

A Differential Privacy Aided Federated Learning Model for Intrusion Detection in IoT Networks

by

Sayedha Suaiba Anwar

Roll No: 20MCSE019P

A thesis submitted for the partial fulfillment of the requirement for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING



Department of Computer Science and Engineering(CSE)

CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Chattogram-4349, Bangladesh

NOVEMBER, 2024

CERTIFICATION

The thesis titled "**A Differential Privacy Aided Federated Learning Model for Intrusion Detection in IoT Networks**" submitted by **Sayedha Suaiba Anwar**, Roll No **20MCSE019P**, Session **2020-2021** has been accepted as satisfactory in partial fulfillment of the requirement for the degree of MASTER OF SCIENCE IN COMPUTER SCIENCE & ENGINEERING on 25/11/24.

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Author: Sayeda Suaiba Anwar

.....
(signature)

Date:

*It is my genuine gratefulness and warmest regard that
I dedicate this work to my beloved
Father and Mother*

Table of Contents

List of Figures	viii
List of Tables	x
List of Algorithms	xi
Terms and Abbreviation	xii
Acknowledgement	xiii
Abstract	xiv
1 Introduction	1
1.1 Introduction	1
1.2 Objective	2
1.3 Motivation	3
1.4 Challenges	3
1.5 Problem Statement	3
1.6 Contribution	4
1.7 Organization of the Thesis	4
1.8 Conclusion	5
2 Literature Review	6
2.1 Introduction	6
2.2 Federated Learning	6
2.2.1 Federated Averaging (FedAvg)	6
2.2.2 Federated Proximal (FedProx)	7
2.3 Differential Privacy	8
2.4 Epsilon(ϵ) and Delta(δ)	10
2.4.1 Epsilon(ϵ) - Privacy Budget	10
2.4.1.1 Theorem 1 : ϵ - Differential Privacy	10
2.4.2 Delta (δ) - Likelihood of Privacy Breach	12
2.4.2.1 Corollary: (ϵ, δ)-Differential Privacy	12
2.5 DL- based Intrusion Detection for Resource-constrained System	13
2.6 Federated Learning in Intrusion Detection	13
2.7 DP Aided FL for Intrusion Detection	15
2.8 Conclusion	16

3	Proposed Methodology/Framework/ Mechanism/ System Architecture	17
3.1	Introduction	17
3.2	Datasets	17
3.2.1	BoT-IoT Dataset	18
3.2.2	N-BaIoT Dataset	19
3.3	Data Preprocessing	19
3.3.1	One-Hot Encoding	20
3.3.2	Standardization	20
3.4	Balancing Imbalanced Data	20
3.4.1	Random Undersampling	20
3.4.2	Synthetic Minority Over-sampling Technique (SMOTE)	21
3.5	Data Distribution	21
3.6	Selection of Deep Learning Model	21
3.7	Proposed BiLSTM-CNN Architecture	22
3.8	Integration of Differential Privacy	25
3.8.1	Evaluation	26
3.9	Conclusion	26
4	Implementation	27
4.1	Introduction	27
4.2	Implementation of BiLSTM-CNN in FL Architecture	27
4.3	Data Distribution Strategy	28
4.4	Implementation of Differential Privacy in FL Architecture	28
4.5	Conclusion	29
5	Experimental Result analysis and Evaluation	30
5.1	Introduction	30
5.2	Experimental Setup	30
5.3	Performance Metrics	31
5.3.1	Accuracy	31
5.3.2	Loss	31
5.3.3	Privacy Parameters	32
5.3.3.1	Epsilon (ϵ)	32
5.3.3.2	Delta (δ)	32
5.4	Result Analysis for BoT-IoT dataset	32
5.4.1	Variation of Accuracy with respect to Communication Rounds	32
5.4.2	Variation of Loss with respect to Communication Rounds	34
5.4.3	Variation of Accuracy with respect to Communication Rounds for varying noise multipliers	35
5.4.4	Variation of Accuracy with respect to Epsilon for varying delta	40
5.5	Result Analysis for N-BaIoT dataset	41

5.5.1	Variation of Accuracy with respect to Communication Rounds . .	41
5.5.2	Variation of Loss with respect to Communication Rounds	43
5.5.3	Variation of Accuracy with respect to Communication Rounds for varying noise multipliers	44
5.5.4	Variation of Accuracy with respect to Epsilon for varying delta . .	47
5.6	Comparison with Existing Research Work	48
5.7	Conclusion	50
6	Conclusion	51
	Bibliography	53

List of Figures

2.1	Overview of Differential Privacy Mechanism in Industrial IoT	9
3.1	Proposed System Architecture	17
3.2	Proposed BiLSTM-CNN Architecture	23
4.1	Visualization of label distribution skew in the BoT-IoT dataset	28
4.2	Proposed Differential Privacy aided FL-Architecture	29
5.1	Schematic View of Implementation Setup	31
5.2	Change of Accuracy vs. No. of Communication Rounds for FedAvg . . .	33
5.3	Change of Accuracy vs. No. of Communication Rounds for FedProx . . .	33
5.4	Change of Accuracy with respect to Number of Clients for FedAvg	34
5.5	Change of Accuracy with respect to Number of Clients for FedProx . . .	34
5.6	Change of Loss with respect to No. of Communication Rounds for FedAvg	35
5.7	Change of Loss with respect to No. of Communication Rounds for FedProx	35
5.8	Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=5$	36
5.9	Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=15$	36
5.10	Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=25$	37
5.11	Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=5$	37
5.12	Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=15$	38
5.13	Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=25$	38
5.14	Change of Epsilon with respect to Accuracy for different values of delta for FedAvg	40
5.15	Change of Epsilon with respect to Accuracy for different values of delta for FedProx	40
5.16	Change of Accuracy vs. No. of Communication Rounds for FedAvg . . .	41
5.17	Change of Accuracy vs. No. of Communication Rounds for FedProx . . .	41
5.18	Change of Accuracy with respect to Number of Clients for FedAvg	42
5.19	Change of Accuracy with respect to Number of Clients for FedProx . . .	42
5.20	Change of Loss with respect to No. of Communication Rounds for FedAvg	43
5.21	Change of Loss with respect to No. of Communication Rounds for FedProx	43

5.22	Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=5$	44
5.23	Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=15$	44
5.24	Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=25$	45
5.25	Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=5$	45
5.26	Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=15$	46
5.27	Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=25$	46
5.28	Change of Epsilon with respect to Accuracy for different values of delta for FedAvg	48
5.29	Change of Epsilon with respect to Accuracy for different values of delta for FedProx	48

List of Tables

3.1	Statistics of BoT-IoT dataset	18
3.2	Statistics of N-BaIoT Dataset	19
3.3	Comparison of accuracy among different DL models	22
3.4	Configuration of Proposed BiLSTM-CNN Model	24
4.1	Hyperparameters Used in BiLSTM-CNN Model	27
4.2	Parameters for DP-FL Architecture	29
5.1	Change of Accuracy for different noise multipliers for varying number of clients using FedAvg on BoT-IoT dataset	39
5.2	Change of Accuracy for different noise multipliers for varying number of clients using FedProx on BoT-IoT dataset	39
5.3	Change of Accuracy for different noise multipliers for varying number of clients using FedAvg on N-BaIoT dataset	47
5.4	Change of Accuracy for different noise multipliers for varying number of clients using FedProx on N-BaIoT dataset	47
5.5	Comparison with existing works using BoT-IoT and N-BaIoT datasets . .	49

List of Algorithms

1	DP-enabled FL Architecture using FedAvg	25
2	DP-enabled FL Architecture using FedProx	26

List of Terms and Abbreviations

IoT	Internet of Things
IDS	Intrusion Detection System
DL	Deep Learning
ML	Machine Learning
FL	Federated Learning
FedAvg	Federated Averaging
FedProx	Federated Proximal
DP	Differential Privacy
SGD	Stochastic Gradient Descent
SMOTE	Synthetic Minority Over-sampling Technique
Non-IID	Non Independent & Identically Distributed
RNN	Recurrent Neural Network
FLOPS	Floating Point Operations
CNN	Convolutional Neural Network
BiLSTM	Bidirectional Long Short Term Memory
η	Learning Rate
E	Local Epochs
C	Fractions of clients participating in training
B	Mini-batch size
λ	Potential Learning Rate decay
μ	Proximal Term
ϵ	Privacy Loss
δ	Likelihood of a Privacy Breach

Acknowledgement

I am overwhelmed to get this opportunity to express my gratitude to those whose constant support and encouragement lead me to the completion of my thesis on Intrusion Detection in Inter of Things (IoT) System. First and foremost, I feel immense proud to express my heartfelt respect and deepest sense of gratitude to my supervisor, Prof. Dr. Mohammed Moshiul Hoque and co-supervisor Prof. Dr. Asaduzzaman for their scholarly guidance, instruction, constructive criticism, valuable suggestions, supervision, inspiration, untiring assistance in all phases of the research work and in the preparation of the report manuscript. Without their pedagogic supervision, heartfelt help, unfailing attention, and constructive criticism this report would not have seen the light of the day. I am ever grateful to the honorable examination board members for their valuable suggestion, helpful comment, and affectionate feeling.

I am incredibly grateful to all the faculty members and the staff of department of CSE, CUET for their support. My friends, colleagues deserves special thanks for their assistance to make this work possible. I express my cordial respect and sincere appreciation to my parents and my husband for their unforgettable kind co-operation, constructive suggestions, sacrifice, and cordial support in all spheres of the working period.

Finally, all praises to Allah, the Almighty because it is He, who give me the strength and courage to accomplish the task and achieve the goal.

