely on a layered stack of protocols that can be broadly classified according to their role in the communication process [9].

Infrastructure and network protocols. At the network layer, IPv6 and its lightweight adaptation for constrained devices, 6LoWPAN, enable end-to-end addressing of billions of devices. Routing is often handled by RPL (Routing Protocol for Low-Power and Lossy Networks), which is specifically designed for networks of small, battery-powered nodes.

Short-range wireless protocols. For local communication, IoT devices rely on a range of wireless technologies. IEEE 802.15.4 underpins low-power personal area networks and is used by standards such as Zigbee and Thread. Bluetooth Low Energy (BLE) provides short-range, low-energy communication typical of wearables and smart home sensors. Wi-Fi remains the most common choice for devices that require higher bandwidth, such as IP cameras and smart speakers.

Long-range and cellular protocols. For wide-area deployments, Low-Power Wide-Area Network (LPWAN) technologies such as LoRaWAN, Sigfox, and NB-IoT enable communication over several kilometres with very low energy consumption. Cellular technologies (4G LTE-M, 5G) are increasingly used for mobile or bandwidth-intensive IoT applications, including connected vehicles and industrial monitoring.

Application-layer protocols. At the top of the stack, several lightweight messaging protocols have been designed for IoT. The Message Queuing Telemetry Transport (MQTT) protocol uses a publish–subscribe model and is widely adopted in smart home and industrial applications. The Constrained Application Protocol (CoAP) offers a REST-like interaction model optimized for constrained devices. The Advanced Message Queuing Protocol (AMQP) and the Extensible Messaging and Presence Protocol (XMPP) are also used in certain IoT scenarios, especially when interoperability with existing enterprise systems is required.

From a security perspective, the diversity of IoT protocols is a double-edged sword. On the one hand, it allows designers to choose the most appropriate technology for each use case. On the other hand, it multiplies the attack surface because each protocol has its own weaknesses, and many IoT implementations disable encryption or authentication to save resources. Attacks such as MQTT topic hijacking, CoAP amplification, and the exploitation of weak BLE pairing mechanisms illustrate the operational impact of these vulnerabilities. The federated learning framework proposed in this thesis is designed to be protocol-agnostic because it operates on traffic features extracted from devices regardless of the underlying communication technology. Figure 3 shows the layered protocol stack typically encountered in IoT deployments and clarifies where these protocol-level risks arise.

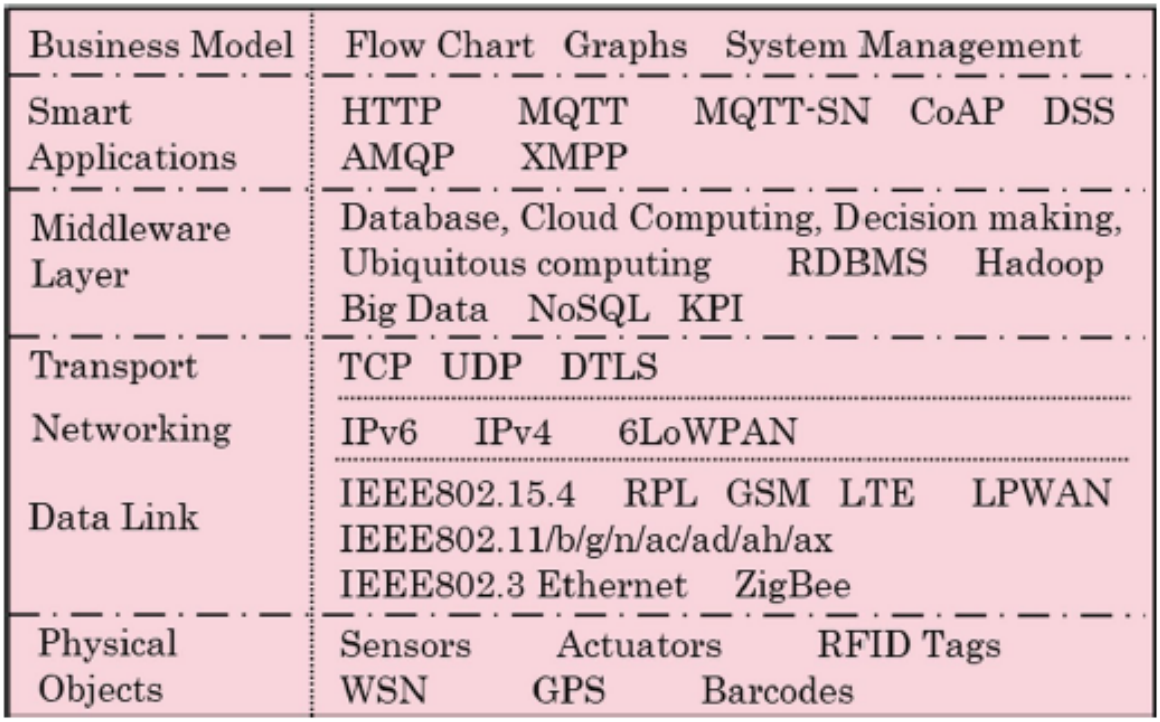


Figure 3. IoT communication protocol stack (adapted from [8]).

1.2.5 IoT Security Requirements and Challenges

Security in IoT is not simply an extension of traditional network security. The unique characteristics of IoT, namely heterogeneity, scale, and resource constraints, introduce specific requirements and challenges [10].

The main security requirements of IoT systems can be summarized as follows:

- Confidentiality: Sensitive data collected or transmitted by IoT devices must be protected against unauthorized disclosure.
- Integrity: Data must not be altered during transmission or storage without detection.
- Availability: IoT services must remain accessible to legitimate users, even in the presence of attacks such as denial of service.
- Authentication: Devices and users must be correctly identified before being granted access to the system.
- Authorization: Access to resources must be controlled according to defined policies.
- Privacy: Personal and sensitive information must be protected against inappropriate collection, use, or sharing.

Despite these requirements, IoT systems face several security challenges:

- Limited computational resources make it difficult to deploy strong cryptographic protocols on small devices.
- Heterogeneity complicates the design of uniform security mechanisms across different platforms and vendors.
- Large attack surface: The enormous number of devices and communication links increases the number of potential entry points for attackers.
- Lack of standardization in security practices leads to inconsistent protection levels.
- Long device lifespan means that many devices remain in operation for years without firmware updates or patches.
- Physical exposure: Many IoT devices are deployed in public or unmonitored locations, making them vulnerable to physical tampering.

These challenges motivate the need for advanced, adaptive, and scalable security solutions, which is the main focus of this thesis.

1.3 IoT Security Attacks

1.3.1 Attack Classification

IoT attacks can be classified according to several criteria, such as the target layer of the IoT architecture, the attack objective, or the attacker's level of access [11]. A common classification distinguishes attacks based on the architectural layer they target:

- Perception-layer attacks target physical devices and sensors. Examples include node tampering, malicious code injection, and eavesdropping on sensor data.
- Network-layer attacks exploit vulnerabilities in communication protocols and routing mechanisms. Typical examples are denial of service (DoS), unwanted scanning attacks, and Sybil attacks.
- Application-layer attacks target software applications and services, including code injection, malicious scripts, and unauthorized access.

Another common classification separates attacks into passive attacks (the attacker only observes the system, for instance through eavesdropping) and active attacks (the attacker modifies or disrupts the system's behaviour).

1.3.2 Common IoT Attack Types

We describe below some of the most frequent and impactful attacks observed in IoT environments.

Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks. In these attacks, the attacker overwhelms the target device or service with a large volume of requests or traffic, causing it to become unavailable to legitimate users. In a DDoS attack, the attack is launched from multiple compromised devices (often IoT devices themselves). DDoS attacks are particularly dangerous in IoT due to the huge number of vulnerable devices that can be recruited into botnets [2].

Man-in-the-Middle (MITM) attacks. The attacker secretly intercepts and possibly modifies communications between two parties who believe they are directly communicating with each other. In IoT, MITM attacks are facilitated by the use of unsecured wireless communication channels and weak authentication.

Botnet attacks. A botnet is a network of compromised devices controlled remotely by an attacker. The Mirai botnet, which first appeared in 2016, infected hundreds of thousands of IoT devices and was used to launch massive DDoS attacks against major Internet services [2, 12]. Variants of Mirai and similar botnets continue to pose a major threat to IoT ecosystems.

Sybil attacks. The attacker creates multiple fake identities (Sybil nodes) to gain a disproportionate influence on the network. Sybil attacks can disrupt routing, voting, and data aggregation mechanisms.

Sinkhole and wormhole attacks. In a sinkhole attack, a malicious node attracts traffic from its neighbours and then drops or manipulates it. In a wormhole attack, two colluding nodes create a tunnel to transmit packets from one location in the network to another, bypassing normal routing.

Replay attacks. The attacker captures legitimate messages and retransmits them later to deceive the receiver into performing unauthorized actions.

Malware and ransomware. Malicious software can infect IoT devices, steal data, or render them inoperable until a ransom is paid.

Data injection and spoofing attacks. The attacker injects false data into the network or impersonates a legitimate device to mislead the system. Figure 4 presents a high-level taxonomy of these common IoT attack categories and prepares the transition to intrusion detection mechanisms.

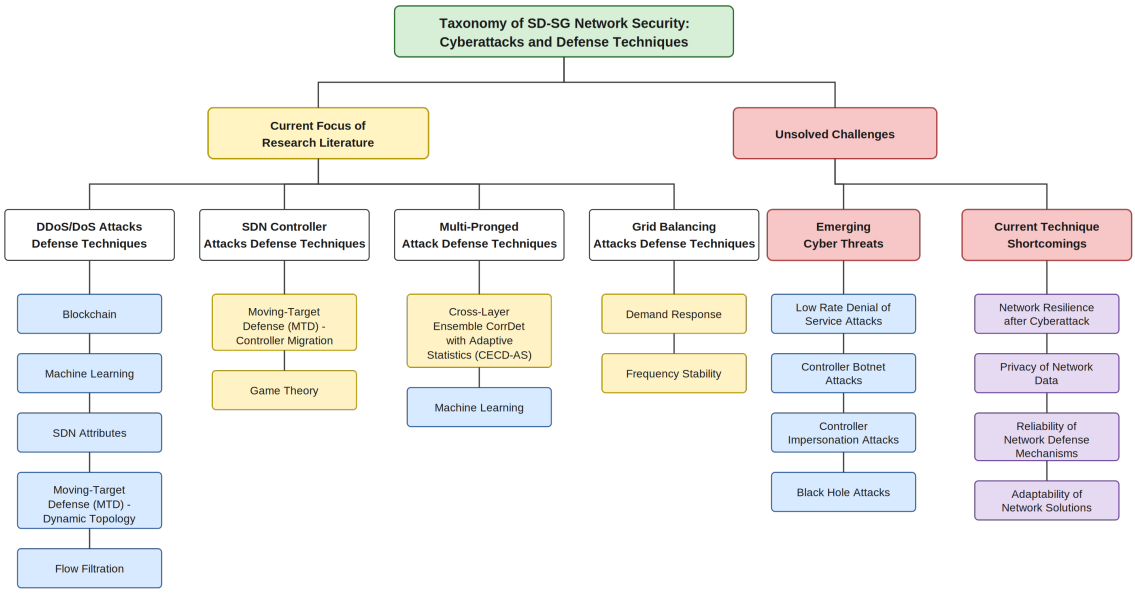


Figure 4. Taxonomy of common IoT security attacks (adapted from [54]).

1.3.3 Intrusion Detection Systems (IDS)

An Intrusion Detection System is a security mechanism designed to detect unauthorized or malicious activities within a network or a host. In the IoT context, IDSs play a critical role in identifying attacks in real time and triggering appropriate responses [13].

IDSs are generally categorized along two main axes: the detection method and the deployment strategy.

Detection methods. Two main approaches exist:

- Signature-based detection relies on a database of known attack patterns (signatures). Incoming traffic is compared against these signatures, and a match triggers an alert. This approach achieves high accuracy for known attacks but fails to detect new or zero-day attacks.
- Anomaly-based detection builds a model of normal system behaviour and flags any deviation as a potential attack. This approach can detect previously unseen attacks but typically produces more false positives.

Hybrid approaches combining both methods also exist and aim to balance detection accuracy and false-alarm rates.

Deployment strategies. IDSs can be deployed at different locations:

- Host-based IDS (HIDS): installed on individual devices, monitoring their local activities.
- Network-based IDS (NIDS): deployed at strategic points in the network to monitor traffic.

- Distributed IDS: composed of multiple cooperating nodes that share information to improve detection.

With the growing complexity of IoT attacks, machine learning and deep learning have become central to the design of modern anomaly-based IDSs, as discussed in the following sections.

1.4 Machine Learning and Deep Learning Fundamentals

1.4.1 Machine Learning Overview

Machine learning is a subfield of artificial intelligence concerned with the development of algorithms that allow computers to learn from data without being explicitly programmed for every task. A classic definition describes it as the field of study that gives computers the ability to learn without being explicitly programmed [14].

In practice, a machine learning system is trained on a set of examples (the training data) and produces a model that can make predictions or decisions on new, unseen data. The quality of the model is evaluated on a separate set of examples (the test data) using appropriate performance metrics [15].

Machine learning has been successfully applied in many domains, including image recognition, natural language processing, medical diagnosis, financial forecasting, and, of course, cybersecurity [16].

1.4.2 Supervised, Unsupervised, and Reinforcement Learning

Machine learning algorithms are traditionally classified into three main categories, depending on the type of data and the learning signal used.

Supervised learning. The training data consists of pairs of inputs and their corresponding expected outputs (labels). The algorithm learns a mapping from inputs to outputs, which it can then apply to new inputs. Two main tasks fall under this category:

- Classification, where the output is a discrete class label (for example, “normal” vs. “attack”).
- Regression, where the output is a continuous value (for example, predicting a sensor reading).

Popular supervised algorithms include Decision Trees, Random Forests, Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Neural Networks [15].

Unsupervised learning. The training data consists only of inputs, without any labels. The algorithm seeks to discover hidden structures or patterns in the data. Typical tasks include:

- Clustering, which groups similar examples together (for example, k-means, DBSCAN).
- Dimensionality reduction, which projects data into a lower-dimensional space (for example, Principal Component Analysis).
- Anomaly detection, which identifies examples that deviate from normal patterns, a task particularly relevant to intrusion detection.

Reinforcement learning. An agent interacts with an environment and learns through trial and error. It receives rewards or penalties based on its actions and aims to maximize the cumulative reward over time. Reinforcement learning has been applied to game playing, robotics, and autonomous systems [17]. Figure 5 summarizes the major machine-learning paradigms discussed in this subsection and highlights the place of supervised attack classification within the broader ML landscape.

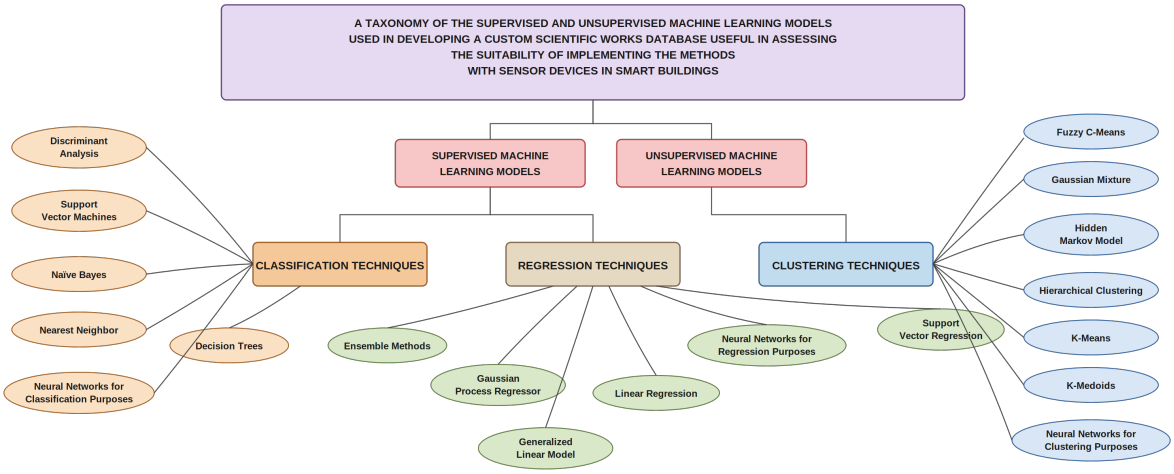


Figure 5. Taxonomy of machine-learning paradigms (adapted from [55]).