Abstract

In the rapidly-developing Internet of Things (IoT) ecosystem, safeguarding the privacy and accuracy of linked devices and networks is of utmost importance, with the challenge lying in effective implementation of Intrusion Detection Systems (IDS) on resource-constrained IoT devices. This study introduces a Differential Privacy-Aided Federated Learning architecture for intrusion detection in IoT contexts as a novel approach to addressing these difficulties. To build an intrusion detection model, we combined components of a Convolutional Neural Network (CNN) with Bidirectional Long Short-Term Memory (BiLSTM). We apply this approach to the Bot-IoT dataset, which was rigorously curated by the University of New South Wales (UNSW) and N-BaIoT dataset. Our major goal is to create a model that delivers high accuracy while protecting privacy, an often overlooked aspect of IoT security. Intrusion detection tasks are distributed across multiple IoT devices using federated learning principles to protect data privacy, incorporating the differential privacy framework to gauge and minimize information leakage, all while investigating the intricate relationship between privacy and accuracy in pursuit of an ideal compromise. We implemented two aggregation methods, namely, Federated Averaging (FedAvg) and Federated Proximal (FedProx). The trade-off between privacy preservation and model accuracy is investigated by adjusting the privacy loss and noise multiplier. Our research enhances IoT security by introducing a deep learning model for intrusion detection in IoT devices, explores the integration of differential privacy in federated learning framework for IoT and offers guidance on minimizing the accuracy- privacy trade-off based on specific privacy and security needs. Our study explores the privacy-accuracy trade-off by examining the effects of varying epsilon values on accuracy for various delta values for a range of clients between 5 to 25. We also investigate the influence of several noise multipliers on accuracy and find a consistent accuracy curve, especially around a noise multiplier value of about 0.5 for both the aggregation methods and with a higher accuracy for FedProx. The findings of this study have the possibilities to enhance IoT ecosystem security and privacy, contributing to the IoT landscape's trustworthiness and sustainability.

Keywords: Federated Learning, Differential Privacy, Intrusion Detection, BiLSTM-CNN, FedAvg, FedProx

Introduction

1.1 Introduction

Internet of Things (IoT) is a paradigm which envisions a world where objects may communicate with one another via networking features and interpret context with the aid of sensors. There are many reasons why IoT technology is being incorporated into systems, including better customer engagement, waste reduction, improved data collection, and technology optimization [1]. The range and intricacy of networked systems have greatly increased in recent years due to the growth of IoT devices, improving a number of elements of daily living and business operations. IoT networks are distinguished by their extensive deployment, varied application areas, and heterogeneous device capabilities. These attributes together make them vulnerable to security risks, such as malicious intrusions and unauthorized access. Strong security measures are required to protect sensitive data and guarantee that IoT ecosystems continue to function as they continue to grow.

Intrusion Detection Systems (IDSs) are an acknowledged strategy for identifying IoT attacks and cyber-threats early on to address such concerns. IDS is a technology which ensures security by continuously scanning network and host traffic for any suspicious activity, which may breach security policy and may jeopardize data confidentiality and integrity [2]. Conventional methods for detecting intrusions in IoT networks frequently depend on anomaly detection or signature-based approaches. These approaches, while somewhat successful, struggle to deal with the dynamic and heterogeneous character of IoT environments, where network circumstances and device behavior might differ greatly. Moreover, the implementation of traditional IDS is made more difficult by the privacy issues related to tracking and evaluating IoT data.

The IDSs which are Machine Learning (ML) based, rely greatly on feature engineering to extract relevant information from network data [3]. Deep Learning (DL)-based IDS can automatically learn complex features without the requirement for feature engineering because the raw data has a complex structure. AI-based IDS training creates privacy issues because it can require sharing of sensitive data across numerous domains [4][5].

To address these issues, there is a rising interest in utilizing cutting-edge technologies like Differential Privacy and Federated Learning to improve the security and privacy-preserving

capabilities of IoT systems. Federated learning (FL), a collaborative learning strategy that provides an alternative to conventional centralized techniques, was introduced in 2016. FL addresses privacy concerns related to data sharing by enabling distributed model training across dispersed IoT devices without the need for centralized data aggregation. In FL, local training of an AI algorithm takes place on clients, which are decentralized edge devices, and parameters on a global model are updated frequently through training rounds. Clients exchange their parameters with an aggregator, which uses them to create a global model, in place of disseminating their data [6]. Recent research, however, highlights the need to adopt privacy-preserving strategies to defend against potential assaults brought on by the sharing parameters used during training cycles. FL protects the privacy of individual data since the data remains on the device and only model updates are shared. But accuracy of the final model heavily depends on the quality of local models' training as the devices may have varying amounts and quality of data. Though FL is concerned about data privacy, aggregated updates from multiple devices can potentially reveal information about individual training data if not properly secured or anonymized. An aggregator with malicious intent, for instance, could alter the parameters it receives in order to deceive the model it is training. In the context of IoT, this could have serious consequences because the network data needed to build an IDS could be sensitive [7]. The trained model's outputs and the model parameters that were traded during training continue to offer possible attack surfaces for privacy leaks. To ease these concerns related to privacy, Differential Privacy (DP) [4], which provides statistical privacy assurances over data against an attacker, can be utilized to disguise training data or model updates. In order to make it hard to determine from the results of a particular function applied to the entire dataset whether or not a particular batch of data was used, DP depends on the dataset being supplemented with random noise. For the purpose of safeguarding sensitive data in IoT environments, DP offers a rigorous framework for quantifying and ensuring privacy assurances during data analysis.

1.2 Objective

Objectives of this research work include -

- To develop a differential privacy aided federated learning model for intrusion detection which will ensure user data privacy in a distributed way.
- To develop a hybrid deep learning model in federated learning scenario to improve the performance of the proposed model as ensuring data privacy, to some extent, degrades the performance of any model.
- Investigate and compare the proposed work with other existing FL based research works to analyze how accuracy-privacy trade-off is minimized.

1.3 Motivation

The investigation is driven by the need to solve cybersecurity issues specific to dynamic, smart IoT systems. These systems' interconnection and the variety of IoT devices they contain make them vulnerable to attacks from malevolent parties. The primary motivation is the requirement to provide efficient and flexible IDS techniques that can function within the limitations of IoT edge devices. The requirement to provide techniques that can safeguard sensitive data while guaranteeing efficient intrusion detection stems from the increased awareness of privacy issues and security threats related to IoT data. Data privacy is at threat when centralizing data for training, which is why privacy-preserving strategies like differential privacy must be investigated. Finding a compromise between protecting data privacy and getting the best possible performance out of the intrusion detection model is the key motivating factor. This demands creative solutions that can provide strong security results while upholding IoT device users' right to privacy. To put it briefly, the goal of this research is to create flexible, precise, and privacy-conscious intrusion detection techniques that are specifically designed to meet the demands of dynamic smart IoT systems.

1.4 Challenges

To address the cybersecurity issues in dynamic smart IoT systems, researchers have been investigating adaptable methods. Recent studies have focused on the predictability of ML and DL-based IDS. Due to the fact that the accuracy of intrusion detection increases with training data, they have attracted a great deal of attention. Edge devices often have minimal memory, computing power, and battery life. Due to resource limitations, privacy concerns, and security issues, centralizing the data for training is not a good option. Furthermore, a compromise is seen between data privacy and the model's accuracy in FL architectures.

1.5 Problem Statement

The fast proliferation of IoT devices needs efficient IDS that balance privacy and accuracy. FL approaches like FedAvg and FedProx can improve privacy by decentralizing model training. However, their efficacy varies. Integrating Differential Privacy (DP) into these systems offers another degree of confidentiality, although it may affect accuracy. This research seeks to compare FedAvg and FedProx in the context of Differential Privacy to investigate how they affect the privacy-accuracy trade-off in IoT intrusion detection to determine the most effective technique for maintaining robust privacy while assuring high detection accuracy. Therefore, The challenge is to optimize the trade-off between privacy and accuracy. Mathematically, this can be framed as:

$$\text{Maximize } A(\theta) \text{ subject to } \epsilon \text{ for given random noise}$$

where random noise affects the privacy parameter ϵ and consequently the model accuracy, $A(\theta)$. The objective is to find the optimal combination of random noise (for DP) and method choice (FedAvg or FedProx) that best balances privacy and accuracy.

1.6 Contribution

Analyzing the current state of privacy concerning IDS, the key research questions which are investigated in this research work are:

- “RQ1: How can an intrusion detection system be developed which will ensure user data privacy in a distributed way?”
- “RQ2: Will the proposed model be able to minimize the tradeoff between performance and privacy?”

In order to respond to the research questions mentioned, an intrusion detection system based on Federated Learning with Differential Privacy assistance employs a hybrid deep learning model. This study makes use of the BoT IoT dataset [8] and N-BaIoT [9] dataset. The following sums up this research’s contributions:

1. Develop a hybrid model exploiting CNN and BiLSTM models to detect intrusions in IoT systems in FL scenarios.
2. Demonstrate a method for achieving a trade-off between accuracy and privacy by suggesting appropriate privacy standards customized to satisfy various security and privacy requirements.
3. Investigate two aggregation techniques (FedAvg and FedProx) to find the more efficient strategy with better accuracy.

So far, all the research works on intrusion detection with federated learning are focused on either developing prediction models for edge devices or enhancing the data privacy of the clients. This research work focuses on both the aspects to observe if the accuracy-privacy tradeoff can be minimized or not.

1.7 Organization of the Thesis

The rest of the thesis is organized as follows -

Chapter 2 : Literature Review - This chapter covers federated learning (FL) and its application in intrusion detection and discusses integrating differential privacy with FL. This chapter also reviews impacts of deep learning methods in intrusion detection on resource-constrained IoT devices.

Chapter 3 : Proposed Methodology - This chapter describes the architecture of the DP-aided FL architecture tailored for intrusion detection in IoT networks. There is discussion on the design principles, components, and workflows, including how FL and DP are

integrated to ensure effective and privacy-preserving intrusion detection.

Chapter 4 : Implementation - This chapter summarizes the implementation details of the proposed system architecture.

Chapter 5 : Experimental Result analysis and Evaluation - This chapter provides the details of evaluation and outcomes along with a comparison between the suggested strategy and current relevant research.

Chapter 6 : Conclusion - This chapter provides the concluding statements along with future works, in short.

1.8 Conclusion

This chapter briefly describes the importance of intrusion detection systems in IoT paradigm. Motivation for using Federated Learning and Differential Privacy in Intrusion Detection are also described in this chapter. After that, contributions to our thesis work are outlined. The next chapter illustrates the literature review of this thesis work.

Literature Review

2.1 Introduction

To know about different existing research works on intrusion detection systems in IoT, a concise overview is described here. This overview illustrates basic concepts of federated learning, federated aggregation methods namely, federated averaging (FedAvg) and federated proximal (FedProx) and Differential Privacy. Then a literature review on Deep Learning based intrusion detection is provided. Finally, existing research work in Differential Privacy-Federated Learning based intrusion detection is introduced here.

2.2 Federated Learning

All personal data, including delicate personal data, can be stored on the device via federated learning, a distributed training technique that enables individuals in various geographic locations to collaborate with one another to develop machine learning models [10]. It is presumed that each of the N users $\{U_1, U_2, \dots, U_N\}$ owns their own database $\{D_1, D_2, \dots, D_N\}$, and they are all prohibited from directly accessing one another's data to enhance their own database.

It contains three basic steps:

1. Each device receives the initial model from the server.
2. Using local data D_i , the device U_i can federally train its own model W_i without having to disclose its own source data.
3. The server replaces each user's local model with the updated global model W' after aggregating the gathered local models $\{W_1, \dots, W_n\}$.

2.2.1 Federated Averaging (FedAvg)

Federated Averaging (FedAvg) [6] is a decentralized algorithm for training models across various devices or servers while keeping training data locally. FedAvg requires that every device separately computes an update to the model and then sends only the modified parameters to the server. The weights of these updates are determined by the quantity of

data points on each device, and the server uses a weighted average to combine them. This method is continued iteratively until the model converges. FedAvg is especially effective in privacy-sensitive situations where data cannot be centralized since it enables collaborative model training without requiring raw data sharing. FedAvg blends server-side model averaging with client-side Local Stochastic Gradient Descent (SGD). Model averaging is comparable to dropout training in that it involves creating an average model using the shared parameters of various customers. The FedAvg approach requires 10-100 times fewer communication rounds than synchronous SGD.

Nevertheless, SGD, for example, is used locally on clients to perform the real optimization. The five hyperparameters for FedAvg are the learning rate η , local epochs E , the fraction of clients participated in training C , local mini-batch size B , and potential learning rate decay λ . While training with SGD, the variables B , E , η , λ and C are frequently employed. E denotes the total iteration numbers through the local data in this case, prior to updating the global model. The global model W_0 is randomly initialized by the algorithm. Following that, a FedAvg communication round consists of the following: The server transmits the current global model W_t to every client in a client subset S_t , $|S_t| = C \cdot K \geq 1$. Every client obtains shared model $W_t^k \leftarrow W_t$ from their local models W_t^k , divides its local data into batches of size B , and then performs SGD of E epochs. The server creates a new global model W_{t+1} by summing up all of the incoming local models, weighted by the clients' trained local models W_{t+1}^k . The amount of local training instances affects the weighting scheme.

2.2.2 Federated Proximal (FedProx)

FedProx (Federated Proximal) [11] technique was developed to tackle the difficulties associated with optimizing federated learning models, especially in situations where the computational power and data distribution of client devices, or nodes, may vary substantially. FedProx can be considered as a re-parameterization and generalization of FedAvg. Convergence is ensured by this strategy when learning over data from different distributions. The processing hardware, network connections, and battery levels of various devices in federated networks are frequently subject to varying resource limits. As a result, expecting each device to do the same amount of work is unreasonable. The key to allowing flexible performance in this scenario is that each of the local objectives can be solved inexactly. This permits the amount of local computation versus communication to be adjusted depending on the number of local iterations accomplished (with more local iterations equating to more precise local solutions). FedAvg is generalized in FedProx by allowing varied amounts of work to be conducted locally across devices dependent on their available system resources, and then aggregating the partial solutions delivered by the stragglers (rather than eliminating these devices).

The γ inexactness measures the amount of local computation performed by the local solver

throughout each round. At the t -th round, γ_k^t measures the amount of local computation used to solve the local subproblem on device k . While allowing for nonuniform quantities of work to be performed across devices might help mitigate the negative effects of system heterogeneity, too many local updates may still lead the approaches to diverge due to the underlying heterogeneous data. A proximal term, μ is introduced to effectively limit the impact of variable local updates. The proximal term is advantageous in two ways: (1) It tackles statistical heterogeneity by limiting local updates to be closer to the starting (global) model, eliminating the need to manually adjust the number of local epochs. (2) It enables the safe incorporation of varied amounts of local effort resulting from system heterogeneity. μ can be adjusted according to the model's current performance. For example, one simple heuristic is to increase μ when seeing the loss increasing and decreasing μ when seeing the loss decreasing.

2.3 Differential Privacy

According to recent studies, privacy leakage is still a concerning issue when it comes to model gradient data. Differential Privacy is widely utilized in FL due to its practical quantitative examination of privacy and strong mathematical theory. This contributes to enhancing gradient transmission security even more. Currently in use are techniques for protecting privacy such as secure Multi-Party Computing (MPC), Differential Privacy (DP), and Homomorphic Encryption (HE). Large-scale, low-power FL activities cannot be conducted using homomorphic encryption and secure multi-party computing due to their high computational complexity. However, these methods can effectively secure user privacy. Due to its minimal processing overhead and good privacy quantification properties, differential privacy has found extensive use in real-world applications. Finding a compromise between model's efficiency and privacy protection is often necessary due to noise [12]. In the domain of statistics and data analysis, Differential Privacy is a framework for protecting privacy. Its main idea is to allow for the valuable information to be extracted from datasets while maintaining the privacy of each individual contributor. It becomes challenging to ascertain whether any particular person's information is included in the dataset due to the controlled noise or randomness that is introduced into the data prior to analysis. For ensuring that statistical insights may be obtained without jeopardizing the confidentiality of sensitive information, a balance between data utility and individual privacy must be struck. Concerns over the privacy of personal information have grown, which has made differential privacy important. As such, it is now a useful tool for academics and organizations looking to analyze datasets while protecting the privacy of the persons involved. For FL to use differential privacy, Geyer et al. [13] customized and added noise to the overall model. The findings demonstrate that differential privacy has essentially no effect on model accuracy when a large number of users are involved. It is used in cloud computing to safeguard the security and privacy of the multi-cloud union.

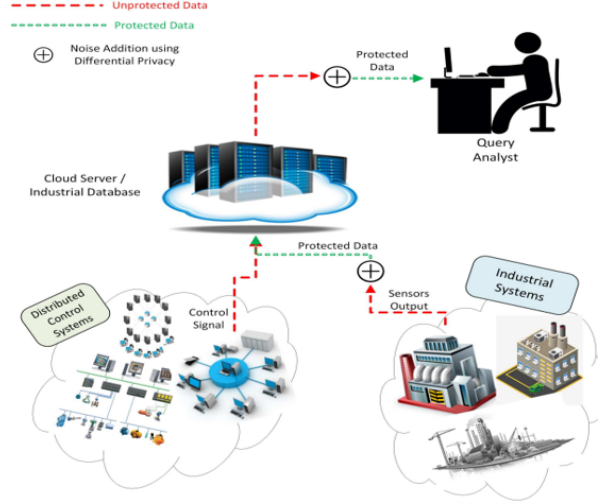


Figure 2.1. Overview of Differential Privacy Mechanism in Industrial IoT

McMahan et al. [14] moved the tailoring operation of the model to the user's local device and demonstrated that the cost of DP, when the data volume of the user is sufficiently big, is computational power rather than model accuracy. The findings of this study are restricted to the assumption that there are vast volumes of people or data; in any other case, the model's accuracy would be considerably reduced. It's possible that this condition doesn't occur in the real environment, though. The intermediate parameter w_i has to be exchanged with the server even though the i -th client's individual dataset D_i is kept locally in FL. This could reveal the clients' confidential data, as shown by model inversion attacks. Two concepts form the basis of the formal definition of DP [4].

Definition 1 (Mechanism): Mechanism can be defined as the generation of a random variable $\kappa(D)$ from a dataset D by a random function κ . For instance, if the input is a dataset of IoT attacks the output could be the flow time plus noise from a typical normal distribution. The weights of a model will be considered as the inputs in this particular case.

Definition 2 (Distance): Distance between two datasets, D and D' can be defined as the number of sample changes which converts D into D' . Such as, if D and D' disagree on at most one individual, then $d(D, D') = 1$. This type of pair of datasets is also considered as neighbors.

Definition 3 (Differential Privacy): For two data sets D and D' , using randomization algorithm κ , with just one instance difference and any possible output S , if procedure κ is satisfied:

$$\Pr(\kappa(D) \in S) \leq e^\epsilon \cdot \Pr(\kappa(D') \in S) + \delta \quad (2.1)$$

It is possible to view the randomization algorithm κ as meeting (ϵ, δ) -differential privacy. The algorithm's permissible error is represented by the parameter δ , while the privacy budget is indicated by the value of ϵ . The additive term δ is not included in the original

definition of "differential privacy". The suggested study strategy makes use of the variation developed in [15] to account for the possibility that plain-differential privacy may be compromised with probability (preferably, less than $1/|D|$).

According to the definition, the more privacy protection there is, the harder it will be for an attacker to tell apart two sets of data because of the increasing similarity of the probability distribution of the randomization algorithm κ operating on the two adjacent data sets. The outcomes of the two data sets using the randomization procedure κ are identical in the extreme scenario where $\epsilon = 0$. However, as the value increases, the degree of privacy protection decreases. It should be mentioned that the approach will introduce too much noise for practical use if ϵ is too tiny. The selection represents a trade-off between the level of privacy protection and the randomization algorithm κ 's applicability. Differentiable privacy can be categorized as a Gaussian, exponential, or Laplace mechanism based on the additional noise. We employ a Gaussian noise mechanism [16] appropriate for numerical data because the model update's gradient information is made up of a numerical matrix, such as

$$M(D) = f(D) + N(0, S_f^2 \sigma^2) \quad (2.2)$$

$N(0, S_f^2 \sigma^2)$ indicates a normal Gaussian distribution whose mean value is 0 and standard deviation is $S_f \sigma$. The value refers to the highest possible absolute distance between $|f(D) - f(D')|$. S_f reflects the global sensitivity of the function. Authors in [15] showed that when $\delta \geq \frac{4}{5} \exp(-(\sigma \epsilon)^2/2)$ and $\epsilon < 1$, the function satisfies (ϵ, δ) -DP.

2.4 Epsilon(ϵ) and Delta(δ)

In differential privacy, ϵ and δ are key parameters that control the privacy guarantees provided by the mechanism, specifically how much an individual's data can influence the results of a query while preserving privacy. Here's a detailed explanation of each:

2.4.1 Epsilon(ϵ) - Privacy Budget

The ϵ parameter [15] is the primary measure of privacy in differential privacy. It defines the privacy loss when the result of a query is published, reflecting the trade-off between privacy and accuracy. The ϵ in the theorem quantifies how much the inclusion or exclusion of a data point can affect the mechanism's output. A smaller ϵ means better privacy protection because it reduces the likelihood that any single individual's data will impact the outcome.