1.4.3 Artificial Neural Networks

An Artificial Neural Network (ANN) is a computational model inspired by the biological neural networks of the human brain. It consists of interconnected processing units called neurons, organized in successive layers: an input layer, one or more hidden layers, and an output layer [18]. Each artificial neuron receives a set of inputs, computes a weighted sum, adds a bias term, and applies a non-linear activation function to produce the output given in (1):

$$y = f\left(\sum_i w_i x_i + b\right) \tag{1}$$

where x_i are the inputs, w_i are the weights, b is the bias, and f is the activation function. Common activation functions include the sigmoid, the hyperbolic tangent (tanh), and the Rectified Linear Unit (ReLU).

The learning process consists of adjusting the weights and biases to minimize a loss function that measures the discrepancy between the network's predictions and the expected outputs. This is typically achieved through the backpropagation algorithm [19], which computes the gradient of the loss with respect to each parameter, combined with an optimization method such as stochastic gradient descent.

Neural networks with a single hidden layer are called shallow, while networks with multiple hidden layers are called deep and form the basis of deep learning [5].

1.4.4 Deep Learning Architectures

Deep learning has revolutionized many areas of artificial intelligence by achieving state-of-the-art performance in tasks such as image classification, speech recognition, and natural language understanding. Its success is largely attributed to the ability of deep neural networks to automatically learn hierarchical representations of data [5, 20]. Several specialized architectures have been proposed for different types of data.

Multilayer Perceptron (MLP). The MLP is the simplest form of deep feedforward neural network. It consists of several fully connected layers, each composed of multiple neurons. MLPs are suitable for tasks involving structured or tabular data, such as network traffic features extracted from IoT traffic flows. They serve as a baseline architecture in many intrusion detection studies.

Convolutional Neural Networks (CNN). CNNs are designed to process data with a grid-like topology, such as images or time-series signals [21]. A typical CNN architecture is composed of convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply learnable filters that detect local patterns, while pooling layers reduce the spatial dimensions of the feature maps. CNNs have been successfully applied to intrusion detection by treating network traffic features as matrices or images.

Recurrent Neural Networks (RNN). RNNs are designed to process sequential data by maintaining an internal state (memory) that captures information about previous inputs. This makes them well-suited for modeling time-dependent data such as network traffic sequences. However, standard RNNs suffer from the vanishing and exploding gradient problems, which limit their ability to learn long-term dependencies.

Long Short-Term Memory (LSTM). LSTMs are a special type of RNN that address the limitations of standard RNNs by introducing a gating mechanism [22]. Each LSTM cell contains three gates, input, forget, and output, which control the flow of information and allow the network to retain relevant information over long sequences. LSTMs have been widely used for sequence-based intrusion detection.

Gated Recurrent Units (GRU). GRUs are a simplified variant of LSTMs that combine the input and forget gates into a single update gate [23]. They offer comparable performance to LSTMs with fewer parameters and faster training.

Autoencoders. An autoencoder is an unsupervised neural network trained to reconstruct

its input at its output through a compressed internal representation. Autoencoders are commonly used for dimensionality reduction and anomaly detection; samples that produce high reconstruction errors are flagged as anomalies [20]. Figure 6 compares the main deep-learning architectures used in the literature and shows why different model families are suited to different traffic representations.

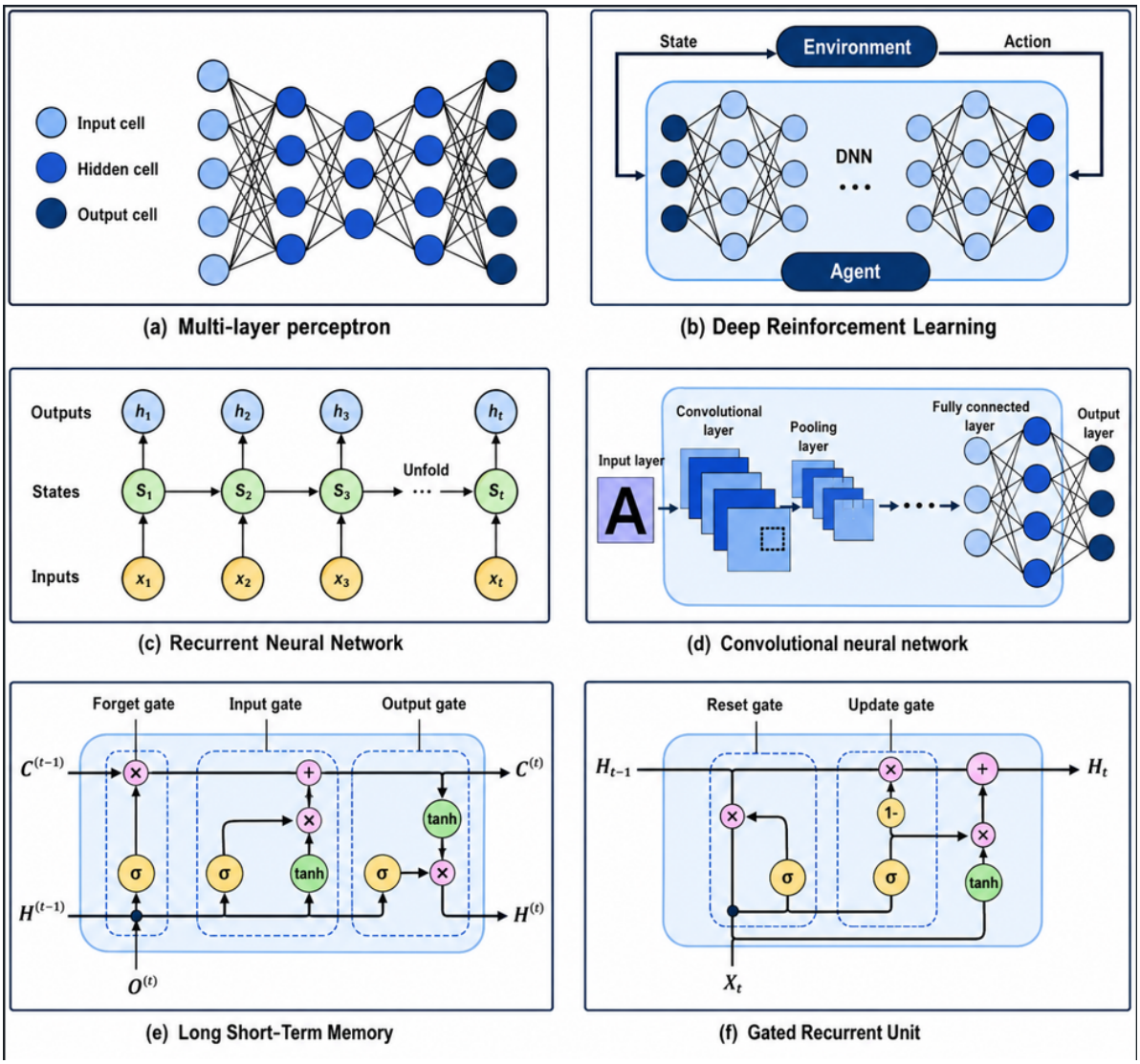


Figure 6. Major deep-learning architectures (adapted from [56]).

Despite their effectiveness, traditional deep-learning models usually require a large amount of data to be centralized at a single location for training. This centralized paradigm raises serious concerns in the IoT context, where data are inherently distributed across many devices and often contain sensitive information. Federated learning has been proposed to address these concerns.

1.5 Federated Learning

1.5.1 Definition

Federated learning is a distributed machine-learning paradigm in which multiple clients, such as IoT devices, smartphones, or organizations, collaboratively train a shared model under the coordination of a central server without sharing their raw data [3]. The original formulation was presented in the context of training language models on mobile devices.

The motivations behind federated learning are multiple:

- Privacy preservation: Raw data never leaves the client's device, which significantly reduces the risk of data leakage.
- Reduced communication cost: Instead of transmitting large volumes of raw data to a central server, only model updates (weights or gradients) are exchanged.
- Compliance with regulations: Data protection laws such as the European General Data Protection Regulation (GDPR) restrict the centralization of personal data. FL offers a natural way to comply with such regulations [24].
- Exploitation of distributed data: Many real-world datasets are inherently distributed and cannot be easily centralized for practical, legal, or ethical reasons.

These advantages make federated learning particularly attractive for IoT security, where data is distributed across millions of devices and often contains sensitive information about users' activities [25].

1.5.2 Federated Learning Architecture

A typical federated learning system consists of two main components: a central server (also called aggregator) and a set of clients. The training process proceeds in successive communication rounds. In each round, the following steps are performed:

- Initialization: The server initializes a global model with random or pre-trained weights.
- Client selection: At each round, the server selects a subset of available clients to participate in training.
- Model distribution: The server sends the current global model to the selected clients.
- Local training: Each selected client trains the received model on its local data for one or more epochs, producing an updated local model.
- Model update transmission: Each client sends its local model update (weights or gradients) back to the server.
- Aggregation: The server aggregates the received updates to produce a new global model.

- Iteration: The previous steps are repeated until convergence or until a predefined number of rounds is reached.

Figure 7 shows this client–server workflow. It illustrates how local training, model-update transmission, and server-side aggregation form a repeated learning cycle without centralizing raw data.

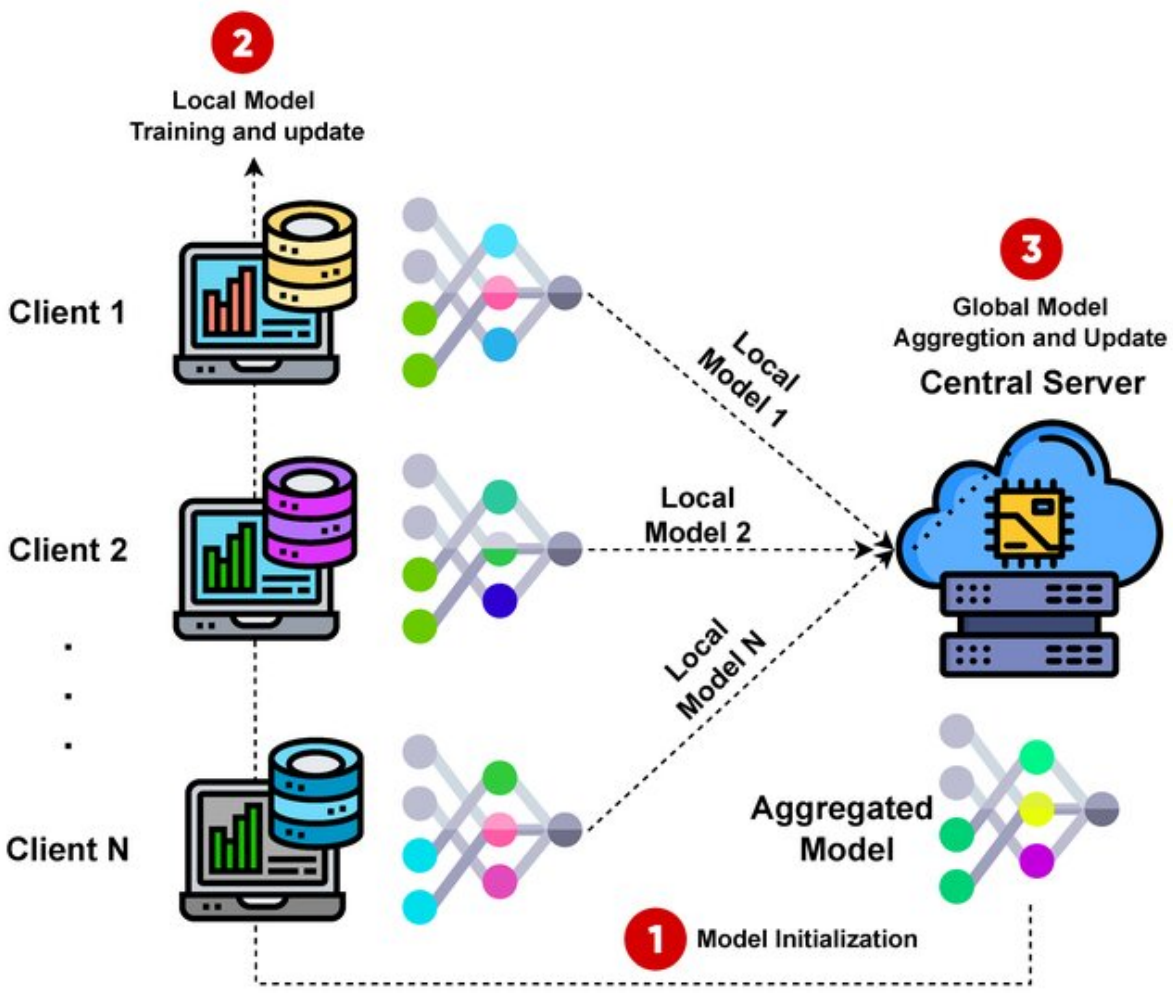


Figure 7. Client–server federated learning architecture (adapted from [24]).

This process allows the global model to benefit from the collective knowledge of all clients without ever accessing their raw data [3].

1.5.3 Types of Federated Learning

Federated learning systems can be classified according to the way data is partitioned across clients. Three main categories are usually distinguished [24].

Horizontal Federated Learning (HFL). The datasets held by different clients share the

same feature space but contain different samples. This is the most common setting and corresponds, for example, to the case of multiple IoT devices collecting similar types of network traffic data but from different users or locations.

Vertical Federated Learning (VFL). The datasets held by different clients share the same sample space but contain different features. This setting is typical of collaborations between organizations that hold complementary information about the same users (for example, a bank and an e-commerce platform).

Federated Transfer Learning (FTL). This is used when the datasets differ in both sample and feature spaces. Transfer learning techniques are applied to bridge the gap between the different domains.

In the context of IoT security attack prediction, horizontal federated learning is the most relevant setting, since different IoT devices or gateways typically observe similar types of features from different subsets of traffic. Figure 8 represents the three main types of federated learning and clarifies why the horizontal setting matches the data organization used in this thesis.

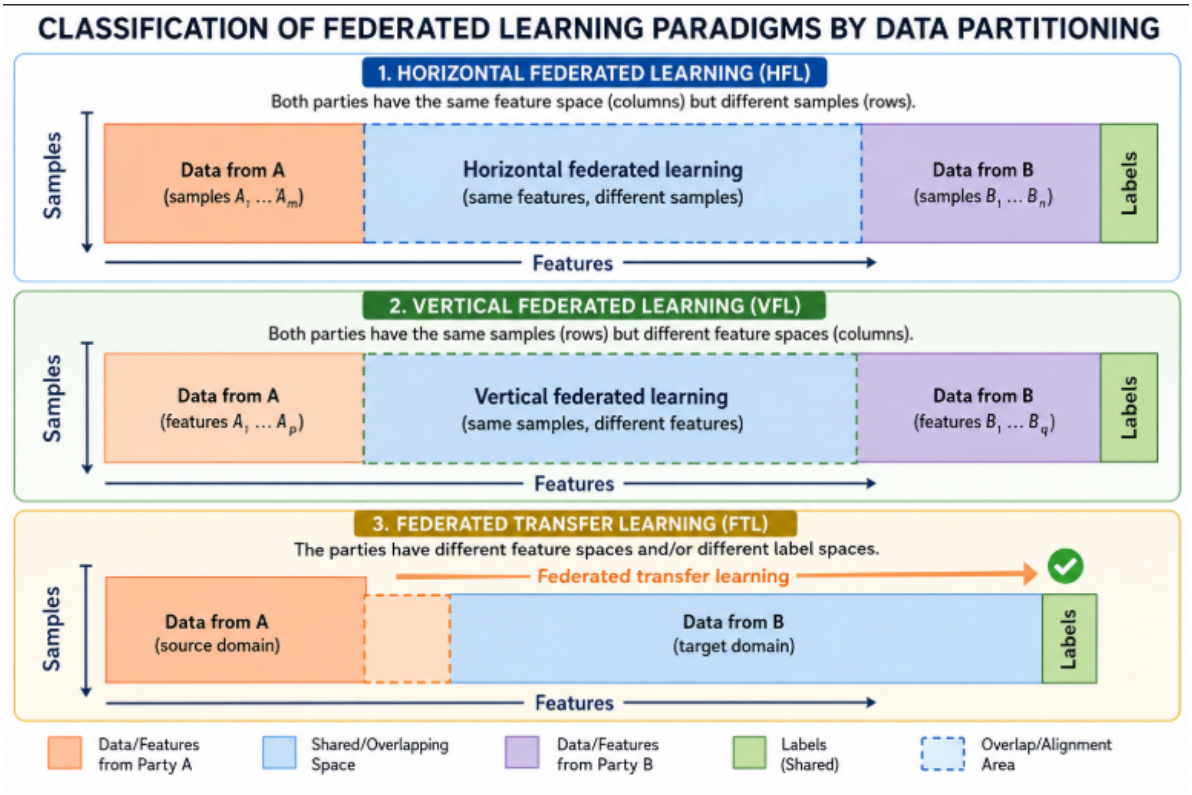


Figure 8. Main types of federated learning (adapted from [24]).

1.5.4 Data Distribution Across Clients: IID and Non-IID Settings

A key assumption that separates federated learning from classical distributed machine learning is the way training data is distributed across clients. In the simplest theoretical setting, each client draws its samples independently from the same underlying probability distribution. This is known as the *independent and identically distributed* (IID) setting, and many convergence results in distributed optimization are derived under this assumption [3, 26].

In practical federated learning deployments, this assumption is rarely satisfied. Clients usually represent different users, devices, or geographic locations, so their local datasets naturally reflect different behaviours and environments. Such partitions are referred to as *non-IID* data, meaning that the data distribution observed by one client can differ from the distributions observed by other clients. The literature commonly distinguishes three forms of non-IID behaviour [26–28]. Label skew occurs when clients hold samples from only some classes, or when class proportions differ strongly from one client to another. Feature skew occurs when the distribution of input features varies across clients, for example because different sensors or deployment environments produce systematically different readings. Quantity skew occurs when clients hold unequal numbers of samples, which can make some clients contribute more strongly than others during aggregation.

The non-IID nature of client data is one of the main challenges in federated learning. It can slow convergence, destabilize training, and bias the global model toward dominant clients. These effects motivated aggregation algorithms designed for heterogeneous settings, such as FedProx [29] and FedNova [30], which are discussed in Section 1.5.5.

1.5.5 Aggregation Algorithms

The aggregation step is a core component of federated learning, as it determines how the local updates from different clients are combined into a new global model. Several aggregation algorithms have been proposed in the literature.

Federated Averaging (FedAvg). FedAvg is the most widely used aggregation algorithm [3]. After local training, each client k sends its updated weights w_k to the server, which computes a weighted average based on the size of each client's local dataset as shown in (2):

$$w_{t+1} = \sum_{k} \left(\frac{n_k}{n} \right) \cdot w_k^{t+1} \tag{2}$$

where n_k is the number of samples on client k, n is the total number of samples across all participating clients, and K is the number of clients. FedAvg is simple, effective, and performs well when client data is independent and identically distributed (IID).

Federated Stochastic Gradient Descent (FedSGD). In FedSGD, each client computes the gradient of the loss function on a mini-batch of its local data and sends this gradient to the server. The server then performs a single gradient descent step on the global model. FedSGD exchanges updates more frequently than FedAvg but increases communication cost [3].

FedProx. FedProx is an extension of FedAvg designed to handle the challenges of client heterogeneity, particularly non-IID data and variable computational capabilities [29]. It adds the proximal term shown in (3) to the local objective function to prevent local updates from drifting too far from the global model:

$$\min_{w} F_k(w) + \left(\frac{\mu}{2}\right) \|w - w_t\|^2 \tag{3}$$

where F_k is the local loss function of client k, w_t is the current global model, and μ is a hyperparameter that controls the strength of the proximal term.

Other algorithms. Many other aggregation algorithms have been proposed to address specific challenges, including FedNova (to handle heterogeneous local updates), SCAFFOLD (to correct client drift in non-IID settings), and personalized federated learning approaches that train a distinct model for each client [26].

1.5.6 Privacy-Preserving Techniques

Although federated learning significantly improves privacy compared to centralized learning by keeping raw data on clients, the shared model updates can still leak sensitive information [26]. Several complementary techniques have been proposed to further strengthen privacy.

Differential Privacy (DP). Differential privacy is a mathematical framework that provides a formal privacy guarantee [31]. It ensures that the output of an algorithm remains approximately the same whether or not any single individual's data is included in the input. In federated learning, DP is typically implemented by adding calibrated random noise (e.g., Gaussian or Laplace noise) to the model updates before sending them to the server [32]. A DP mechanism is characterized by two parameters, ε (the privacy budget) and δ, which together quantify the strength of the privacy guarantee. Smaller values of ε correspond to stronger privacy protection but typically result in reduced model accuracy; this is known as the privacy–utility trade-off.

Secure Aggregation. Secure aggregation is a cryptographic protocol that allows the server to compute the sum of clients' updates without accessing individual updates [33]. It is typically based on techniques such as secret sharing or masking with pairwise random values. As a result, the server learns only the aggregated result, which protects each client's contribution.

Homomorphic Encryption (HE). Homomorphic encryption allows computations to be performed directly on encrypted data, producing an encrypted result that, once decrypted, matches the result of the same computations performed on the plain data [34]. In FL, clients can encrypt their updates before sending them to the server, which can then aggregate them in the encrypted domain. HE provides strong privacy guarantees but introduces significant computational overhead.

Secure Multi-Party Computation (SMPC). SMPC protocols allow multiple parties to jointly compute a function over their inputs while keeping those inputs private [35]. SMPC

can be used to implement secure aggregation in FL without relying on a trusted server.

These mechanisms are often combined to achieve a suitable balance between privacy, utility, and efficiency.

1.5.7 Advantages and Challenges

Federated learning offers several key advantages:

- Privacy by design: raw data never leaves the device.
- Lower communication cost: only model updates are transmitted, not raw data.
- Scalability: FL can leverage data from a large number of distributed clients.
- Regulatory compliance: FL naturally aligns with data protection regulations.

However, FL also faces several important challenges [26, 28]:

- Statistical heterogeneity (non-IID data): Client data distributions can vary significantly, which degrades the performance of standard aggregation algorithms.
- System heterogeneity: Clients differ in computational power, memory, battery, and network bandwidth, leading to stragglers and drop-outs during training.
- Communication overhead: Even though FL reduces the volume of exchanged data compared to centralized learning, repeated communication rounds can still be costly for resource-constrained IoT devices.
- Privacy limitations: the exchange of model updates alone does not guarantee full privacy, which motivates the integration of additional mechanisms such as secure aggregation, homomorphic encryption, and differential privacy.

These challenges are at the core of current federated learning research and directly motivate the design choices of the framework proposed in this thesis.

1.6 Conclusion

In this chapter, we introduced the fundamental concepts necessary to understand the contributions of this thesis. We first described the Internet of Things, its architecture, applications, and security requirements, and then reviewed the main types of attacks that threaten IoT systems and the role of intrusion detection systems in mitigating them. We presented the foundations of machine learning and deep learning, highlighting the architectures most commonly used in security analytics. Finally, we provided a detailed overview of federated learning, including its architecture, types, data-distribution settings, aggregation algorithms, and privacy-preserving mechanisms. These elements form the theoretical basis upon which

the rest of the work is built. The next chapter reviews the state of the art in IoT attack prediction, with a particular focus on federated learning-based approaches.

Chapter 2

State of the Art

2.1 Introduction

The growing exposure of IoT systems to cyber-attacks has motivated a large body of research on intrusion and attack detection. Over the years, the field has progressed through three main waves. The earliest approaches relied on classical rule-based or signature-based techniques. They were later complemented by machine learning (ML) methods that could discover attack patterns automatically from network data. More recently, deep learning (DL) methods have delivered state-of-the-art results by learning hierarchical representations from large volumes of traffic. However, all of these approaches share a common assumption: that raw network data can be centralized at a single location for training. In an IoT context, this assumption is increasingly problematic, both for privacy and for scalability reasons. This has led to the emergence of a fourth wave of research based on federated learning (FL), in which collaborative models are trained across distributed IoT devices without exchanging their raw data.

The purpose of this chapter is to survey the state of the art in IoT attack prediction and to position our contribution within this landscape. We first describe the main benchmark datasets and evaluation metrics that are commonly used to train and compare intrusion detection models. We then review the literature along four progressive categories: classical techniques, machine learning-based techniques, deep learning-based techniques, and federated learning-based techniques. The chapter ends with a comparative discussion of existing work and a summary of the main open challenges, which together motivate the framework proposed in this thesis.

2.2 Datasets and Evaluation Metrics

2.2.1 Benchmark Datasets

The quality and realism of the dataset used for training and evaluation has a decisive influence on the reliability of any intrusion detection system. Over the last years, several benchmark datasets have been released, each targeting different attack types, network topologies, and IoT scenarios. We briefly describe the most widely used ones in the context of IoT attack detection.

N-BaIoT. N-BaIoT is one of the first widely adopted datasets dedicated to IoT botnet detection [36]. It was produced by infecting nine commercial IoT devices in a controlled laboratory with two well-known malware families, Mirai and BASHLITE. For each device, the authors extracted 115 statistical features computed from sliding time windows of network traffic. The dataset contains both benign traffic and malicious flows generated by ten different attack types, which allows both binary anomaly detection and fine-grained multi-class classification.

CICIDS2017. CICIDS2017 is a general-purpose intrusion-detection dataset generated over five days of realistic network activity [37]. It contains benign flows modelled on 25 simulated users and a variety of attacks such as DoS, DDoS, brute-force, SQL injection, infiltration, port scans, and botnet traffic. Flow-level features are extracted with CICFlowMeter, producing more than 80 features per flow, which makes the dataset suitable for both classical ML and deep learning pipelines.

Bot-IoT. The Bot-IoT dataset was designed to reflect realistic IoT traffic mixed with botnet-based attacks, including DoS, DDoS, reconnaissance (OS and service scans), keylogging, and data exfiltration [38]. It was generated on a testbed built at UNSW Canberra that combines legitimate and simulated IoT devices. The dataset is large (tens of millions of records in its raw form) and is typically used in a reduced form of about 5% of the original data for machine learning experiments.

TON_IoT. The TON_IoT dataset extends the Bot-IoT line of work by integrating heterogeneous data sources, including telemetry from IoT and Industrial IoT sensors, operating-system logs (Windows and Linux), and network-traffic captures [4]. It covers a wide range of attacks (DoS/DDoS, ransomware, data injection, MITM, password cracking, scanning, XSS, and backdoor), making it particularly suitable for evaluating IDSs in more complex IoT/IIoT environments.

CICIoT2023. CICIoT2023 is one of the most recent large-scale benchmark datasets for IoT security [39]. It was collected in a realistic testbed of 105 physical IoT and network devices and contains 33 different attacks grouped into seven classes: DDoS, DoS, reconnaissance, web-based attacks, brute force, spoofing, and Mirai-based attacks. Because of its scale, diversity, and realism, it has rapidly become a reference dataset for evaluating both centralized and federated intrusion detection approaches.

In addition to these datasets, older resources such as NSL-KDD and KDD Cup 99 are

still used for comparison purposes, even though they no longer reflect the characteristics of modern IoT traffic. The choice of dataset is a key design decision because it strongly affects generalization: a model trained on a dataset that does not reflect the deployment environment can achieve very high accuracy in the laboratory yet fail in the field.

Table I presents the benchmark datasets most frequently used in IoT intrusion-detection research.

Table I: Benchmark IoT intrusion-detection datasets

Dataset	Main Characteristics	Approx. Scale	Typical Use
N-BaIoT [36]	Benign and Mirai/BASHLITE botnet traffic collected from 9 commercial IoT devices with 115 statistical features	10 attack types	Binary and multi-class botnet detection
CICIDS2017 [37]	Five days of realistic enterprise traffic with benign behaviour and diverse attacks extracted with CICFlowMeter	80+ flow features	General intrusion-detection benchmarking
Bot-IoT [38]	Realistic IoT traffic mixed with botnet attacks including DoS, DDoS, reconnaissance, keylogging, and exfiltration	Tens of millions of records	Large-scale botnet and attack classification
TON_IoT [4]	Heterogeneous telemetry, operating-system logs, and network traffic covering IoT and IIoT attacks	22.5M network-flow records	Multi-source IoT/IIoT intrusion detection
CICIoT2023 [39]	Large-scale realistic testbed with 105 IoT and network devices and 33 attack types	33 attacks in 7 classes	Modern large-scale IoT attack benchmarking

2.2.2 Evaluation Metrics

Assessing the performance of an intrusion detection system requires a combination of metrics that jointly capture detection quality and efficiency. The most commonly used metrics can be grouped into three categories.

Classification metrics. These metrics measure the ability of the model to correctly classify normal and malicious samples. They are derived from the confusion matrix, which counts true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

- Accuracy.
$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4}$$
It reflects the overall correctness of the model but can be misleading when classes are imbalanced, which is common in intrusion detection.

- Precision.
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$
It measures the proportion of predicted attacks that are truly malicious, and is a key indicator of false-alarm control.

- Recall (or Detection Rate).
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$
It indicates how many of the actual attacks are correctly detected.

- F1-score.
$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$
It provides a harmonic mean of precision and recall and is the most informative single metric for imbalanced attack detection.

- Macro F1-score.
$$\text{Macro F1-score} = \frac{1}{N} \sum_{i=1}^{N} \text{F1-score}_i \tag{8}$$
where N represents the number of classes and F1-score_i denotes the F1-score computed for class i. This metric assigns equal importance to all classes by averaging their individual F1-scores, making it particularly suitable for evaluating imbalanced multiclass attack detection scenarios.

Efficiency metrics. In the IoT context, the ability to detect attacks quickly and with limited resources is as important as raw accuracy. Relevant efficiency metrics include training time, inference (detection) latency per sample, model size (number of parameters), and memory footprint. These metrics are especially critical when deploying detection models directly on resource-constrained IoT devices.

Federated learning-specific metrics. Federated settings introduce additional performance dimensions that centralized approaches do not capture.

- Communication cost. Usually measured as the total number of bytes exchanged between clients and server, or the number of communication rounds needed to reach a target accuracy. This is a central concern for bandwidth-constrained IoT deployments.

- Convergence speed. The number of rounds required for the global model to stabilize, which strongly depends on the aggregation algorithm and on the degree of data heterogeneity across clients.
- Privacy metrics. When differential privacy is used, the privacy budget (ε, δ) characterizes the strength of the privacy guarantee. In cryptographic approaches, security is typically evaluated against a formal threat model (e.g., honest-but-curious server, colluding clients). Empirical privacy can also be measured through the success rate of membership-inference and reconstruction attacks against the trained model.
- Robustness to heterogeneity. The accuracy degradation observed when moving from IID to non-IID client data distributions is often reported, as it quantifies how well an FL approach adapts to realistic device heterogeneity.