2.4.1.1 Theorem 1 : ϵ - Differential Privacy

A randomized algorithm κ is ϵ -differentially private if for all pairs of datasets D and D' that differ by a single element (i.e., D and D' are neighbors), and for all possible outputs

S , the following inequality holds:

$$\frac{Pr(\kappa(D) \in S)}{Pr(\kappa(D') \in S)} \leq e^\epsilon \quad (2.3)$$

where:

- $Pr(\kappa(D) \in S)$ is the probability that the mechanism κ outputs S when given input dataset D .
- $Pr(\kappa(D') \in S)$ is the probability that the mechanism κ outputs S when given input dataset D' .

An intuitive explanation of why ϵ -differential privacy is useful and how the ϵ parameter functions mathematically is provided here which will focus on the theorem's components [15].

Privacy Sensitivity

The inequality in the definition of ϵ -differential privacy measures how much the inclusion or exclusion of a single individual's data can affect the output of the mechanism. If this ratio is too large, it means that the mechanism is very sensitive to the data of a single individual, and their privacy could be compromised.

Adding Noise to Control Sensitivity

To achieve privacy, differential privacy typically introduces random noise into the output. This noise ensures that the result of the mechanism remains statistically similar regardless of whether a single individual's data is included in the dataset.

- If the ratio $\frac{Pr(\kappa(D) \in S)}{Pr(\kappa(D') \in S)}$ is close to 1, it means the presence or absence of a single individual's data has little to no effect on the output of $\S$.
- If the ratio is greater than e^ϵ , the result of the mechanism becomes more dependent on the inclusion of a particular individual's data, and privacy is compromised.

Exponential Function, e^ϵ

The factor e^ϵ is crucial because it controls how much the probabilities can differ. For ϵ -differential privacy to hold, the probability of any outcome S should not change too much depending on whether a particular individual is included in the dataset. The term e^ϵ ensures that the change is bounded.

- **Small Epsilon:** The term e^ϵ close to 1, meaning that the mechanism's output is nearly independent of any individual's data. A smaller value of epsilon corresponds to

stronger privacy. This means that the mechanism is less likely to reveal information about any individual's data. With smaller ϵ , the noise added to the result of the query is larger, making the output less sensitive to any individual's participation.

- **Large Epsilon:** The term e^ϵ grows larger, meaning that the mechanism's output becomes more sensitive to changes in individual data. A larger epsilon means weaker privacy protection. The mechanism is more likely to reveal information about an individual's data, as the noise added to the query result is smaller, making it more accurate but less private.

Thus, ϵ serves as the baseline control on privacy, ensuring that the probability ratio between the outputs of two neighboring datasets remains within an exponentially bounded range, protecting individual privacy while allowing for meaningful analysis on the dataset as a whole.

2.4.2 Delta (δ) - Likelihood of Privacy Breach

The δ parameter [17] is used in (ϵ, δ) -differential privacy to allow for some small probability of a "privacy breach" — meaning, with probability at most δ , the mechanism may violate the privacy guarantee of ϵ . In other words, δ represents the probability that the privacy guarantee does not hold. While ϵ determines how much privacy is preserved most of the time, δ allows for a slight relaxation, meaning there could be rare cases where the privacy condition is not satisfied (i.e., the privacy breach occurs). In some cases, the privacy guarantee may be relaxed slightly to allow for a small probability of a privacy breach. This is formalized as (ϵ, δ) -differential privacy.

2.4.2.1 Corollary: (ϵ, δ) -Differential Privacy

A randomized mechanism κ satisfies (ϵ, δ) -differential privacy if for all neighboring datasets D and D' , and for all possible outputs S , the following condition holds:

$$\frac{\Pr(\kappa(D) \in S)}{\Pr(\kappa(D') \in S)} \leq e^\epsilon + \delta \quad (2.4)$$

Here

- δ is a small positive number that controls the probability of a privacy violation. This allows for some probability of failure, which is useful when strict differential privacy is difficult to achieve without sacrificing too much utility [17]
- In this case, the mechanism will work as expected for almost all inputs with probability at least $1 - \delta$. The mechanism could, with probability δ , output something that violates the differential privacy guarantee [17].

Smaller delta:

A smaller δ means that such violations (privacy breaches) are less likely. Ideally, we want

δ to be very small, even approaching zero, so the privacy breach is negligible.

Larger delta:

A larger δ means that the probability of a privacy violation is higher, which weakens the overall privacy guarantee.

2.5 DL- based Intrusion Detection for Resource-constrained System

Recent years have seen a notable number of studies on the efficacy of Deep Learning (DL) in intrusion detection. Nevertheless, implementation challenges are rarely discussed. DL-based techniques in particular suffer from huge model sizes and high model complexity. IoT devices have limited storage and processing power, thus installing more complex intrusion detection models is not practicable [18]. High classification performance, simple models, and minimal model sizes should all be features of NIDS for IoT [19].

At the beginning of the research, conventional ML techniques were used. Although they offer a useful answer for a sizable volume of traffic data, these conventional ML techniques have restricted performance due to shallow learning. Numerous DL-based models have demonstrated outstanding performance as DL has progressed. Yin et al. [20] suggested a RNN based IDS. Regarding detection rate and accuracy for both binary and multiclass classification, the method outperformed traditional methods. He et al. [21] utilized LSTM and multimodal deep autoencoder to build an IDS which achieved more accurate detection rate through combining traffic features. Garg et al. [22] proposed an IDS based on CNN using Gray Wolf Optimization (GWO). The experimental findings demonstrate that using DARPA98, KDD99, and synthetic data sets, this strategy can achieve minimal features and improved detection performance. Xu et al. [23] developed the $\log - \cosh$ conditional variational autoencoder (CVAE), which is more effective in producing different intrusion data for classes with imbalances because it can represent the complicated distribution of observed data. Popoola et al. [24] presented an LSTM autoencoder (LAE) technique coupled with the DL algorithm for IoT intrusion detection. Using AE, low-dimensional traffic data is first reduced in dimension before being fed to a classifier for classification. To reduce the model's complexity, Mirsky et al. [25] divided traffic data into multiple categories using a clustering approach and calculated the reconstruction loss of each characteristic group separately over the AE network. Bovenzi et al. [26] suggested a hierarchial NIDS by first introducing each traffic feature independently into the AE network and then weighing the total of premodality losses.

2.6 Federated Learning in Intrusion Detection

IDSs, which use traffic monitoring and analysis to identify potential security risks and assaults, have long been recognized as essential parts of defending ICT systems. IDS techniques are often classified as signature and anomaly based systems, while there are

numerous other classifications as well [27]. The latter takes advantage of certain features in network traffic, so any deviation from this pattern could be seen as a possible attack. The former cannot be used to identify a fresh attack because it depends on pre-existing network patterns. Recent initiatives have been put out in the IoT context by taking particular IoT devices and technology into consideration [28]. The majority of recommended IDS strategies for IoT are based on centralized strategies, in which devices transmit local data to cloud-based data centers or servers with substantial processing power so that it can be subjected to ML/DL analysis. Recent ideas have been put forth by taking particular IoT technology and devices into account. To begin, end users may be concerned about their privacy if local IoT device data becomes public, as this may allow hackers to deduce their patterns of behavior. Second, in dynamic IoT contexts, the time it takes to discover possible threats may be essential in avoiding their spread. Third, numerous IoT scenarios involve devices with limited resources interacting through wireless systems that have throughput and bandwidth limitations. Consequently, the continuous transfer of device network data may constitute a substantial overhead for IoT networks with a huge number of linked devices. FL is a perfect option for on-device training in resource-constrained devices while keeping privacy-preserving techniques using decentralized data since it addresses these challenges.

Rey et al. [29] suggested a FL framework for malware detection in IoT combining both supervised and unsupervised federated models (multilayer perceptron and autoencoder) which is tested for different numbers of hidden layer using the N-BaIoT dataset. While maintaining privacy, outcomes for FL approaches are nearly identical to those for centralized models. Friha et al. [30] introduced an IDS which is FL-based for agricultural IoT infrastructure that allows the devices to train local models for different sets of clients, including 5, 10, and 15. 50 communication rounds were used in all. The experiment's findings demonstrate that, after 50 rounds, the system's accuracy is very similar to that of the centralized model while still protecting the confidentiality of the client's information. Nguyen et al. [31] developed an anomaly detection system which is self learning. It uses a FL approach which can identify hacked IoT devices in the network. Suggested model is constructed with Long Short Term Memory (LSTM) and Gated Recurrent Units (GRUs). It discovers unusual deviations from the devices' predetermined communication patterns by continuously analyzing the network traffic of the devices without the need for human intervention. Liu et al. [32] suggested a deep anomaly detection architecture for distributed edge devices to perceive time series data using the FL approach for the Industrial IoT. Additionally, it uses LSTM modules for time-series predictions and CNN-LSTM model to extract fine-grained features from past observations utilizing time series data, hence avoiding gradient-dispersion issues and memory loss. Khoa et al. [33] recommended using deep belief networks to be installed in IoT gateways in order to detect possible attacks on a particular IoT subnet. Next, various models are aggregated using FL. However, the authors

withhold information on the implementation that is being utilized as well as specifics of the evaluation that account for variables like distribution of the data, number of clients, or number of training cycles. Preuveneers et al.'s [34] used FL for intrusion detection which resulted in training and testing accuracy of about 97% on the CICIDS2017 data set, indicating increased effectiveness and confidentiality of training. Zhao et al. [35] proposed an intrusion detection technique with greater classification accuracy than conventional approaches, which is based on the FL-LSTM framework.

The development of FL aided IDS techniques for IoT is still in its early stages, according to our literature study. Many of the earlier works don't discuss the implementation that is being utilized or specifics regarding the evaluation method, including the number of clients or training sessions. Additionally, the majority of the works either ignore the model deployed among the clients or they take into account situations where the data of the clients are linked to a subset of the dataset that has the similar number of samples for every attack under consideration.