A meaningful comparison of IoT attack prediction systems therefore requires reporting not only classification metrics but also communication and privacy costs, especially when federated learning is considered.

2.3 Literature Review

Following the pattern observed in the broader intrusion detection literature, we organize this review in four progressive categories. The first two (classical and machine learning-based techniques) summarize the background against which modern systems are built. The third category (deep learning) represents the current centralized state of the art. The fourth category (federated learning) contains the work most directly related to our contribution.

2.3.1 Classical Techniques

Classical intrusion detection systems rely on predefined signatures or expert rules to identify malicious traffic. Well-known open-source systems such as Snort and Suricata exemplify this approach: each incoming packet is compared against a database of known attack patterns, and a matching rule triggers an alert. Signature-based systems offer high precision on known attacks, low false-alarm rates on the patterns they cover, and are easy to interpret by human analysts [13].

However, these systems have three well-documented limitations that become particularly severe in IoT environments. First, they cannot detect zero-day attacks, i.e., attacks whose signatures are not yet in the database. Second, maintaining an up-to-date signature database for the heterogeneous and rapidly evolving landscape of IoT malware is costly and error-prone. Third, many IoT devices have insufficient computational resources to run a full signature-matching engine locally. As shown in [2], the ease with which variants of the Mirai botnet circumvent classical rule sets highlights the structural insufficiency of signature-based detection for IoT. These limitations motivated the shift to learning-based approaches.

Table II summarizes representative classical IDS platforms and their main limitations in IoT settings.

Table II: Classical signature-based IDSs for IoT

System / Approach	Detection Method	Strengths	Limitations in IoT
Snort	Rule / signature-based	Mature, high precision on known attacks	Heavy rule base, poor on zero-day, high resource cost
Suricata	Rule / signature + multi-threaded	Better throughput than Snort, supports scripting	Still signature-dependent, not suited for constrained devices
Bro / Zeek	Protocol analysis, policy-based	Deep protocol visibility, flexible scripting	Requires powerful hardware, limited IoT protocol support
OSSEC (HIDS)	Log analysis, host-based rules	Lightweight per-host monitoring	Needs log access on each device, weak for network-level IoT attacks

2.3.2 Machine Learning-based Techniques

Machine-learning-based intrusion-detection systems learn attack patterns directly from labelled or unlabelled traffic data, which makes them more adaptable than rule-based approaches. An early but still influential survey is presented in [16], which reviews algorithms such as decision trees, random forests, Support Vector Machines (SVMs), Naive Bayes, and k-Nearest Neighbors (k-NN). In the specific context of IoT, a number of studies have applied these classical ML algorithms to benchmark datasets.

For example, the study introducing Bot-IoT [38] evaluated several ML classifiers, including SVMs and recurrent neural networks, and showed that ensemble methods and shallow deep models can achieve very high detection rates on botnet traffic, albeit with long training times and substantial feature engineering. Similarly, early work on the N-BaIoT dataset demonstrated that anomaly-based methods can separate benign from malicious IoT traffic with high accuracy once informative statistical features are extracted from packet streams [36].

Despite these encouraging results, classical ML methods share two important limitations in the IoT setting. First, they rely heavily on handcrafted features, whose design requires substantial domain expertise and is hard to transfer across datasets. Second, their performance tends to degrade on complex attacks that require capturing temporal or sequential patterns in the traffic, which is precisely where deep learning becomes advantageous. Table III summarizes representative ML-based studies and their reported performance.

Table III: Machine-learning approaches for IoT intrusion detection

Reference	Algorithm(s)	Dataset	Reported Performance
[16]	Survey of DT, RF, SVM, NB, k-NN, ANN	Multiple (KDD, NSL-KDD, etc.)	Summarizes accuracy ranges of 85–99% depending on attack type
[36]	Statistical features + anomaly scoring	N-BaIoT (Mirai, BASHLITE)	TPR $\approx$ 100%, very low FPR on 9 IoT devices
[38]	SVM, RNN, LSTM baselines	Bot-IoT	Accuracy $>$ 99% for DoS/DDoS after feature reduction
[37]	Random Forest with feature selection	CICIDS2017	Best F1 reported with RF on top-selected features

2.3.3 Deep Learning-based Techniques

The success of deep learning in image recognition and natural language processing [5] has naturally extended to intrusion detection. Deep models reduce the need for handcrafted features, learn hierarchical representations of traffic, and scale well with the volume of data, which makes them particularly attractive for IoT security analytics.

A foundational contribution to IoT attack detection using deep learning is reported in [36]. That study trained a deep autoencoder per device on benign traffic only and flagged as anomalous any sample whose reconstruction error exceeded a device-specific threshold. The approach achieved a true-positive rate close to 100% while keeping the false-positive rate low, and became a reference example for unsupervised deep anomaly detection in IoT.

Subsequent works have explored a wider range of architectures. Convolutional Neural Networks (CNNs) have been applied to the classification of attack flows after reshaping traffic features into image-like tensors, sometimes combined with recurrent layers to capture temporal dependencies. Long Short-Term Memory (LSTM) networks [22] and Gated Recurrent Units (GRUs) have been widely adopted to model traffic as sequences, showing strong performance on slow, long-lasting attacks such as Advanced Persistent Threats and certain DDoS patterns. Hybrid architectures combining CNNs with LSTMs or Bidirectional LSTMs have been reported to achieve accuracies above 98% on datasets such as N-BaIoT, Bot-IoT, and TON_IoT.

However, these centralized deep-learning approaches share an important structural limitation: they assume that a large, representative training dataset is available on a central server. In IoT, this raises three concerns. First, transferring massive traffic logs from millions of devices to a cloud server is costly in bandwidth and energy. Second, raw traffic

can carry sensitive information about users' activities. Third, legal frameworks such as the GDPR restrict the centralization of personal data. These issues motivated the exploration of federated learning for IoT security, which is the focus of the next subsection. Table IV summarizes representative centralized deep-learning studies and their main limitations.

Table IV: Centralized deep-learning approaches for IoT intrusion detection

Reference	Architecture	Dataset	Key Results / Limitation
[36]	Deep Autoencoders (per device)	N-BaIoT	TPR $\approx$ 100%; unsupervised, but needs one model per device
[38]	LSTM / RNN baselines	Bot-IoT	Very high accuracy; heavy training on full dataset
Hybrid CNN-LSTM works	CNN + LSTM / Bi-LSTM	N-BaIoT, Bot-IoT, TON_IoT	Accuracy $>$ 98%; large models, centralized training
Centralized DL (general)	DNN / CNN / GRU	CICIDS2017, CICIoT2023	State-of-the-art accuracy; privacy and bandwidth concerns

2.3.4 Federated Learning-based Techniques for IoT Security

Following the introduction of FedAvg [3], federated learning has been rapidly adopted as a framework for collaborative security analytics. In the IoT domain, several seminal and recent works illustrate the main directions of current research.

Taxonomy of FL-based IDS in IoT. Before reviewing individual systems, it is useful to organize the field around a taxonomy that captures the main design axes shared by current proposals. Drawing on the surveys in [40–42], we identify five complementary dimensions along which an FL-based IDS for IoT can be characterized.

- Federation architecture. Most proposals follow the classical client–server architecture inherited from FedAvg, where a central server aggregates model updates from IoT clients. Variants include hierarchical (edge–fog–cloud) architectures, in which an intermediate fog layer aggregates updates from groups of nearby devices before forwarding them to the cloud, and decentralized peer-to-peer architectures, in which devices exchange updates directly without a central server.
- Data partitioning. The setting can be horizontal (clients share the same feature space but hold different samples), vertical (clients hold different features about the same entities), or based on federated transfer learning. For IoT attack detection, the horizontal

setting is by far the most common, as different devices or gateways observe similar traffic features from different subsets of users.

- Local model family. FL-based IDS systems differ in the type of local model trained on each device. Three main families appear in the literature: (a) fully connected DNN / MLP classifiers trained on flow features; (b) recurrent models (RNN, GRU, LSTM) that treat traffic as sequences; and (c) autoencoder-based anomaly detectors that flag samples with high reconstruction error. Hybrid CNN-LSTM models form a fourth, less common, category.
- Aggregation strategy. FedAvg remains the default aggregation rule, but several works adopt FedProx (to handle heterogeneous clients), FedSGD (for finer-grained updates), or robust aggregation rules (Krum, median, trimmed mean) that resist poisoning attacks by malicious clients.
- Privacy mechanism. FL by itself keeps raw data local, but several works strengthen privacy by adding one or more complementary mechanisms: differential privacy (Gaussian or Laplace noise added to updates), secure aggregation (cryptographic masking of updates), or homomorphic encryption of the exchanged parameters. Systems can therefore be classified as plain FL, FL + DP, FL + secure aggregation, FL + HE, or combinations thereof.

Figure 9 shows these dimensions as a compact taxonomy. This figure provides a bridge between the general design space of FL-based IDSs and the individual systems reviewed next.

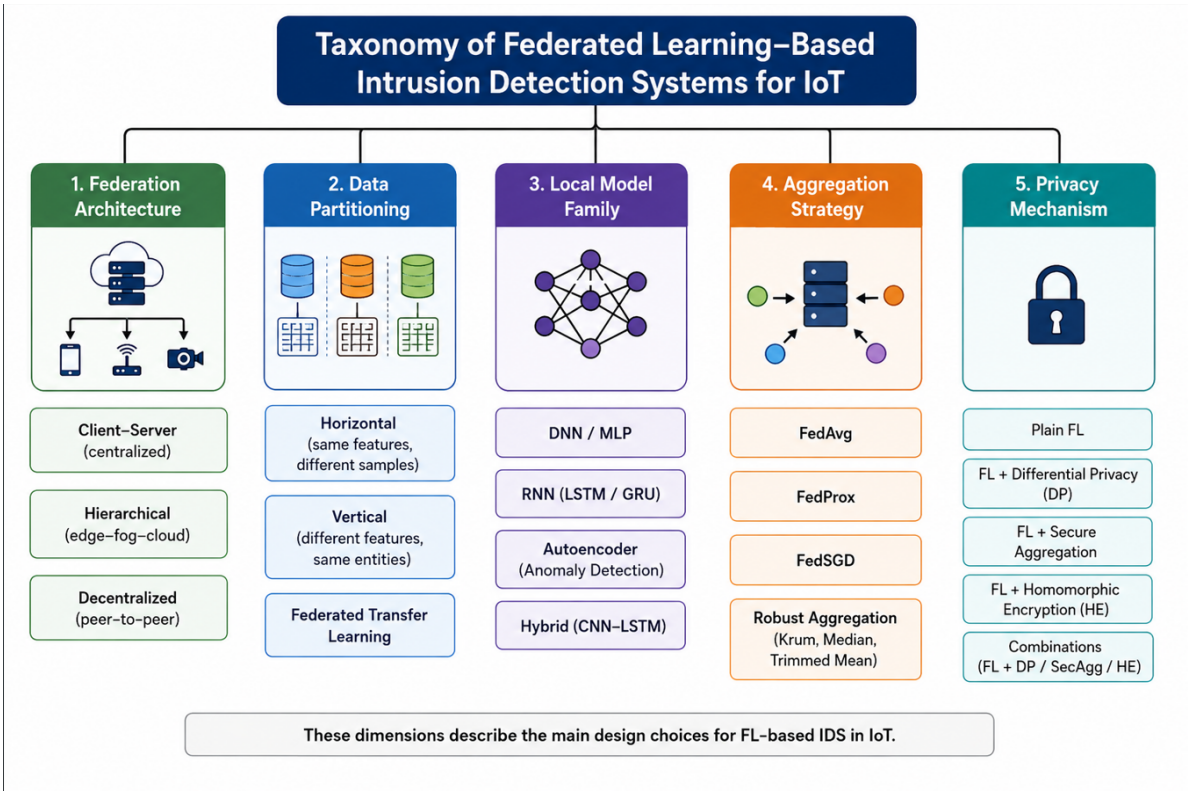


Figure 9. Taxonomy of federated-learning-based IDSs for IoT.

Using this taxonomy as a guide, we now review the most representative works in the literature.

DÏoT. DÏoT is often cited as one of the earliest systems to employ a federated-learning approach for anomaly-based intrusion detection in IoT [25]. It builds one communication profile per IoT device type using recurrent neural networks and aggregates these profiles across many gateways without sharing raw traffic. The reported system achieved a detection rate of about 95.6% on attacks launched from devices compromised by Mirai, with an average detection delay of a few hundred milliseconds. It demonstrated that federated learning can achieve near-centralized accuracy while keeping data local, and it established a blueprint followed by many subsequent works.

Federated anomaly detection with GRUs. A federated GRU-based anomaly-detection approach for IoT security attacks is presented in [45]. The method trains Gated Recurrent Units locally on each device and aggregates them through FL rounds with an additional ensembler module at the server to stabilize the global model. The reported results show accuracy comparable to centralized training on benchmark IoT datasets while preserving data locality.

FELIDS. FELIDS is a federated-learning-based intrusion-detection system initially designed for agricultural IoT networks [46]. It integrates three deep-learning classifiers, namely a deep neural network (DNN), a CNN, and an RNN, each trained in a federated fashion over

datasets such as CSE-CIC-IDS2018, MQTTset, and InSDN. The reported results indicate that federated deep models can match centralized baselines in many cases while offering natural privacy benefits. The study also highlights the need to address the assumption that all clients are trustworthy, leaving the handling of poisoning attacks to future work.

FL-LSTM for IoT WSNs. Recent work has revisited the combination of federated learning and LSTM for intrusion detection in IoT-based wireless sensor networks [43]. The reported model achieves higher accuracy and a lower false-positive rate than centralized counterparts on multiple datasets while keeping raw data on the sensor nodes. This study is representative of the current trend toward privacy-preserving, sequence-aware federated models.

Surveys and systematic reviews. The emerging FL-IDS literature is synthesized by several recent surveys [40–42]. These reviews consistently identify three recurring issues in IoT-oriented FL-IDS research: non-IID data across heterogeneous devices, communication overhead in bandwidth-constrained networks, and the need for formal privacy guarantees beyond the mere fact of keeping data local.

Privacy-preserving FL-IDS. Privacy-preserving federated learning for intrusion detection in industrial IoT settings has been evaluated in [44], which discusses the trade-off between privacy budget and detection performance. This line of research is the most directly related to our thesis because it combines collaborative training across distributed IoT clients, deep models capable of capturing attack patterns, and formal privacy mechanisms on top of federated averaging. Table V compares representative FL-based IDS proposals for IoT.

Table V: Federated-learning IDS proposals for IoT

System / Reference	Local Model	Aggregation	Privacy Mechanism	Dataset
DÏoT [25]	GRU profiles	FedAvg	FL only (data locality)	In-lab IoT + Mirai
[45]	GRU + ensembler	FedAvg	FL only	Modbus IoT traffic
FELIDS [46]	DNN / CNN / RNN	FedAvg	FL + gRPC encryption	CSE-CIC-IDS2018, MQTTset, InSDN
[44]	DNN	FedAvg	FL + Differential Privacy	Industrial IoT datasets
FL-LSTM [43]	LSTM	FedAvg	FL only	Multiple IoT-WSN datasets

2.4 Comparative Discussion and Open Challenges

Across the four categories reviewed above, a consistent trend can be observed. Classical rule-based systems remain useful as a first line of defence, but they cannot cope with the scale and diversity of modern IoT threats. Machine learning-based techniques have shown that data-driven detection is feasible, yet they depend heavily on feature engineering and struggle with complex temporal attacks. Deep learning has shifted the state of the art by learning rich representations directly from raw traffic, achieving very high detection rates on benchmark IoT datasets such as N-BaIoT, Bot-IoT, TON_IoT, and CICIoT2023. Federated learning, finally, adds a distributed and privacy-aware dimension that is particularly well-suited to the IoT context.

A comparative reading of existing FL-IDS works for IoT, however, reveals that many proposals focus on one dimension of the problem at a time. Some works mainly evaluate detection accuracy under IID data partitions, without measuring how the system degrades when devices observe different attack distributions. Others address communication efficiency but assume a semi-trusted server. Still others integrate differential privacy or secure aggregation, but on small-scale datasets that do not fully reflect the heterogeneity of real IoT deployments. As summarized in the surveys mentioned above, several open challenges therefore remain.

- Statistical and system heterogeneity. Most existing FL-IDS studies assume relatively homogeneous clients. In practice, IoT devices differ widely in traffic patterns, hardware, battery, and connectivity, which leads to non-IID data and to stragglers during training.
- Formal privacy guarantees. Keeping raw data on the device does not, by itself, prevent information leakage from shared model updates. Relatively few works in the IoT FL-IDS literature integrate differential privacy with a principled analysis of the privacy–utility trade-off.
- Communication overhead. Repeated model exchanges over constrained IoT links can become a bottleneck. Techniques such as gradient sparsification, model compression, or adaptive client selection remain underexplored in the specific context of IoT attack prediction.
- Robustness to adversarial clients. Federated learning is vulnerable to poisoning and backdoor attacks by malicious clients. The intersection of FL-IDS and defences against such attacks is an active but still maturing area.

These observations directly motivate the core issues addressed in this thesis: collaborative distributed training across IoT clients, protection of raw network traffic through federated learning combined with privacy-oriented aggregation, and explicit handling of device heterogeneity. The next chapter presents the proposed FedShield framework.

2.5 Conclusion

In this chapter, we reviewed the state of the art in IoT attack prediction. We first described the main benchmark datasets used in the community (N-BaIoT, CICIDS2017, Bot-IoT, TON_IoT, and CICIoT2023) and the evaluation metrics relevant to both centralized and federated settings. We then organized the literature into four progressive categories, from classical rule-based systems to modern federated learning-based intrusion detection. The comparative discussion highlighted that, although federated learning has become a very active direction for IoT security, several challenges remain open, in particular statistical and system heterogeneity, formal privacy guarantees, and communication efficiency. These challenges form the backdrop of our work and highlight important directions for future research.

Chapter 3

Proposed Framework

3.1 Introduction

After presenting the theoretical background in Chapter 1 and reviewing related work in Chapter 2, this chapter presents the proposed FedShield framework. The goal is to describe the conceptual design, the system architecture, the local model, the federated training procedure, and the privacy-preserving mechanism before turning to implementation and experimental evaluation.

FedShield is a federated learning pipeline for multi-class network intrusion detection. It combines the Federated Averaging algorithm of McMahan *et al.* [3] with a mask-based simulated Secure Aggregation protocol inspired by Bonawitz *et al.* [33]. The Flower framework [47] handles orchestration, and the local classifier is a lightweight Multi-Layer Perceptron trained on flow-level features from the TON_IoT dataset [4].

Several recent studies have shown that federated learning can approach centralized intrusion-detection performance on IoT benchmarks. Examples include DÏoT [25], the GRU-based system of Mothukuri *et al.* [45], FELIDS [46], the LSTM-based model of Anwar *et al.* [43], and the privacy-oriented work of Ruzafa-Alcázar *et al.* [44]. However, many of these studies either rely only on data locality or use differential privacy with a visible utility cost. In contrast, FedShield integrates mask-based Secure Aggregation and checks at every round that its output matches plain FedAvg up to floating-point precision.

The rest of the chapter is organised as follows. Section 3.2 presents the framework architecture. Section 3.3 describes the local model design. Section 3.4 explains the federated training procedure. Section 3.5 presents the privacy-preserving mechanism.

3.2 Framework Architecture

FedShield follows the federated learning setting introduced in Chapter 1. It includes one central server and ten participating clients, all simulated in the same Python process. The

server initialises the global model, sends it to the clients at the start of each round, aggregates the masked client updates, recovers the unmasked global update through SecAgg, and evaluates the resulting global model on a held-out test set. Each client keeps its local data partition private and sends only masked model parameters to the server.

To create non-IID data, we use a Dirichlet distribution with concentration parameter $\alpha = 0.3$. For each class c, client proportions are drawn as $\mathbf{p}_c \sim \text{Dirichlet}(\alpha, \ldots, \alpha)$. Samples of class c are then assigned to the ten clients according to these proportions, with a one-percent floor so that every client receives at least a small share of each class. This creates strong heterogeneity. The largest client receives 1,228,637 samples, while the smallest receives 228,891. The class mixtures also vary clearly across clients. This setup reflects realistic IoT deployments, where different gateways observe different traffic patterns [26].

The test set stays on the server and remains the same across all training rounds. This provides a stable estimate of the global model's generalisation performance. After each round, the server broadcasts the updated global weights and the process repeats. As illustrated in Figure 10, these steps form a complete FedShield workflow from local training to privacy-preserving aggregation.

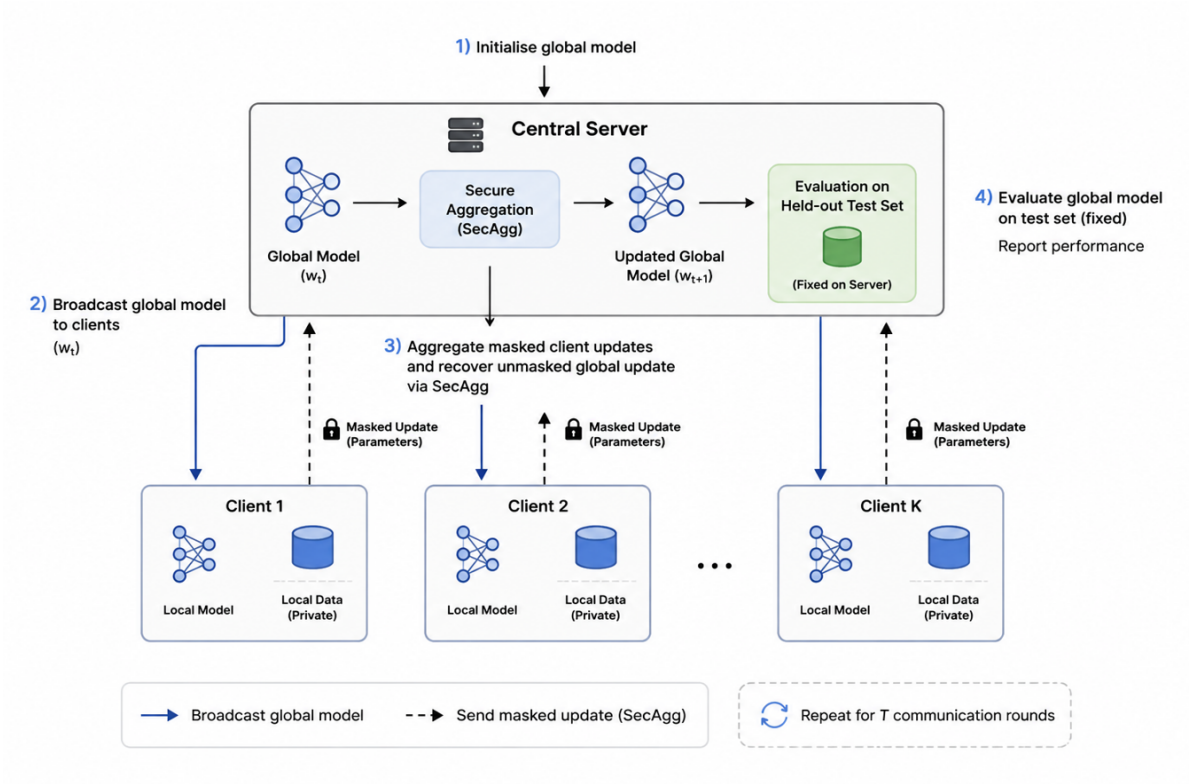


Figure 10. FedShield system architecture.

3.3 Local Model Design

All clients use the same local model. It is a Multi-Layer Perceptron with three fully connected hidden layers of 256, 128, and 64 ReLU neurons, followed by a 9-class softmax output layer. A dropout rate of 0.3 is used after each hidden layer to reduce overfitting [48]. The model has 49,929 trainable parameters distributed across eight weight tensors. This corresponds to about 195 KB in `float32` format. Figure 11 shows the local MLP architecture used by all clients and connects the input feature vector to the final nine-class output.

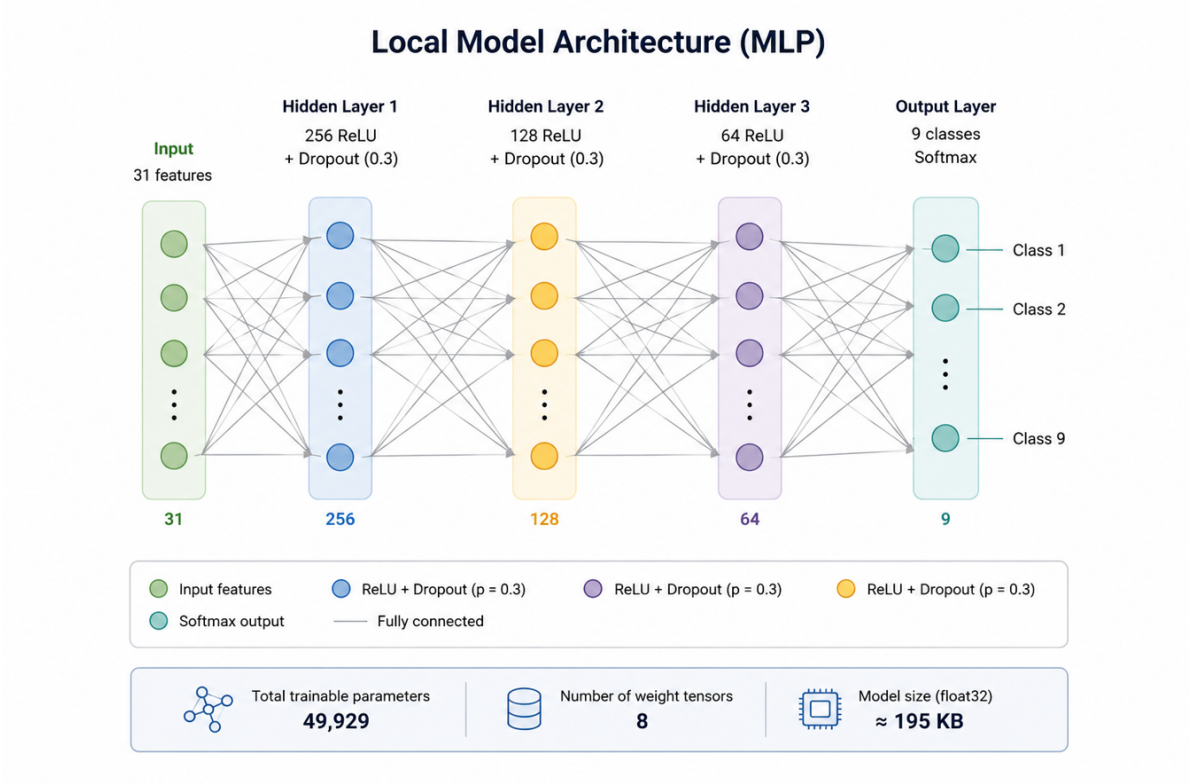


Figure 11. FedShield local MLP architecture.

The choice of an MLP is deliberate. The 31 TON_IoT features are tabular flow statistics, so they do not have a clear sequential or spatial structure. For this type of data, convolutional and recurrent architectures do not have a clear advantage. Work on tabular learning also shows that simple models can remain strong baselines [49]. The small size of the model is another advantage because it better matches the resource limits of IoT devices.

To reduce the effect of class imbalance, we replace standard categorical cross-entropy with the Focal Loss proposed by Lin *et al.* [50]:

$$\mathcal{L}_{\mathrm{FL}}(p_t) = -\alpha\,(1-p_t)^{\gamma}\,\log(p_t), \tag{9}$$

where p_t is the predicted probability of the true class. The factor $(1-p_t)^{\gamma}$ reduces the contribution of easy examples and focuses the learning process on harder ones. We use the

recommended parameters from the original paper [50]: $\gamma = 2.0$ and $\alpha = 0.25$. Predictions are clipped to $[10^{-7}, 1 - 10^{-7}]$ for numerical stability. The optimiser is Adam [51] with its default hyperparameters (learning rate 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$).

3.4 Federated Training Procedure

FedShield uses the standard Federated Averaging algorithm [3]. At communication round r, the server broadcasts the current global parameters $\boldsymbol{\theta}_r$ to all $K = 10$ clients. Each client k then trains for $E = 3$ local epochs with the Adam optimiser on its local dataset $\mathcal{D}_k$ of size n_k, using a batch size of 2048. We denote the updated local parameters by $\boldsymbol{\theta}_k^{(r+1)}$. The server then computes a weighted average based on local dataset sizes:

$$\boldsymbol{\theta}_{r+1} = \sum_{k=1}^{K} \frac{n_k}{N} \boldsymbol{\theta}_k^{(r+1)}, \qquad N = \sum_{k=1}^{K} n_k. \tag{10}$$

In FedShield, the parameters transmitted over the network are not the raw client weights but their *masked* versions. The unmasking step is described in Section 3.5.

One complete run contains $R = 30$ communication rounds. We do not use early stopping. In federated training, consecutive rounds are not directly comparable in the same way as centralized epochs because each round starts from a newly aggregated global model. For this reason, convergence is controlled by a fixed round budget. After each round, the global model is evaluated on the held-out test set. We record the loss, accuracy, Macro F1, Weighted F1, and the SecAgg verification error described in Section 3.5.

Three design choices should be noted. First, we set the number of local epochs to $E = 3$ to limit client drift under the strongly non-IID Dirichlet split. Longer local training would increase the divergence between clients, which plain FedAvg cannot fully correct [29]. Second, the participation fraction is set to one, so all ten clients take part in every round. This is feasible in simulation and strengthens each aggregation step, although a real deployment might require partial participation. Third, the batch size of 2048 is a practical compromise between gradient noise and per-round training time on the CPU execution environment used in the experiments.

3.5 Privacy-Preserving Mechanism

The privacy component of FedShield is a mask-based Secure Aggregation scheme inspired by Bonawitz *et al.* [33]. Its purpose is to mimic the aggregation behaviour of SecAgg while still allowing the server to compute the weighted average required by FedAvg. In this implementation, the masks are not created through cryptographic key agreement. Instead, they are generated from deterministic seeds known to both the client and the server. The mechanism should therefore be understood as a functional simulation of SecAgg that lets us verify

the aggregation pipeline experimentally. The security implications of this simplification are discussed below.

For each communication round r and each client k, the mask $\mathbf{m}_k^{(r)}$ is generated as a tensor with the same shape as the model parameters and filled with standard Gaussian samples produced by `np.random.RandomState(`s`).randn(…)` with seed

$$s = \texttt{RANDOM\_SEED} + 1000 \cdot r + k, \tag{11}$$

where `RANDOM_SEED` $= 42$. After local training, the client obtains its updated parameters $\boldsymbol{\theta}_k^{(r+1)}$ and uploads the masked version $\boldsymbol{\theta}_k^{(r+1)} + \mathbf{m}_k^{(r)}$, together with its sample count and client identifier.