2.7 DP Aided FL for Intrusion Detection

As previously noted, despite the privacy benefits afforded by FL, recent study indicated that multiple attacks are still conceivable during the federated training process, for which privacy protecting measures are in use. Siachos et al. [36] compared different ML algorithms when DP mechanism is added to each of them. However, they didn't disclose details about the privacy parameters used and added noise. Alazab et al. [37] compared the effectiveness of FL in IDS with other traditional DL algorithms. But they did it based on older datasets. Pedro et al. [7] provides an analysis of the performance of FL with different DP perturbation techniques under different privacy parameters, though the research didn't focus much on the model deployed at both the client and server end. Lu et al. [38] utilized blockchain based on the application of DP such that the federated training process may utilize the same computing power needed for the consensus mechanism. They did not, however, disclose assessment information pertaining to the DP technique that was being used. In addition, Zhao et al. [39] evaluated how using Gaussian noise based DP approach in a FL context affected accuracy. A fog architecture for FL-based DDoS attack detection and mitigation was presented by Li et al. [40]. But privacy approaches are not integrated; only DoS attacks are taken into account. Khoa et al. [33] also took into consideration the use of FL when implementing an intrusion detection strategy in Industry 4.0 situations. They used various ML algorithms and datasets to compare their approach, but they neglected to take privacy concerns into account. Moreover, Attota et al. [41] developed an IDS strategy for IoT using an ensemble method and an optimization technique for feature extraction. They used a dataset of traffic from a single IoT protocol for validation. Furthermore, Man et al. [42] recently employed a federated CNN for intrusion detection in IoT with the goal of lowering communication overhead. The majority of the

other studies are predicated on old industrial data and do not take into account current IIoT attack scenarios. Also, Privacy-preserving measures are not integrated.

2.8 Conclusion

In this chapter, related research works on intrusion detection systems are described briefly. Some basic concepts are also described in detail. The proposed workflow on DP Aided FL Intrusion Detection System is described in the next chapter.

Chapter 3

Proposed Methodology/Framework/ Mechanism/ System Architecture

3.1 Introduction

Though federated learning architecture is quite suitable for privacy preservation, there are many scopes for innovation and development in this research area. However, A Differential Privacy based Federating Learning model is proposed where a customized BiLSTM-CNN model is implemented for intrusion detection. Addressing the challenges of privacy-accuracy tradeoff our proposed model overcomes the existing limitations. The proposed architecture can be visualized from Fig. 3.1.

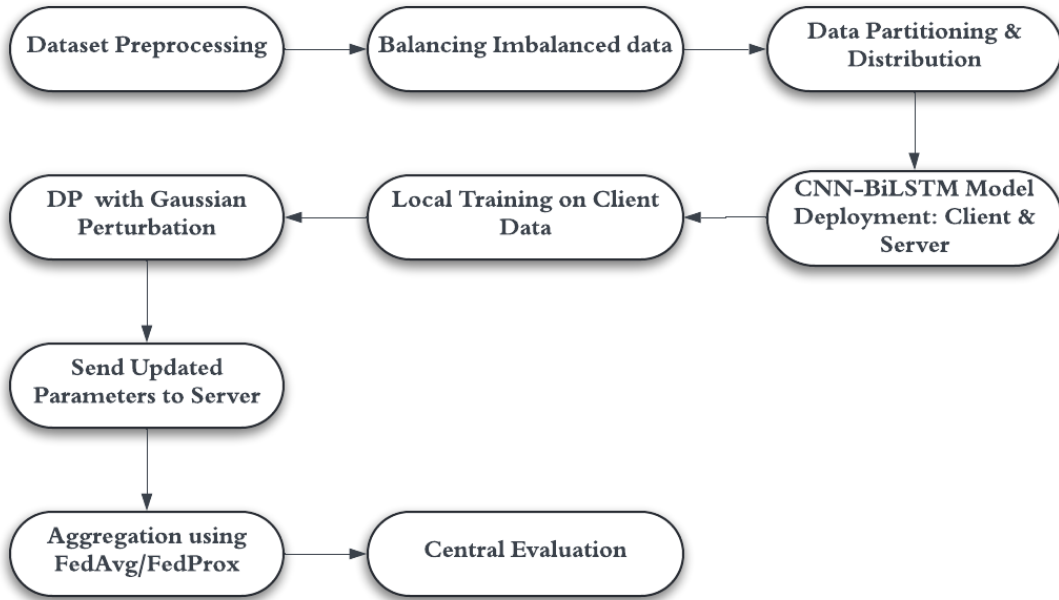


Figure 3.1. Proposed System Architecture

3.2 Datasets

We have used two publicly available datasets, namely BoT-IoT dataset [8] and N-BaIoT dataset [9] in this research work.

3.2.1 BoT-IoT Dataset

BoT-IoT dataset, curated by University of New South Wales (UNSW) which was produced by simulating a realistic smart home network, consists of data on both actual and simulated attacks.

Statistics of the dataset is described in Table 3.1. The Bot-IoT is made up of a number

Table 3.1. Statistics of BoT-IoT dataset

Category	Sub-category	Number of instances
Information gathering	Service Scanning	1463364
	OS Fingerprinting	358275
Denial of Service	DDoS-TCP	19547603
	DDoS-UDP	18965106
	DDoS-HTTP	19771
	DoS-TCP	12315997
	DoS-UDP	20659491
	DoS-HTTP	29706
Information theft	Keylogging	1469
	Data theft	118
Total		73360900

of sets and subsets with different file types, sizes, and feature counts. These sets include the raw set, the entire set, the 5% sub-set and 10-Best subset. The raw collection was processed using the Argus program, which examines network data from a network interface card (NIC) or PCAP file, and contains over 70 GB of packet capture files (PCAP). For Bot-IoT, PCAP files were processed using Argus, and the generated data was then entered into a MySQL database by an Argus client software. The entire set of data from the MySQL database was exported as comma-separated values (CSV). SQL queries and stored procedures in the MySQL database were used to introduce further features into the 5% subset and the 10 best subset of the Bot-IoT. The Entire Set, the first processed data collection, comprises CSV files with over 72 million instances, each of which represents a network flow. Argus network flow metadata was used to extract 26 independent features; however, it does not contain the 14 "additional" or computed features that the authors in [6] introduced. Additionally, Bot-IoT includes two frequently used subsets. Approximate 3.6 million instances, 43 independent characteristics, and 3 dependent features make up the 5% Subset, which is recommended as a more manageable variation. The 10-Best Subset has the same instances as the original 5% subset, while only having 10 of the original 43 independent qualities in the final subset. Of its sixteen independent features, ten are considered to be the best by Koroniotis et al. [6], while six are flow identifiers. It contains three dependent features. The mapping of the joint entropy along with the correlation coefficient of the 5% Subset of the 43 independent characteristics yielded the 10-Best features. The features that performed the best in both tests were selected as the 10-Best

features.

3.2.2 N-BaIoT Dataset

The absence of publicly available botnet datasets, particularly for the IoT, is addressed with N-BaIoT dataset. It presents actual traffic information that was collected from 9 commercial IoT devices that were genuinely infected with BASHLITE and Mirai. A deep autoencoder is trained and optimized using two thirds of the benign data from each of the nine IoT devices. The purpose of this was to record typical network traffic patterns. Nevertheless, the dataset can also be utilized for multi-class classification since the malicious data can be split up into 10 attacks that are each carried out by 2 botnets. Details of the N-BaIoT dataset is provided in Table 3.2.

Table 3.2. Statistics of N-BaIoT Dataset

Category	Number of instances
Norm	555932
g_combo	515156
g_scan	255111
g_junk	261789
g_udp	946366
m_ack	643821
m_scan	537979
m_syn	733299
m_udp	1229999
m_udpp	523304
Total	6202756

3.3 Data Preprocessing

We have employed encoding, standardization and sampling techniques for preprocessing the dataset. One-hot encoding is the method used to encode the categorical values in the dataset. Given that the traffic data comprises elements of several dimensions, the degree of each dimension in the data set varies considerably. Therefore, to ensure data reliability Standardization is used. Assuming that the dataset has a Gaussian distribution and that each variable is measured at a distinct scale, standardization ensures that each variable contributes equally to the analysis. If a dataset is utilized that is not evenly distributed, the classification algorithm favors the majority class. Unbalanced classes in the dataset were handled using the Random Undersampling and SMOTE (Synthetic Minority Over-sampling Technique) techniques.

3.3.1 One-Hot Encoding

One-hot encoding is a method for converting categorical variables into binary vectors. Each category is assigned a vector of all zeros, with the exception of a single member set to 1. The position of the 1 corresponds to the category's index in a predetermined list of all probable categories. Without assuming an ordinal link between categories, this encoding approach guarantees that each category value is represented separately. In machine learning and statistical modeling tasks, where categorical data needs to be converted into a format appropriate for algorithms that need numerical inputs, one-hot encoding is frequently used.

3.3.2 Standardization

Numerical data can be transformed into a standard scale using the data preprocessing technique known as standardization, sometimes referred to as z-score normalization. Each feature (or variable) in this process is rescaled to have a mean of 0 and a standard deviation of 1. With a standard deviation of 1, this transformation centers the data around zero, adjusting the data points in relation to the dataset's mean and distribution. Standardization is especially important when the dataset's attributes differ in scale or unit. It speeds up algorithm convergence during training, prevents larger scale features from dominating smaller scales, and ensures that all features are treated equally by the model. Standardization is applied independently to each feature, typically using the formula:

$$x_{\text{standardized}} = \frac{x - \mu}{\sigma}$$

where x is a data point, μ is the mean of the feature, and σ is the standard deviation of the feature.

3.4 Balancing Imbalanced Data

If a dataset is utilized that is not evenly distributed, the classification algorithm favors the majority class. We handled unbalanced classes in the dataset using the Random Undersampling and SMOTE (Synthetic Minority Over-sampling Technique) techniques.

3.4.1 Random Undersampling

Random undersampling is a technique used in imbalanced classification issues where one class (typically the minority class) has significantly less instances than the others. This method incorporates randomly selecting a subset of samples from the majority class (which has more instances) so that their size corresponds to the number of examples in the minority class. Random undersampling tries to balance the class distribution and reduce

bias towards the majority class during model training by lowering the amount of majority class samples used. While random undersampling is straightforward to execute, it can result in the loss of potentially significant information from the majority class, affecting the classifier's overall performance. As a result, it is frequently used in conjunction with other techniques, such as oversampling the minority class or using more advanced sampling procedures, to achieve greater class balance while keeping enough data for training.

3.4.2 Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE (Synthetic Minority Over-sampling Technique) is a method for addressing class imbalance in machine learning datasets. It works by creating synthetic samples for the minority class based on existing data. Starting with a minority class instance, SMOTE selects its k nearest neighbors in the feature space and generates synthetic instances along the line segments that join the instance to its neighbors. This procedure is repeated until the intended balance of minority and majority classes is reached. SMOTE effectively enhances the representation of the minority class, which improves the robustness and performance of classifiers trained on imbalanced data.