On the server side, our custom `SecAggFedAvg` strategy extends the Flower `FedAvg` class. It first applies the weighted average in equation (10) to the masked weights. Because the seeds in equation (11) depend only on the round index and the client identifier, both of which are known to the server, the server can reconstruct each $\mathbf{m}_k^{(r)}$ and compute the weighted sum of all masks, $\sum_k (n_k/N)\, \mathbf{m}_k^{(r)}$. Subtracting this quantity from the aggregate of the masked weights recovers the true Federated Averaging output:

$$\sum_{k=1}^{K} \frac{n_k}{N} \left(\boldsymbol{\theta}_k^{(r+1)} + \mathbf{m}_k^{(r)} \right) \; - \; \sum_{k=1}^{K} \frac{n_k}{N}\, \mathbf{m}_k^{(r)} \; = \; \sum_{k=1}^{K} \frac{n_k}{N}\, \boldsymbol{\theta}_k^{(r+1)}. \tag{12}$$

The strategy then checks the unmasking step by measuring the maximum element-wise absolute difference between the SecAgg result and a direct plain-text FedAvg computed from the unmasked weights. This value is recorded as the SecAgg verification error for the round.

This design has two useful properties. First, masked aggregation is mathematically equivalent to plain FedAvg up to floating-point rounding. In this experiment, the mechanism therefore does not reduce detection performance, as confirmed in Section 4.4. Second, the server receives masked updates during communication, which lets the implementation exercise the same aggregation and unmasking logic that a Secure Aggregation layer would require. However, because the seeds are shared deterministically, the server can reconstruct the masks. For this reason, the current implementation is a functional simulation rather than a full cryptographic deployment.

The limitation therefore comes from the threat model rather than from the aggregation algebra. A server that knows the seed rule can reconstruct each mask and recover each client update. A real deployment would replace these deterministic seeds with masks that no single party can reconstruct, following the original Secure Aggregation protocol of Bonawitz *et al.* [33]. At a high level, such a protocol proceeds in three steps. First, every pair of clients (k, j) establishes a shared secret through a *Diffie–Hellman key agreement*: each client publishes a public key, and any two clients can then independently compute the same pairwise secret without transmitting it to the server. Second, each client derives its pairwise masks from these secrets and adds them to its update so that, by construction, the mask used by

client k with respect to client j is the exact negative of the mask used by client j with respect to client k. When the server sums all masked updates, these pairwise terms cancel, and the server obtains the correct aggregate without observing any individual client contribution in the clear. Third, to tolerate clients that disconnect mid-round, the pairwise secrets are protected with a *secret-sharing* scheme, such as Shamir's secret sharing, so that the masks associated with a disconnected client can be reconstructed by the remaining participants and removed from the aggregate.

Integrating this cryptographic protocol would turn the present functional simulation into a genuinely privacy-preserving system, but it would also add extra key-exchange and secret-sharing rounds. Importantly, this extension would not change the detection results reported in this thesis, because the aggregate computed by cryptographic Secure Aggregation is mathematically the same quantity that the simulation already recovers in equation (12). The cryptographic layer protects *how* the sum is computed, not *what* sum is computed. This thesis therefore focuses on the federated learning pipeline and on verifying that mask-based aggregation can be integrated into a Flower-based system without harming utility, while a complete cryptographic implementation is left for future work.

3.6 Conclusion

This chapter presented the proposed FedShield framework before the experimental part of the thesis. It described the client–server architecture, the local MLP classifier, the FedAvg-based training procedure, and the mask-based Secure Aggregation mechanism. The next chapter presents the implementation setup, the TON_IoT data preparation pipeline, the experimental results, and the comparative discussion with previous works.

Chapter 4

Implementation and Experimental Evaluation

4.1 Introduction

After presenting the proposed FedShield framework in Chapter 3, this chapter turns to the implementation and experimental part of the thesis. Its goal is to implement the proposed framework, train it on a realistic IoT intrusion-detection dataset, and study its behaviour in a federated and privacy-preserving setting.

The system evaluated in this chapter is **FedShield**. It is the federated learning pipeline described in Chapter 3, combining Federated Averaging, a lightweight Multi-Layer Perceptron, and a mask-based simulated Secure Aggregation protocol.

The rest of the chapter is organised as follows. Section 4.2 describes the implementation environment and tools. Section 4.3 presents the dataset and preprocessing pipeline. Section 4.4 reports the experimental results. Section 4.5 compares FedShield with related federated IDS studies.

4.2 Environment and Tools

FedShield is implemented in Python 3.12 and executed in a Kaggle notebook running on CPU. The deep learning components use TensorFlow 2.19 and the Keras API. Federated orchestration is handled by the Flower framework [47] in version 1.29.0 through its in-process simulation mode, `fl.simulation.start_simulation`. Each simulated client is assigned 1 CPU, so the ten clients share the available CPU resources in a controlled way. Other libraries include `scikit-learn` for preprocessing and metrics, `pandas` and `numpy` for data processing, `matplotlib` and `seaborn` for visualisation, and `joblib` for saving trained artefacts.

Because the dataset is large, memory management is important. We load only the

32 required columns and explicitly treat "`-`" and "`(empty)`" as `NaN` markers. After each client training step, intermediate DataFrames are deleted, `gc.collect()` is called, and `keras.backend.clear_session()` is used to avoid memory buildup across the thirty rounds. The NumPy and TensorFlow seeds are fixed to 42 for reproducibility. The SecAgg masks use the deterministic seed rule described in Section 3.5.

Table VI summarizes the implementation environment and the main software tools used in the experiments.

Table VI: FedShield implementation environment

Component	**Version / Setting**
Platform	Kaggle notebook
Execution mode	CPU
Language	Python 3.12
TensorFlow	2.19
API	Keras
Flower (`flwr`)	1.29.0
Simulation mode	`fl.simulation.start_simulation`
Client resources	`num_cpus = 1` per client (Ray scheduling)
pandas, NumPy, scikit-learn	latest stable
matplotlib, seaborn	latest stable
joblib	latest stable
NumPy / TensorFlow seed	42
SecAgg mask seed scheme	Deterministic (see Section 3.5)

This configuration provides a controlled and reproducible setting for evaluating the proposed framework.

4.3 Dataset and Preprocessing

4.3.1 Dataset

The experiments use the network-traffic component of the TON_IoT dataset [4], which was introduced in Chapter 2 as one of the most representative public benchmarks for intrusion detection in IoT and Industrial IoT environments. The data is distributed in twenty-three CSV files (`Network_dataset_1.csv` to `Network_dataset_23.csv`) and contains 22,339,021 flow records described by 32 columns, namely 31 flow-level features and one target column.

Each record is labelled with one of ten categories: normal traffic and nine attack types (backdoor, DDoS, DoS, injection, MITM, password, ransomware, scanning, and XSS). Among the 31 features, 17 are numeric and describe flow-level statistics such as source

and destination ports, durations, byte counts, packet counts in both directions, and HTTP request and response body lengths. The remaining 14 features are categorical and cover protocol, DNS, SSL/TLS, and HTTP attributes such as `proto`, `service`, `conn_state`, `ssl_version`, `ssl_cipher`, `http_method`, and `weird_notice`. No payload content is used at any stage. The model is trained only on metadata-style flow features, which is consistent with the privacy goal of the framework.

4.3.2 Data Preprocessing

A consistent preprocessing pipeline is applied to the raw TON_IoT data before federated training begins.

Class filtering. The MITM (man-in-the-middle) class is removed because it is severely under-represented in the dataset. It contains only 1,052 samples out of more than 22 million records, which is too few to be reliably learned during training or meaningfully evaluated on the test set. With such limited support, the class would not produce stable per-class metrics, and its samples would be split even more thinly once the data is partitioned across the ten clients by the Dirichlet sampler. Keeping it would therefore add noise to the evaluation without giving the model a realistic chance to learn it. After this filtering step, the dataset retains nine classes and 22,337,969 rows.

Missing-value handling. Missing values in the raw CSV files are encoded by the markers "`-`" and "`(empty)`", which are interpreted as `NaN` at load time. Numeric features are then filled with zero, while categorical features are filled with the explicit token "`missing`".

Categorical and target encoding. Each categorical feature is mapped to integer codes through a dedicated `LabelEncoder`. The target column is encoded with an additional `LabelEncoder` into integer indices.

Train/test split. A stratified 80/20 train/test split is computed on the full preprocessed data with seed 42. Stratification preserves the original class proportions on both sides of the split. This produces 17,870,375 training samples and 4,467,594 test samples. Because this split is applied before class capping, the test set never contains downsampled data and remains representative of the original distribution.

Training-set class capping. The dataset is strongly imbalanced. In the test set, the scanning and DDoS classes together contribute more than 2.6 million flows, while the ransomware class contains only 14,561 samples. To reduce this imbalance during training, each class on the training side is capped at a maximum of 1,000,000 samples through random subsampling with a fixed seed. Smaller classes are kept unchanged. This capping step is

applied *only* to the training set, so the test set keeps the original class distribution and provides a faithful estimate of operational performance. After capping, the training set contains 6,463,968 samples, while the test set remains at 4,467,594 samples.

Feature scaling. Numeric features are standardised with a `StandardScaler` fitted on the training set only and then applied to both subsets. This avoids information leakage from the test set into training. All arrays are finally cast to `float32` for TensorFlow compatibility.

Final shapes. At the end of the pipeline, the training tensor has shape $(6{,}463{,}968, 31)$ and the test tensor has shape $(4{,}467{,}594, 31)$. The training side is partially balanced through capping, while the test side keeps the original class distribution.

Table VII presents the resulting class distribution after preprocessing and training-set capping.

Table VII: Class distribution after preprocessing

Class	Train (capped)	Test (uncapped)
backdoor	406,493	101,623
ddos	1,000,000	1,233,002
dos	1,000,000	675,066
injection	362,127	90,532
normal	637,104	159,276
password	1,000,000	343,713
ransomware	58,244	14,561
scanning	1,000,000	1,428,032
xss	1,000,000	421,789
Total	**6,463,968**	**4,467,594**

As shown in Table VII, the training set is partially balanced, whereas the test set preserves the original imbalance. This distinction is important for interpreting the evaluation results in the next section.

4.4 Experimental Results

After thirty communication rounds, FedShield reaches an accuracy of 0.9584, a Macro F1 of 0.8988, and a Weighted F1 of 0.9592 on the 4,467,594-sample held-out test set. Macro precision and macro recall are 0.8738 and 0.9336, respectively. Table VIII summarises the final test-set results.

Table VIII: Final global model performance

Metric	Value
Accuracy	0.9584
Macro precision	0.8738
Macro recall	0.9336
Macro F1	0.8988
Weighted precision	0.9618
Weighted recall	0.9584
Weighted F1	0.9592

The gap between Macro F1 and Weighted F1 is important. Weighted F1 (0.9592) reflects performance on a realistic traffic distribution because it gives the same importance to every sample. As a result, dominant classes have more influence on this metric. Macro F1 (0.8988), in contrast, gives the same importance to every class and is therefore more sensitive to the minority classes. Taken together, these metrics show that FedShield performs very well on most traffic but is weaker on the rarest attacks. The per-class results confirm this pattern. Table IX reports per-class precision, recall, and F1-score. As shown in Table IX, eight of the nine classes achieve an F1-score above 0.81. The four largest classes (DDoS, DoS, scanning, and XSS) reach F1 scores between 0.96 and 0.98. The main exception is ransomware, which reaches an F1 of 0.6536. Its recall is relatively high (0.8193), but its precision is low (0.5437), which means that the model detects many ransomware samples but also produces a noticeable number of false positives. The backdoor class shows the opposite pattern: recall is almost perfect (0.9994), while precision is lower (0.8026). This means the detector rarely misses backdoor traffic but sometimes confuses other traffic with that class.

Table IX: Per-class classification performance

Class	Precision	Recall	F1-score	Support
backdoor	0.8026	0.9994	0.8903	101,623
ddos	0.9939	0.9526	0.9728	1,233,002
dos	0.9551	0.9981	0.9761	675,066
injection	0.8020	0.8379	0.8195	90,532
normal	0.9551	0.9118	0.9329	159,276
password	0.8589	0.9691	0.9107	343,713
ransomware	0.5437	0.8193	0.6536	14,561
scanning	0.9873	0.9524	0.9695	1,428,032
xss	0.9660	0.9616	0.9638	421,789

Figure 12 shows the convergence behaviour over the 30 rounds and summarises the main training signals used to evaluate FedShield. Two observations are clear. First, the model converges quickly. Accuracy increases from 0.8012 at round 1 to 0.9421 at round 4, which is an improvement of more than 14 percentage points in the first three rounds. After round 10,

accuracy stays in the $[0.95, 0.96]$ range while Macro F1 continues to improve slowly until it reaches 0.8988 at round 30.

Second, the training curve is not perfectly smooth. The loss shows visible spikes at rounds 23, 26, and 29, where it rises to about 0.015–0.020 before dropping again. This behaviour is consistent with the known sensitivity of plain FedAvg to non-IID client drift [29]. When the server aggregates updates from clients with very different local distributions, the global model can move away from the test-set optimum for a short time. Even with these spikes, the final test accuracy remains close to the best value observed during training, which suggests that the model has stabilised rather than overfitted.

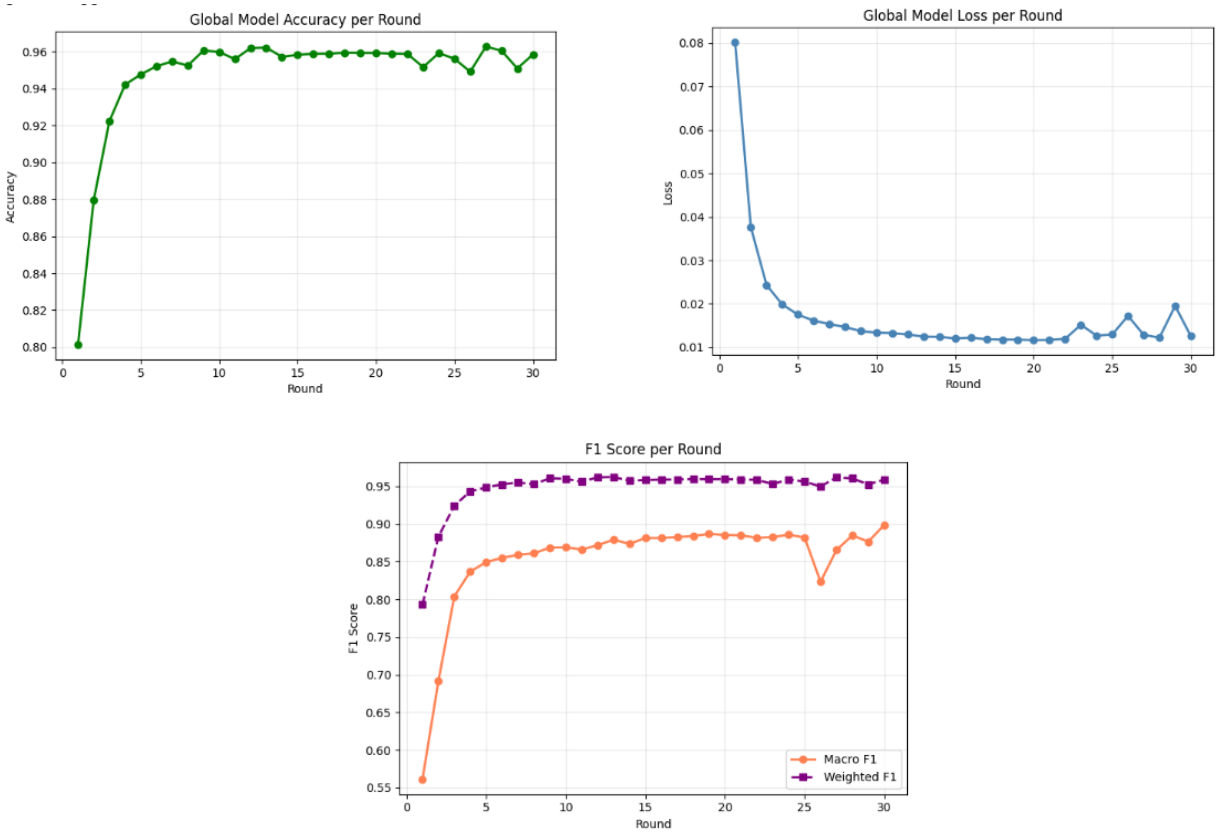


Figure 12. FedShield per-round training behaviour.

The SecAgg verification error stays negligible throughout training. As shown in the bottom panel of Figure 12, the maximum element-wise absolute difference between the SecAgg-recovered aggregate and the direct plain-text FedAvg aggregate ranges from 2.6×10^{-7} to 1.91×10^{-6} over the thirty rounds. These values are at the level of single-precision floating-point rounding. This confirms equation (12) in practice: mask-based SecAgg does not introduce measurable utility loss compared with plain FedAvg. In other words, the masking layer preserves the numerical behaviour of FedAvg in this experiment. A deployable privacy guarantee would still require cryptographic key establishment instead of deterministic shared seeds.

The communication cost of FedShield is modest. The shared model contains 49,929 parameters, which corresponds to about 195 KB in `float32` representation. In every round, the server broadcasts this model to all ten clients and receives one masked update of the same size from each of them. A single round therefore exchanges roughly 3.8 MB in both directions combined, and the complete run of thirty rounds transfers about 114 MB in total (around 57 MB downloaded and 57 MB uploaded). This volume is far smaller than what a centralized approach would require, since transferring the raw TON_IoT traffic instead of the model would involve several gigabytes of data.

Figure 13 shows the confusion matrix, while Figure 14 presents the corresponding per-class precision, recall, and F1-score values. Together, these figures explain where the final model performs well and where minority-class errors remain. Most misclassifications occur between classes with similar flow signatures. Injection samples are sometimes predicted as XSS, and ransomware is the only class whose precision falls below 0.6. The other off-diagonal entries are small in both absolute and normalised form.

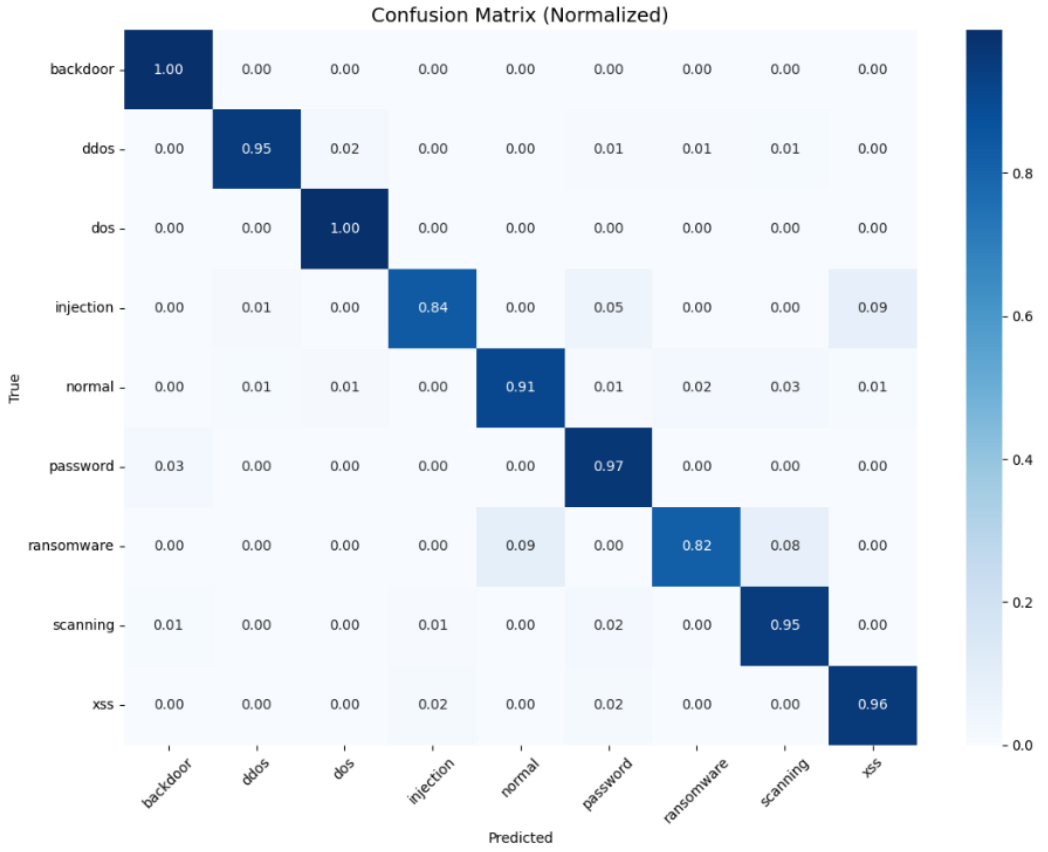


Figure 13. FedShield confusion matrix on the test set.

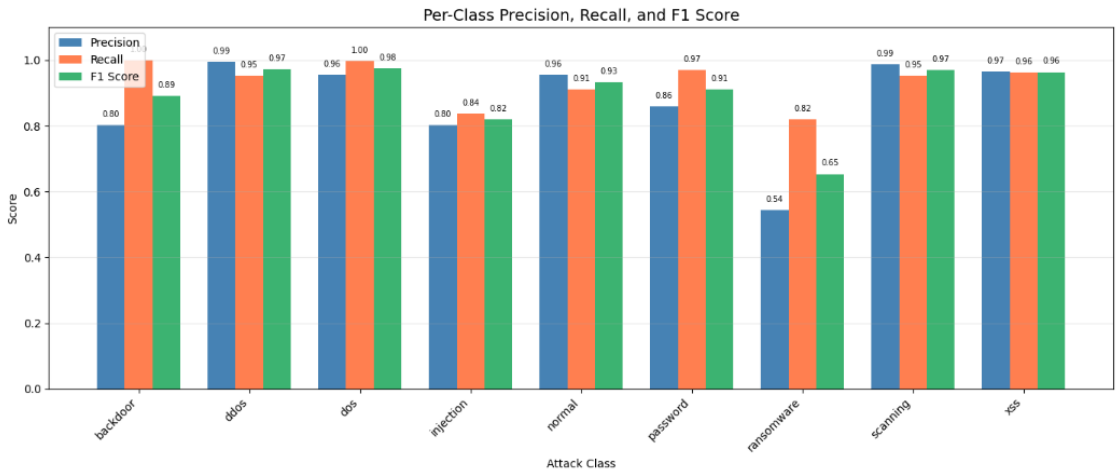


Figure 14. FedShield per-class precision, recall, and F1-score.

4.5 Comparative Discussion

Three observations stand out from the experimental study, and they all relate directly to the research questions formulated in the Introduction.

First, federated learning is a viable strategy for IoT intrusion detection on TON_IoT. The final FedShield global model reaches an accuracy of 0.9584 and a Weighted F1 of 0.9592 despite an aggressively non-IID Dirichlet split with $\alpha = 0.3$ and a comparatively lightweight MLP local model. The fact that practical detection performance remains in the high nineties confirms that the federated paradigm is not intrinsically incompatible with operationally useful intrusion detection in IoT. The cost of distribution is mostly absorbed by the minority classes, which is consistent with the qualitative pattern reported in the federated learning literature [26, 29].

Second, mask-based Secure Aggregation does not introduce measurable utility loss in this experimental setting. The SecAgg verification error remains in the 10^{-7}–10^{-6} range throughout training, which is the floating-point precision boundary of `float32`. This is the empirical counterpart of equation (12): the masks cancel exactly in the aggregation, and the resulting global model is indistinguishable from what plain FedAvg would have produced. We do not observe any round in which the SecAgg-equipped pipeline performs worse than plain FedAvg, which is consistent with the analysis of Bonawitz *et al.* [33].

Third, the bottleneck of the system is class imbalance rather than federation or aggregation. The only class with a substantially lower F1 is ransomware (0.6536), and the dominant reason is its small support: 58,244 training samples and 14,561 test samples, distributed unevenly across ten clients by the Dirichlet sampler. Focal Loss [50] partially compensates for this, but the underlying scarcity is intrinsic to the dataset. We do not observe degradation of the ransomware F1 across the federated phase that could be attributed to FedAvg-specific effects beyond this scarcity, which suggests that any serious improvement on this class would

require either targeted oversampling strategies or rare-class-aware aggregation, both of which are left for future work.

Positioning against previous works. Because the local model used in FedShield is a Multi-Layer Perceptron, the most informative comparisons are with federated-learning studies that use feed-forward classifiers (MLP/DNN) on similar IoT intrusion-detection benchmarks. Recurrent and hybrid models are reported separately for context, since their inductive biases are designed for sequential rather than tabular flow features. Table X situates FedShield within this landscape. It includes MLP/DNN studies as primary comparisons and recurrent-model works as contextual baselines.

Table X: Comparison with federated IDS studies

Work	**Local model**	**Dataset**	**Privacy**	**Reported result**
Belarbi *et al.* [52]	DNN	TON_IoT	Data locality	Centralized F1 $\approx 94\%$
Ruzafa-Alcázar *et al.* [44]	DNN	TON_IoT	FL + Differential Privacy	Reports utility cost as privacy budget ε tightens
FELIDS [46]	DNN / CNN / RNN	CSE-CIC-IDS2018	Data locality	Up to $\approx 94.15\%$ accuracy
DÏoT [25]	GRU per device	Lab IoT (Mirai)	Data locality	$\approx 95.6\%$ detection rate
Mothukuri *et al.* [45]	GRU + ensembler	Modbus IoT	Data locality	accuracy for: IID$\approx 97.5\%$, Non-IID$\approx 95.2\%$
Anwar *et al.* [43]	LSTM	IoT-WSN datasets	Data locality	accuracy$\approx 95.52\%$
FedShield	**MLP (49.9k)**	**TON_IoT**	**FL + Secure Aggregation**	**Acc.** 0.9584**, Macro F1** 0.8988

Four points emerge from this comparison. First, Belarbi *et al.* [52] also study a DNN-based federated IDS on TON_IoT. They report a centralized DNN F1-score of about 94%, while a randomly initialized federated DNN degrades under non-IID data and only approaches the centralized baseline after pre-training. By contrast, FedShield reaches a Weighted F1 of 0.9592 and a Macro F1 of 0.8988 under a strong Dirichlet split and without any pre-training step.

Second, Ruzafa-Alcázar *et al.* [44] use a DNN with FedAvg on the Industrial-IoT subset of TON_IoT and add Differential Privacy on top. They report a visible utility cost as the privacy budget becomes tighter. FedShield occupies a different point in the design space. It integrates mask-based Secure Aggregation rather than Differential Privacy and is empirically lossless in terms of detection performance, although the threat models of the two approaches are not directly equivalent.

Third, FELIDS [46] evaluates DNN, CNN, and RNN classifiers in a federated setting on the CSE-CIC-IDS2018 dataset and reports accuracies up to about 94.15%, without any privacy mechanism beyond data locality. FELIDS therefore provides a useful DNN-based

federated baseline, although on a different dataset than the one used in this thesis.

Fourth, recurrent-model approaches such as DÏoT [25], the GRU-based ensembler of Mothukuri *et al.* [45], and the LSTM-based FL-IDS of Anwar *et al.* [43] provide useful context. DÏoT reports a detection rate of about 95.6% on a Mirai-based testbed, Mothukuri *et al.* report about 97.5% accuracy under IID and 95.2% under non-IID partitions on Modbus IoT traffic, and Anwar *et al.* report about 95.52% accuracy on IoT-WSN datasets. These results are competitive, but they rely on heavier sequence-aware architectures. In this thesis, the input is a 31-dimensional tabular feature vector. For this type of data, MLP-style models remain reasonable and often competitive baselines on tabular problems [49].

Three caveats should be stated clearly. First, the cited studies use different datasets and different non-IID settings, so any apparent ranking is partly a function of dataset difficulty and partition strategy rather than of method quality. Second, the reported metrics are not always defined in the same way. Some papers report a detection rate, while others report multi-class accuracy or weighted F1. For this reason, the table should be read as a positioning exercise rather than as a strict leaderboard. Third, FedShield is evaluated under an honest-but-curious server threat model, which is weaker than the protection expected from a full cryptographic implementation of Secure Aggregation [33].

Limitations of the current study. Three limitations of the present empirical study are worth stating plainly. The current implementation is a single-run study with seed 42, without statistical-significance tests across multiple seeds. This is common in FL-IDS papers of comparable scope, but it reduces the strength of any quantitative claim. The mask seeds used in the SecAgg protocol are deterministic and therefore do not protect against a server that fully knows the seed scheme. A deployment would require a Diffie–Hellman-style key agreement, as in the original SecAgg design [33]. Finally, the experiments evaluate a single architecture (MLP), a single non-IID partitioning ($\alpha = 0.3$), and a single number of clients ($K = 10$). Ablations on these design choices would strengthen the empirical analysis and are left for future work.

4.6 Conclusion

This chapter presented FedShield, the experimental implementation of the federated learning framework proposed in this thesis for collaborative and privacy-preserving IoT intrusion detection. The framework was implemented in Python with TensorFlow and the Flower federated learning library [47]. It was trained on the TON_IoT dataset [4], which was partitioned across ten non-IID clients using a Dirichlet sampler with $\alpha = 0.3$. The local model is a lightweight Multi-Layer Perceptron with 49,929 parameters, trained with Focal Loss [50] and Adam [51], and aggregated through Federated Averaging [3] extended with a mask-based Secure Aggregation scheme inspired by Bonawitz *et al.* [33].

After thirty communication rounds, the final global model reaches an accuracy of 0.9584,

a Macro F1 of 0.8988, and a Weighted F1 of 0.9592 on a held-out test set of 4,467,594 samples across nine classes. The masking stage preserves the FedAvg aggregate up to `float32` rounding, which shows that this privacy-oriented layer can be added without measurable loss in detection performance. A deployable privacy guarantee, however, would require replacing the deterministic seed rule with a cryptographic key-establishment protocol. The main weak point of the model is the ransomware class, whose F1-score of 0.6536 is mainly explained by its limited support in the dataset.

Overall, the experiments support the main claim of the thesis. Distributed IoT clients can collaboratively predict security attacks without sharing raw network traffic, and secure-aggregation-style masking can be integrated without reducing detection performance in a measurable way. With a full cryptographic key-establishment layer, this design could be extended into a more deployable privacy-preserving solution. The broader conclusion of the thesis and the future directions opened by this work are discussed in the final part of the manuscript.

General Conclusion

The work presented in this thesis was motivated by a simple but increasingly urgent observation: as the Internet of Things expands across homes, industries, healthcare, and transportation, it also expands the attack surface that must be protected. Traditional intrusion detection approaches, whether rule-based or based on centralized machine learning, struggle to follow this evolution because they require sensitive network traffic to be collected and processed at a single point. This creates privacy, bandwidth, and regulatory concerns. The central question of the thesis was therefore to study how federated learning can be designed and implemented so that distributed IoT devices can collaboratively predict security attacks, while keeping raw network traffic local and while taking client heterogeneity into account.

To answer this question, the thesis followed a progressive path. Chapter 1 introduced the Internet of Things, its security challenges, the main types of attacks that target IoT environments, and the foundations of machine learning, deep learning, and federated learning. Chapter 2 reviewed the state of the art in IoT attack prediction and organized the literature into four families, from classical approaches to federated learning-based intrusion detection. It also identified the open challenges that motivated the proposed framework. Chapter 3 then introduced the proposed FedShield framework and its architectural components. Chapter 4 evaluated it on the TON_IoT dataset under a non-IID federated setting.

Summary of Contributions

The contributions of this thesis can be summarized on three levels: a theoretical and bibliographic contribution, a methodological contribution, and an experimental contribution.