3.5 Data Distribution

The preprocessed dataset is distributed among the clients in Non-IID (Non Independent and Identically Distributed) manner. The training data for any client in FL is greatly influenced by the use of specific local devices, so there may be large changes in the data distribution among linked clients. The term "Non-IID" refers to this phenomenon [6]. A local distribution $P_k(x, y)$, where x is considered as the input feature and y is the label, is intended to be followed by each data sample (x, y) for a supervised learning job on client k . Non-IID indicates that P_k varies depending on the client. We used the Horizontal FL framework for our research. Although the conditional feature distribution $P_k(x|y)$ is distributed evenly among the clients, the label distributions $P_k(y)$ on the clients deviate in a common Non-IID category known as label distribution skew [43].

3.6 Selection of Deep Learning Model

We have explored a number of DL methods, including CNN, LSTM, BiLSTM, CNN-LSTM, and CNN-BiLSTM, for both FedAvg and FedProx and observed that CNN-BiLSTM achieved the highest accuracy. Table 3.3 shows the accuracy comparison among DL models for both FedAvg and FedProx, which eventually proves the better performance of CNN-BiLSTM.

Table 3.3. Comparison of accuracy among different DL models

Deep Learning Models	BoT-IoT		N-BaIoT	
	FedAvg	FedProx	FedAvg	FedProx
CNN	97.83	98.18	96.95	97.45
LSTM	96.99	97.20	96.50	96.77
Bi-LSTM	97.17	97.12	96.33	96.88
CNN-LSTM	96.91	97.52	95.93	96.77
CNN-BiLSTM	98.00	99.24	97.20	98.60

3.7 Proposed BiLSTM-CNN Architecture

Bidirectional Long Short-Term Memory networks (BiLSTM) are a form of recurrent neural network (RNN) developed to solve the problem of learning long-term dependencies in sequential input. BiLSTMs, as opposed to regular LSTMs, process data in both the forward and backward directions at the same time. BiLSTMs can capture contextual information from both past and future inputs, making them especially useful for natural language processing (NLP), speech recognition, and time series prediction. BiLSTMs can model complicated patterns and dependencies throughout full sequences by keeping separate hidden states for forward and backward sequences, which is critical for applications that need context awareness over time. This feature has resulted in their broad acceptance in a variety of disciplines where sequential data analysis is critical, giving strong performance in tasks that need a deep knowledge of temporal correlations.

Convolutional Neural Networks (CNNs) use a hierarchical structure to efficiently analyze and extract features from input data. CNN's core building elements are convolutional layers, pooling layers, and fully connected layers. Convolutional layers use learnable filters (kernels) to produce feature maps from input data, capturing spatial patterns. Pooling layers then lower the dimensionality of each feature map while retaining the most relevant information, resulting in improved translation invariance and computational efficiency. Finally, fully connected layers use the preceding layers' high-level characteristics to classify or regress input data into the appropriate output classes or values. CNNs are designed to automatically learn hierarchical data representations, beginning with low-level features at lower layers and progressing to more abstract and complex features at higher layers, making them especially useful for tasks that require spatial hierarchies of features, such as image recognition and, in some cases, sequential data analysis.

BiLSTM-CNN for intrusion detection is a cybersecurity-focused use of the hybrid deep learning architecture. In this case, the model combines Bidirectional Long Short-Term Memory networks (BiLSTM) with Convolutional Neural Networks (CNN) to improve the accuracy and effectiveness of identifying threatening activity in network traffic. BiLSTMs excel at capturing temporal relationships and event sequences, which are critical for understanding network activity over time. Meanwhile, CNNs excel at extracting spatial

patterns and local information from packet headers or network traffic. Combining these features allows BiLSTM-CNN models to effectively learn complicated patterns indicative of various forms of cyber threats, such as denial-of-service attacks or infiltration attempts. This technique combines the features of both architectures to provide comprehensive detection capabilities, allowing for real-time monitoring and proactive response to possible security threats in network environments.

The proposed model is composed of one input layer, one convolution layer with activation, regularization, normalization and dropout layers, one BiLSTM layer with the same layers, one dense layer, and one output layer. CNNs are considered as the best choice for prediction as they don't have to lose important data when learning input features. Fig. 3.2 depicts a layered view of the proposed model.

A CNN1D model is used to generate convolutional layers. Using a single-directional

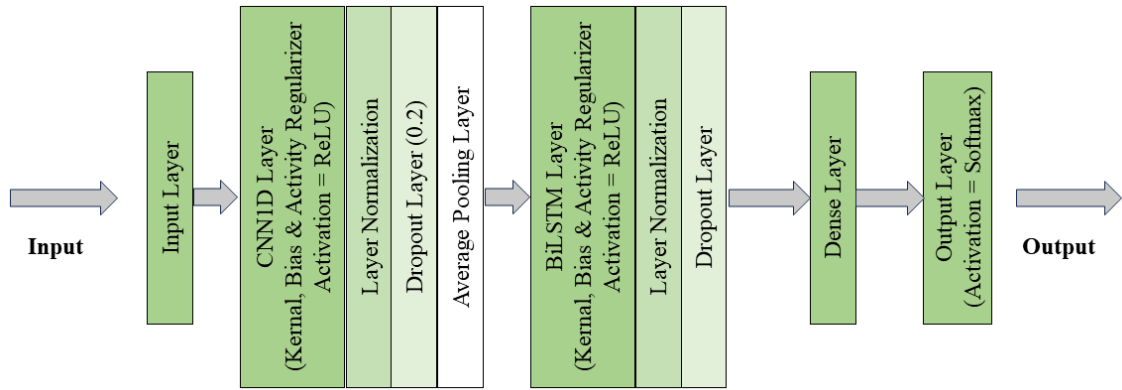


Figure 3.2. Proposed BiLSTM-CNN Architecture

kernel, temporal access is utilized for 1D convolution. It is usual practice to process time series data with CNN1D, which requires two-dimensional input data. The activation, regularization, normalization, and dropout layers are brought together with the convolutional layer. Activation layer utilizes ReLU activation. An axis of features is used to normalize data in the layer normalization process. To avoid overfitting, the dropout layer employs 11-12 functions to randomly remove neurons from the activity regularization layer. A pooling layer minimizes the computational load needed for data analysis as well as the size of a complicated feature. An average pooling layer is utilized here.

One major issue with DL models is overfitting. Kernel, bias, and activity regularizers were employed by the BiLSTM layer to lower the likelihood of overfitting. The activity regularizer penalizes the layer's output, its bias is penalized by the bias regularizer, and its kernel is penalized by the kernel regularizer. The values for kernel, bias, and activity regularization are calculated by the regularizers using the values of 11-12. The next step is layer normalization, which often lowers error rates to speed up and stabilize learning. A RNN's hidden state dynamics may stabilize with layer normalization. Layer normalization avoids the formation of extra dependencies between training sessions by computing nor-

malizing statistics from the hidden layer's input neurons, in contrast to other ways, such as batch normalization. The use of dropout layers and regularization reduced the chance of overfitting. The 11-12 factors explain the functionality of the regularization layer. With a frequency that corresponds to the dropout rate, the dropout layer sets a fraction of input units to 0 periodically and enhances overfitting prevention during training. For preventing overfitting, the training dataset for a RNN must be substantially altered.

Overfitting is reduced with a dropout layer because it ignores some neurons during the training phase. As a neural network gains knowledge, the weights of its neurons adapt to its surroundings. This suggests that changes in weight are not relayed back to the neuron and their effect on downstream neuron activity is momentarily mitigated during the forward transit. The final dropout layer provides input to the dense layer, which then transmits its output to the output layer. ReLU activation is used in the dense layer along with 64 neurons. There are exactly as many neurons employed in the output layer as there are classes in the dataset. The detailed configuration of the model can be obtained from Table 3.4.

Table 3.4. Configuration of Proposed BiLSTM-CNN Model

Name of Layer	No. of Layers	Configuration
Input Layer	1	Input Features
CNN1D Layer	1	Filter = 64, Kernel Size = 5, Padding = same, Bias, Kernel and Activity regularizer, Activation = 'ReLU'
BiLSTM Layer	1	Number of units = 64, Bias, Kernel and Activity regularizer, Activation = 'ReLU'
AveragePooling1D	1	
Layer Normalization	2	Axis = 1, Center = 'True', Scale = 'True'
Regularization Layer	2	L1=0.0001, L2=0.0001
Dropout Layer	2	Dropout Rate = 0.2
Dense Layer	1	Number of units = 64, Activation = 'ReLU'
Output Layer	1	No. of neurons = No. of output classes, Activation = 'Softmax'
No. of Trainable Parameters	77,250 (BoT-IoT) and 75,889 (N-BaIoT)	
Floating Point Operations (FLOPS)	3.06M (BoT-IoT) and 3.35M (N-BaIoT)	

A CNN can effectively capture the temporal and spatial correlations related to an anomaly detection task when the optimal filters are used. In addition, the architecture fits the anomaly detection data better because it has reusable weights and fewer parameters. The dimension of the features is decreased by the pooling layer, lowering computational power.

3.8 Integration of Differential Privacy

We have used FedAvg and FedProx Algorithms to aggregate the model parameters in the server's end. Differential Privacy is integrated at the client's end to anonymize the parameter updates through adding random noise.

Algorithm 1: DP-enabled FL Architecture using FedAvg

Input: Communication Round Number R , Client Number C , Epochs E , Learning Rate η

Output: Updated Global Model W_G

```

1 Client Side:
2 LocalUpdate( $k, W_G^i$ ):
3   Let  $x \leftarrow$  the inputs and  $y \leftarrow$  local data label
4   Local input normalization and initialize weights
5 for each epoch  $1.....E$  do
6   |   compute loss, gradient
7   |   each device  $k \in S_t$  updates  $w^i$  for  $E$  epochs of SGD with step size  $\eta$  to obtain  $w_i^k$ 
8 end
9 Utilize the  $(\epsilon, \delta)$ -DP technique on the weights to obtain  $\kappa(w)$ 
10 Send  $\kappa(w)$  back to the server
11 Server Side:
12 Initialization of  $W_G^0$ 
13 for each round  $i \leftarrow 0, 1....$  do
14   |    $S_t \leftarrow$  pick a set of clients out of  $C$  for each client  $k \in S_t$  do
15   |   |    $w_i^k \leftarrow$  LocalUpdate( $k, W_G^i$ )
16   |   end
17 end
18 Calculate new weight
19  $w_i^G \leftarrow \sum \frac{D_i}{D} w_i^k$ 

```

The overall working procedure is described in Algorithm 1 and 2.