- **Theoretical and bibliographic contribution.** The thesis offers a structured review of IoT intrusion detection. The review is organized around classical, machine learning-based, deep learning-based, and federated learning-based approaches. It is supported by comparative tables and by a taxonomy of federated learning-based intrusion detection systems for IoT. This synthesis provides a clear entry point for further work in the same area.
- **Methodological contribution.** The thesis proposes FedShield, a federated learning framework that combines known and validated components in a single coherent pipeline. The framework uses a lightweight Multi-Layer Perceptron with 49,929 trainable param-

eters, a Federated Averaging training loop [3], and a mask-based Secure Aggregation mechanism inspired by Bonawitz *et al.* [33]. The implementation is built on top of the Flower framework [47], which makes the design compatible with a production-grade federated learning stack.

- **Experimental contribution.**

  The framework is evaluated on the TON_IoT dataset [4], partitioned across ten non-IID clients with a Dirichlet split ($\alpha = 0.3$). The evaluation covers thirty communication rounds and uses a stratified train/test split computed before class capping. This keeps the test set representative of the original class distribution. The experiments show that the ten clients can collaboratively predict nine attack categories with strong detection performance, while their raw network traffic remains local.

Answers to the Research Questions

The experimental findings answer the five sub-questions formulated in the Introduction.

- **Feasibility of federated learning on a realistic IoT dataset.** After thirty communication rounds, the final FedShield model reaches an accuracy of 0.9584, a Macro F1 of 0.8988, and a Weighted F1 of 0.9592 on a held-out test set of 4,467,594 samples. A centralized baseline was not retrained from scratch in this thesis, so the comparison must remain cautious. However, these values are in the same range as strong TON_IoT results reported in recent federated IDS studies. They show that the federated configuration can achieve strong multi-class detection while keeping raw traffic local.

- **Behaviour under non-IID data.** The Dirichlet split with $\alpha = 0.3$ creates a strongly heterogeneous federation. Client sizes range from about 228,891 samples to 1,228,637 samples, and class mixtures vary clearly across clients. Despite this, the model converges quickly: accuracy rises from 0.8012 at round 1 to 0.9421 at round 4, and then stays in the $[0.95, 0.96]$ range after round 10. Some loss spikes appear at rounds 23, 26, and 29, which is consistent with the sensitivity of FedAvg to non-IID drift. The system nevertheless recovers from these fluctuations. The main effect of heterogeneity appears on minority classes, especially ransomware, whose F1-score is 0.6536.

- **Privacy–utility trade-off.** The privacy mechanism evaluated in this thesis is mask-based Secure Aggregation. At every round, the SecAgg-recovered weights are compared with a direct plain-text FedAvg aggregation of the same client updates. The maximum element-wise difference remains between 2.6×10^{-7} and 1.91×10^{-6}, which is at the level of `float32` rounding. In this experimental setting, the secure-aggregation layer therefore introduces no measurable utility loss. A broader comparison with other privacy mechanisms, such as differential privacy or homomorphic encryption, remains outside the scope of this thesis.

- **Deployability on resource-constrained IoT devices.** The local model contains 49,929 trainable parameters, which corresponds to about 195 KB in `float32` format. This compact size is compatible with the memory and communication constraints of many IoT gateways. The same model is sent to every client at each round, so the per-round communication cost remains bounded by this footprint. The results also show that a lightweight MLP can be effective on tabular flow features, without requiring heavier sequence-aware architectures.
- **Reproducibility and realism.** The full pipeline is implemented with Flower, using custom client and strategy classes that integrate mask generation, masked upload, aggregation, and verification into a standard FedAvg loop. This implementation keeps a clear separation between the client API, the server-side strategy, and the orchestration layer. It therefore provides a more realistic basis for future deployment than a purely manual simulation.

Limitations

Although the experimental results support the relevance of the proposed framework, several limitations should be acknowledged.

- **Simulation-based federation.** All experiments are conducted in a simulated federation. The ten clients are virtual processes running on slices of the same dataset. This is a common methodological choice in federated learning research, but it does not capture device drop-out, network jitter, firmware diversity, asynchronous arrivals, or energy constraints.
- **Rare-class difficulty.** The ransomware class is the only class whose F1-score falls clearly below the rest of the model. This is mainly a data-scarcity issue: the class has only 58,244 training samples and 14,561 test samples, and these samples are further split across ten clients by the Dirichlet sampler. Focal Loss partially reduces this effect but does not remove it. The MITM class is not included in the experiments because it contains only about one thousand samples in the full dataset, which is too few for reliable training and evaluation.
- **Simulated Secure Aggregation.** The masks used in the current SecAgg protocol are generated from deterministic seeds. This is enough to verify the algebraic behaviour of mask cancellation and to show that masking does not harm detection performance. It does not, however, provide the same protection as a deployable cryptographic protocol. A real implementation would require secure pairwise key agreement, such as Diffie–Hellman, as in the original Secure Aggregation design.

Future Work

The limitations above naturally suggest several directions for future work. They can be grouped into five families, each extending one of the research sub-questions answered in this thesis.

- **Deepening the heterogeneity analysis.** Cross-dataset evaluation on additional benchmarks, including CICIDS2017, Bot-IoT, and CICIoT2023, would help verify whether the framework generalizes beyond TON_IoT. Future work should also explore different non-IID partitioning strategies, different values of the Dirichlet concentration parameter, partitioning by device type, client drop-out, and concept drift. Replacing FedAvg with FedProx or FedNova would be a direct way to address heterogeneous clients.
- **Broadening the privacy–utility study.** The privacy component can be strengthened in several ways. A full cryptographic Secure Aggregation implementation, based on Diffie–Hellman key agreement or Shamir's secret sharing, would replace deterministic seeds with a deployable protocol. Adding a differential-privacy layer with an explicit privacy accountant would provide a complementary formal guarantee and would allow a direct comparison between privacy mechanisms. Byzantine-robust aggregation rules, such as Krum, the trimmed mean, or the median, would extend the threat model to malicious clients.
- **Pushing deployability further.** Deploying FedShield on real IoT devices, such as Raspberry Pi class boards or industrial edge gateways, would expose the framework to real network conditions and hardware constraints. Combined with model compression techniques such as quantization or knowledge distillation, this would provide a stronger answer to the deployability question.
- **Confirming reproducibility in realistic conditions.** The Flower-based implementation is already a step toward realistic deployment. A full client-server experiment with physically separated nodes, intermittent connectivity, and asynchronous client arrivals would be the natural next step. Such a setup would also make it possible to measure wall-clock training time, energy consumption, and recovery from network failures.

Bibliography

[1] Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. Computer Networks, 54(15), 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010

[2] Kolias, C., Kambourakis, G., Stavrou, A., & Voas, J. (2017). DDoS in the IoT: Mirai and other botnets. Computer, 50(7), 80–84. https://doi.org/10.1109/MC.2017.201

[3] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017), PMLR 54 (pp. 1273–1282).

[4] 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

[5] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

[6] Ashton, K. (2009). That 'Internet of Things' thing. RFID Journal, 22(7), 97–114.

[7] ITU-T (2012). Recommendation ITU-T Y.2060: Overview of the Internet of Things. International Telecommunication Union.

[8] Choudhary, A. (2024). Internet of Things: a comprehensive overview, architectures, applications, simulation tools, challenges and future directions. Discover Internet of Things, 4, 31. https://doi.org/10.1007/s43926-024-00084-3

[9] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 17(4), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095

[10] Sicari, S., Rizzardi, A., Grieco, L. A., & Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. Computer Networks, 76, 146–164. https://doi.org/10.1016/j.comnet.2014.11.008

[11] Alaba, F. A., Othman, M., Hashem, I. A. T., & Alotaibi, F. (2017). Internet of Things security: A survey. Journal of Network and Computer Applications, 88, 10–28. https://doi.org/10.1016/j.jnca.2017.04.002

[12] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., … Zhou, Y. (2017). Understanding the Mirai botnet. In 26th USENIX Security Symposium (USENIX Security 17) (pp. 1093–1110). USENIX Association.

[13] Zarpelão, B. B., Miani, R. S., Kawakani, C. T., & de Alvarenga, S. C. (2017). A survey of intrusion detection in Internet of Things. Journal of Network and Computer Applications, 84, 25–37. https://doi.org/10.1016/j.jnca.2017.02.009

[14] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3), 210–229. https://doi.org/10.1147/rd.33.0210

[15] Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

[16] Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 1153–1176. https://doi.org/10.1109/COMST.2015.2494502

[17] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

[18] Haykin, S. (2009). Neural networks and learning machines (3rd ed.). Pearson.

[19] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536. https://doi.org/10.1038/323533a0

[20] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

[21] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

[22] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[23] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1724–1734).

[24] Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology, 10(2), Article 12, 1–19. https://doi.org/10.1145/3298981

[25] Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., & Sadeghi, A.-R. (2019). DÏoT: A federated self-learning anomaly detection system for IoT. In 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS) (pp. 756–767). IEEE. https://doi.org/10.1109/ICDCS.2019.00080

[26] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., … 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

[27] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018). Federated learning with non-IID data. arXiv preprint arXiv:1806.00582. https://arxiv.org/abs/1806.00582

[28] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine, 37(3), 50–60. https://doi.org/10.1109/MSP.2020.2975749

[29] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. In Proceedings of Machine Learning and Systems (MLSys) (Vol. 2, pp. 429–450).

[30] Wang, J., Liu, Q., Liang, H., Joshi, G., & Poor, H. V. (2020). Tackling the objective inconsistency problem in heterogeneous federated optimization. In Advances in Neural Information Processing Systems (NeurIPS) (Vol. 33, pp. 7611–7623).

[31] Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In S. Halevi & T. Rabin (Eds.), Theory of Cryptography (TCC 2006), Lecture Notes in Computer Science, vol. 3876 (pp. 265–284). Springer. https://doi.org/10.1007/11681878_14

[32] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16) (pp. 308–318). ACM. https://doi.org/10.1145/2976749.2978318

[33] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., & Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17) (pp. 1175–1191). ACM. https://doi.org/10.1145/3133956.3133982

[34] Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09) (pp. 169–178). ACM. https://doi.org/10.1145/1536414.1536440

[35] Yao, A. C. (1982). Protocols for secure computations. In Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS '82) (pp. 160–164). IEEE. https://doi.org/10.1109/SFCS.1982.38

[36] Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., & Elovici, Y. (2018). N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Computing, 17(3), 12–22. https://doi.org/10.1109/MPRV.2018.03367731

[37] 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) (pp. 108–116). SciTePress. https://doi.org/10.5220/0006639801080116

[38] Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Future Generation Computer Systems, 100, 779–796. https://doi.org/10.1016/j.future.2019.05.041

[39] Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. Sensors, 23(13), 5941. https://doi.org/10.3390/s23135941

[40] Agrawal, S., Sarkar, S., Aouedi, O., Yenduri, G., Piamrat, K., Alazab, M., Bhattacharya, S., Maddikunta, P. K. R., & Gadekallu, T. R. (2022). Federated learning for intrusion detection system: Concepts, challenges and future directions. Computer Communications, 195, 346–361. https://doi.org/10.1016/j.comcom.2022.09.012

[41] Campos, E. M., Saura, P. F., González-Vidal, A., Hernández-Ramos, J. L., Bernabe, J. B., Baldini, G., & Skarmeta, A. (2022). Evaluating Federated Learning for intrusion detection in Internet of Things: Review and challenges. Computer Networks, 203, 108661. https://doi.org/10.1016/j.comnet.2021.108661

[42] Lavaur, L., Pahl, M.-O., Busnel, Y., & Autrel, F. (2022). The evolution of federated learning-based intrusion detection and mitigation: A survey. IEEE Transactions on Network and Service Management, 19(3), 2309–2332. https://doi.org/10.1109/TNSM.2022.3177512

[43] Anwar, R. W., Abrar, M., Salam, A., & Ullah, F. (2025). Federated learning with LSTM for intrusion detection in IoT-based wireless sensor networks: A multi-dataset analysis. PeerJ Computer Science, 11, e2751. https://doi.org/10.7717/peerj-cs.2751

[44] Ruzafa-Alcázar, P., Fernández-Saura, P., Mármol-Campos, E., González-Vidal, A., Hernández-Ramos, J. L., Bernal-Bernabe, J., & Skarmeta, A. F. (2023). Intrusion detection based on privacy-preserving federated learning for the industrial IoT. IEEE

Transactions on Industrial Informatics, 19(2), 1145–1154. https://doi.org/10.1109/TII.2021.3126728

[45] Mothukuri, V., Khare, P., Parizi, R. M., Pouriyeh, S., Dehghantanha, A., & Srivastava, G. (2022). Federated-learning-based anomaly detection for IoT security attacks. IEEE Internet of Things Journal, 9(4), 2545–2554. https://doi.org/10.1109/JIOT.2021.3077803

[46] 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

[47] 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. (2020). Flower: A friendly federated learning research framework. arXiv preprint arXiv:2007.14390. https://arxiv.org/abs/2007.14390

[48] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(56), 1929–1958.

[49] Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? In Advances in Neural Information Processing Systems (NeurIPS) (Vol. 35).

[50] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 2980–2988). https://doi.org/10.1109/ICCV.2017.324

[51] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations (ICLR). https://arxiv.org/abs/1412.6980

[52] O. Belarbi, T. Spyridopoulos, E. Anthi, I. Mavromatis, P. Carnelli, and A. Khan, "Federated deep learning for intrusion detection in IoT networks," arXiv preprint arXiv:2306.02715, 2023. https://arxiv.org/abs/2306.02715

[53] Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., & Sikdar, B. (2019). A survey on IoT security: Application areas, security threats, and solution architectures. IEEE Access, 7, 82721–82743. https://doi.org/10.1109/ACCESS.2019.2924045

[54] Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. SN Computer Science, 2(3), Article 160. https://doi.org/10.1007/s42979-021-00592-x

[55] Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A survey of machine and deep learning methods for Internet of Things (IoT) security. IEEE Communications Surveys & Tutorials, 22(3), 1646–1685. https://doi.org/10.1109/COMST.2020.2988293

[56] Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., Niyato, D., Dobre, O., & Poor, H. V. (2021). Federated learning for Internet of Things: A comprehensive survey. IEEE Communications Surveys & Tutorials, 23(3), 1622–1658. https://doi.org/10.1109/COMST.2021.3075439

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de La Recherche Scientifique

كلية العلوم والتكنولوجيا
قسم الرياضيات والإعلام الآلي

Faculté des Sciences et de la Technologie

Département Des Mathématiques & de l'Informatique

جامعة غرداية

الرقم ...../2026/ ق.ر.ا /ك.ع.ت/ج.غ

شهادة الترخيص بالإيداع

أنا الأستاذ: Mme. BRAHIM Nacera

بصفتي رئيس والمسؤول عن تصحيح مذكرة تخرج ماستر الموسومة بـــــ

A Federated Learning Framework For Collaborative and Privacy-Preserving IoT Security Attack Prediction

من انجاز الطالب(ة): ADDOUN Mohammed

والطالب(ة): FERTAS Mouad

الكلية: العلوم والتكنولوجيا.
القسم: الرياضيات والاعلام الالي.
الشعبة: اعلام الي.
التخصص: الأنظمة الذكية لاستخراج المعارف.
تاريخ التقييم/المناقشة: **22/06/2026**
أشهد ان الطالب (الطلبة) قد قام (قاموا) بالتعديلات والتصحيحات المطلوبة من طرف لجنة المناقشة وان المطابقة بين النسخة الورقية والالكترونية استوفت جميع شروطها.

امضاء المسؤول عن التصحيح

مصادقة رئيس القسم

جامعة غرداية
قسم الرياضيات والاعلام الآلي
كلية العلوم والتكنولوجيا

رئيس قسم الرياضيات والإعلام الآلي
الحاج موسى ياسين