Algorithm 2: DP-enabled FL Architecture using FedProx

Input: Communication Round Number R , Client Number C , Epochs E , Learning Rate η , γ -inexactness, Proximal Term μ

Output: Updated Global Model W_G

```

1 Client Side:
2 LocalUpdate ( $k, W_G^i$ ):
3 Let  $x \leftarrow$  the inputs and  $y \leftarrow$  local data label
4 Local input normalization and initialize weights
5 for each epoch  $1 \dots E$  do
6   compute loss, gradient
7   each device  $k \in S_t$  updates  $w^i$  for  $E$  epochs of SGD with step size  $\eta$  to obtain  $w_i^k$ 
   which is a  $\gamma_k^i$ -inexact minimizer of:
8    $w_i^k \approx \arg \min_w h_k(w; w^i) = \kappa(w) + \frac{\mu}{2} \|w - w^i\|^2$ 
9 end
10 Utilize the  $(\epsilon, \delta)$ -DP technique on the weights to obtain  $\kappa(w)$ 
11 Send  $\kappa(w)$  back to the server
12 Server Side:
13 Initialization of  $W_G^0$ 
14 for each round  $i \leftarrow 0, 1, \dots$  do
15    $S_t \leftarrow$  pick a set of clients out of  $C$  for each client  $k \in S_t$  do
16      $w_i^k \leftarrow$  LocalUpdate( $k, W_G^i$ )
17   end
18 end
19 Calculate new weight
20  $w_i^G \leftarrow \sum \frac{D_i}{D} w_i^k$ 

```

3.8.1 Evaluation

Centralized Evaluation is considered in this proposed architecture. All the updated parameters are sent back to the server, the global model is updated based on the aggregation of locally updated parameters. Performance of the model is assessed based on the validation set at the server's end.

3.9 Conclusion

The workflow for the proposed architecture is described in this chapter. The process of the implementation of this model is illustrated in the next chapter.

Implementation

4.1 Introduction

Our proposed model is assessed based on two different datasets. This chapter describes the data distribution process and implementation strategies of the proposed architecture with detailed configuration.

4.2 Implementation of BiLSTM-CNN in FL Architecture

The proposed BiLSTM-CNN model is placed both at the server's and clients' end. The hyperparameters used in the mentioned model are described in Table 4.1. The model was

Table 4.1. Hyperparameters Used in BiLSTM-CNN Model

Hyperparameters Used	Configuration Details
Optimizers	DPGradientDescentGaussian (For Client) SGD (For Server)
Loss Function	Sparse Categorical Cross-entropy
Learning Rate	0.001
Batch Size	64
Early Stopping	Monitor = 'loss', Patience = 5

trained for 50 communication rounds and a 64-batch size at the client's side using the DPSGD (Differentially Private Stochastic Gradient Descent) optimizer. DPSGD functions similarly to mini-batch gradient descent. Additionally, the mini-batches are distributed using a random data sample. Server end used SGD optimizer. The most crucial element in a neural network is undoubtedly its loss functions. Sparse categorical cross entropy loss function was employed here. Model overfitting was decreased using four different methods. At the BiLSTM and CNN layers, regularizers were applied first. L1-L2 approaches are used for regularization in kernel, bias, and activity regularizers. Then the dropout layer was applied. Finally, if the training loss did not reduce throughout the training period, an early stopping technique was utilized. Early stopping reduces the possibility of overfitting, when training of a model happens over a notable number of epochs. These methods completely rule out overfitting of the model.

4.3 Data Distribution Strategy

Since the dataset will be dispersed among the clients at random in Non-IID manner, each client will receive a different class distribution. The use of certain local devices has a substantial impact on the training data for each client in FL, the distribution of data among linked clients may change significantly. We used the Horizontal FL framework for our research. Horizontal FL, also known as homogeneous FL, characterizes situations when the training data of participating customers have the same space of features but a distinct sample space. The most common reason for non-IID in horizontal FL is skew in the label distribution. Although the conditional feature distribution $P_k(x|y)$ is distributed evenly among the clients, the label distributions $P_k(y)$ on the clients deviate in a common Non-IID category known as label distribution skew [43]. Example of data distribution of BoT-IoT dataset can be visualized from Fig. 4.1.

CLIENT 1	seq	stddev	N_IN_Conn_P_SrcIP	min		Label
	97288	1.461296	34	0.00		1
	26557	0.315755	100	3.48		0
	31324	1.694439	76	0.00		2
CLIENT 2	seq	stddev	N_IN_Conn_P_SrcIP	min		Label
	62921	0.268666	100	2.97		2
	99168	1.823185	63	0.00		3
	105063	0.822418	100	2.98		4

Figure 4.1. Visualization of label distribution skew in the BoT-IoT dataset

4.4 Implementation of Differential Privacy in FL Architecture

The Differential Privacy-enabled FL architecture is shown in Fig. 4.2.

Clients serve as representations of the final device where local training is conducted. To train the global model provided by the aggregator and produce a model update, each client is in charge of using its own local data. The reasoning necessary to employ the proper DP algorithm is also provided to clients. Furthermore, the aggregator serves as the primary hub for compiling model adjustments made by clients and transforming them into an aggregated model that is distributed to customers at the beginning of every training cycle. Table 4.2 lists the parameters that were used to implement this DP focused FL architecture.

A certain client selection approach is used by the aggregator to determine which clients will be part of the training process. During the first training phase, the aggregator builds a global model W_G and distributes its weights (W_i^G) among the selected clients. Each client creates a unique model update with locally selected weights by utilizing its own data to

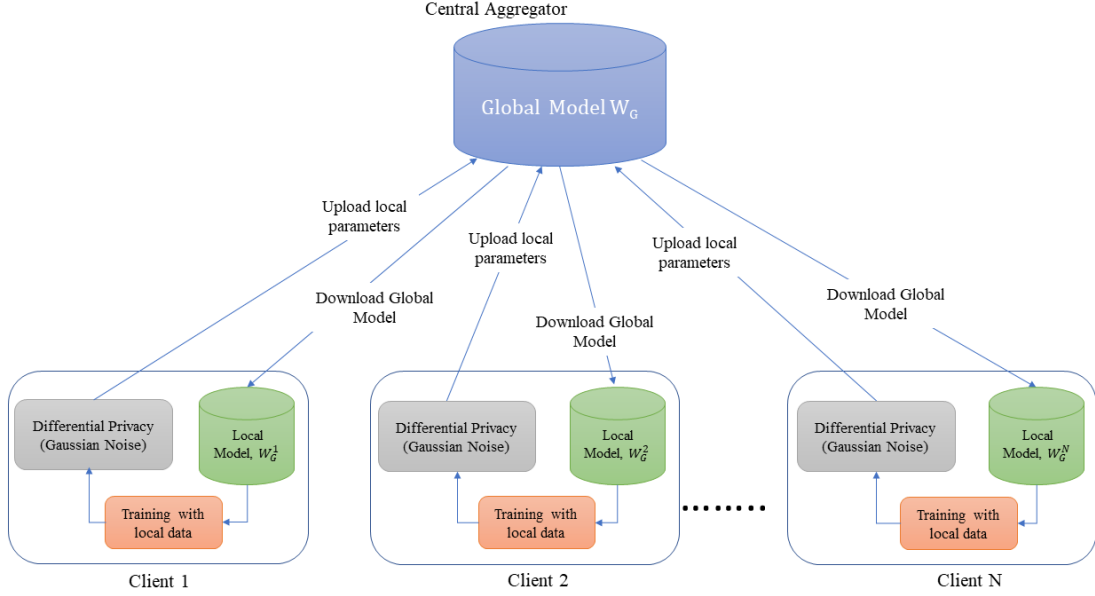


Figure 4.2. Proposed Differential Privacy aided FL-Architecture

Table 4.2. Parameters for DP-FL Architecture

Parameters	Description
No. of Clients	5 to 25
No. of communication rounds	10 to 50
Noise Applied	Gaussian Noise
Noise Multiplier	0 to 1
Fraction Fit	0.5
Privacy Parameters	Epsilon (ϵ) and Delta (δ)

train the shared model. Note that the number of epochs E determines how many epochs a client executes. The Gaussian DP perturbation process is used by the clients to make the calculated weights anonymous. The aggregator then gets the anonymized weights that are produced. The aggregator compiles all model changes received by using FedAvg and FedProx.

FedAvg takes the average of the weights that FL clients supply to build the global model. Let $W_k = (w_i^k)$ represent the weights of the client k and $W_G = (w_i^G)$ represent the weights of the general model, respectively. Then, $w_i^G = \sum \frac{D_i}{D} w_i^k$, where the size of total data and the each client's data are represented by D and D_i , respectively. Until reaching a specific round R or obtaining a specific goal accuracy, these stages are repeated.

4.5 Conclusion

The implementation procedure is discussed in this chapter. The following chapter describes the experimental result analysis of our proposed model in detail.

Chapter 5

Experimental Result analysis and Evaluation

5.1 Introduction

This chapter analyzes the experimental result of our proposed BiLSTM-CNN model in DP-Aided FL Architecture. We have executed the training phase of this model for 50 communication rounds. In this chapter, firstly we have outlined the system configuration, evaluation parameters. Afterward, we have shown some experimental results based on accuracy, loss, number of communication rounds, number of clients, varying values of noise multipliers, two privacy parameters, namely, epsilon (ϵ) and delta (δ). Finally, a comparison of our proposed model with other existing works is discussed in this chapter.

5.2 Experimental Setup

All the experiments in this research are conducted using intel 11th generation core i5-11400H processor with a 8 GB RAM. The simulations are performed using a combination of the Flower framework with the TensorFlow backend. Differential Privacy is integrated using Tensorflow Privacy. 80% of the dataset used is considered as the training set and rest is utilized for validation. The training set is distributed among different numbers of clients varying from 5 to 25. The deployed model is evaluated at the server's end using the validation set. Secure gRPC channel is utilized during client-server communication. The client-server network has been setup in CLI. We created two python files named server.py and client.py. The server is run first and the clients are connected with the server using gRPC channel. The differential privacy implementation is done in clients' end. Server aggregated the updated parameters from the clients and completed the centralized evaluation. The complete experimental setup can be visualized from Figure 5.1.

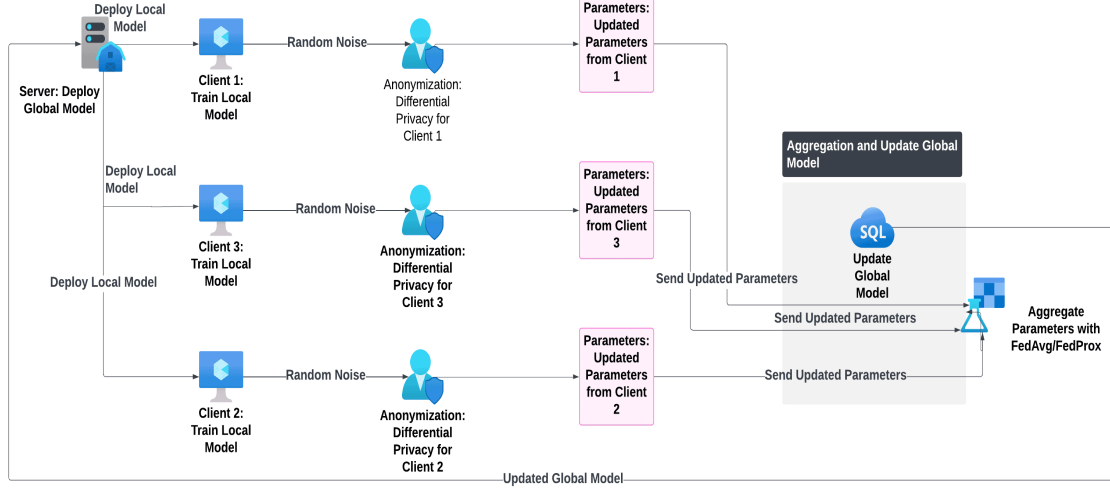


Figure 5.1. Schematic View of Implementation Setup

5.3 Performance Metrics

5.3.1 Accuracy

Calculation of accuracy for centralized evaluation can be obtained from the following equation:

$$\text{Accuracy}(\theta) = \frac{1}{N} \sum_{i=1}^C \left(\frac{1}{n_i} \sum_{j=1}^{n_i} 1(\hat{y}_i^{(j)} = y_i^{(j)}) \right) \times 100 \quad (5.1)$$

Here, C = Total number of clients.

n_i = Number of samples held by client i .

N = Total number of samples across all clients.

$\hat{y}_i^{(j)}$ = Predicted label for the j – th sample of the client.

$y_i^{(j)}$ = True label for the j – th sample of the client.

$1(\cdot)$ is the indicator function that returns 1 if the indicator function is true (predicted label equals to true label), and 0 otherwise.

5.3.2 Loss

Aggregation of Local losses: Each client (or device) computes its own local loss using its local data and the current global model parameters. Let L_i denote the local loss function for client i , where θ represents the global model parameters. Average Loss can be obtained from the following equation.

$$L(\theta) = \sum_{i=1}^C \frac{n_i}{N} L_i(\theta) \quad (5.2)$$

Here, C = Total number of clients. n_i = Number of samples held by client i . N = Total number of samples across all clients.

5.3.3 Privacy Parameters

5.3.3.1 Epsilon (ϵ)

The amount of information about an individual that can be deduced from the published data is referred to as privacy loss. Epsilon(ϵ) is a differential privacy parameter which determines the level of privacy provided by a differentially private mechanism. A lower epsilon indicates that more privacy is assured, whereas a higher epsilon indicates that less privacy is guaranteed [14].

5.3.3.2 Delta (δ)

Delta(δ) is another differential privacy parameter that limits the likelihood of an extreme privacy breach. A smaller delta indicates a lesser likelihood of a privacy breach, whereas a bigger delta indicates a higher likelihood of a privacy breach [14]. In other words, delta assesses the likelihood that an attacker can learn something about a person even if they have access to both the differentially private output and the entire dataset. A smaller delta indicates that something is less likely to occur, whereas a greater delta indicates that it is more likely to occur.

5.4 Result Analysis for BoT-IoT dataset

5.4.1 Variation of Accuracy with respect to Communication Rounds

Fig. 5.2 and 5.3 demonstrate the variation of Accuracy for the increasing number of communication rounds for BoT-IoT dataset using FedAvg and FedProx, respectively. For FedAvg with the increasing number of communication rounds, the accuracy of the model increases. As federated evaluation is considered in this research work, after each communication round the server end's model is evaluated based on the validation dataset. Accuracy has increased in a constant manner with the communication rounds for BoT-IoT dataset using FedAvg.

With the increasing number of communication rounds the accuracy of the model increases for each group of the clients using FedProx. It is observed that the range of accuracy for FedProx is much better than the accuracy using FedAvg. In Fig. 5.4 and Fig. 5.5 We can see how the accuracy changes with varying numbers of clients for both the aggregation methods respectively. In case of FedAvg, it is observed that as the number of clients is increasing the accuracy of the model decreases slightly. The reason behind this can be explained as client number increases, it brings more updated parameters to the server, making it a bit difficult to perform simultaneous model parameter aggregation. As our proposed model is quite lightweight with a smaller number of parameters, the reduction of accuracy is not that much higher with the increasing number of clients.

In case of FedProx, the graph is more stable as FedProx allows for variable amount of

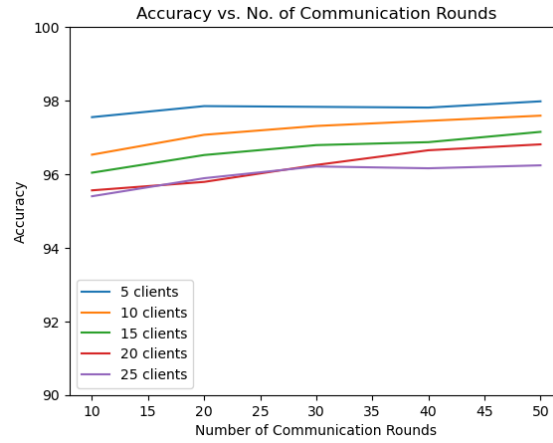


Figure 5.2. Change of Accuracy vs. No. of Communication Rounds for FedAvg

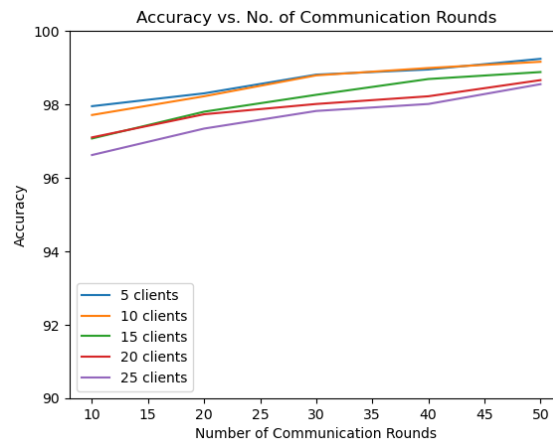


Figure 5.3. Change of Accuracy vs. No. of Communication Rounds for FedProx

work to be performed locally across devices based on their available system resources and then aggregate the partial solutions sent from the clients. It helps to alleviate negative impacts of the system heterogeneity. Proximal μ is used to limit the impact of variable local updates.



Figure 5.4. Change of Accuracy with respect to Number of Clients for FedAvg



Figure 5.5. Change of Accuracy with respect to Number of Clients for FedProx

5.4.2 Variation of Loss with respect to Communication Rounds

Fig. 5.6 and Fig 5.7 describe the variation of Loss for the increasing number of communication rounds for BoT-IoT dataset using FedAvg and FedProx respectively. With the increasing number of communication rounds, loss of the model decreases. Initially loss is higher for a larger number of clients for BoT-IoT, but, after 50 communication rounds the loss is minimized to a desired level for various client numbers. In case of FedProx, loss decreases with the increasing communication rounds in a constant manner.

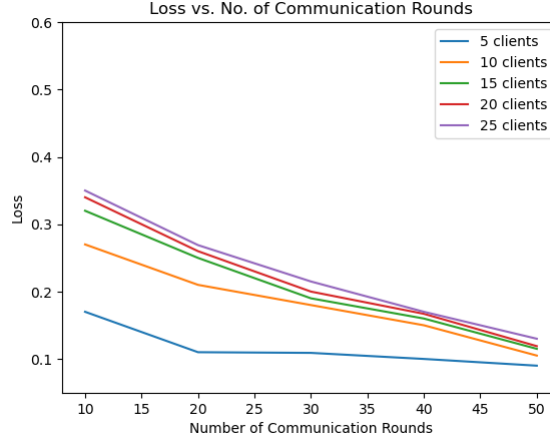


Figure 5.6. Change of Loss with respect to No. of Communication Rounds for FedAvg

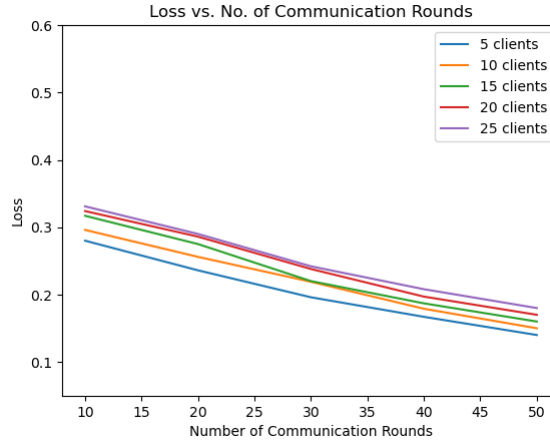


Figure 5.7. Change of Loss with respect to No. of Communication Rounds for FedProx

5.4.3 Variation of Accuracy with respect to Communication Rounds for varying noise multipliers

One of the key contributions of this research work is tailoring differential privacy at the client's end to safeguard the data privacy of the clients. Differential Privacy seeks to achieve a compromise between the need to share helpful information from datasets and the need to safeguard individuals' sensitive information inside those datasets. The basic principle behind differential privacy is to add a predetermined amount of noise or randomness to data before releasing or analyzing it, making it more difficult to discern specific particulars about any individual's data. Gaussian noise is added to the client's data to implement differential privacy. We added noise on different scales. It was done to observe the model's accuracy with different numbers of communication rounds for a varying number of clients.

Fig. 5.8, 5.9 and 5.10 provide the detailed illustration with different noise multipliers ranging from 0 to 1 for BoT-IoT dataset using FedAvg for 5, 15 and 25 clients respectively.

When the number of clients is 5, the accuracy curve for noise multiplier 0.75 is better

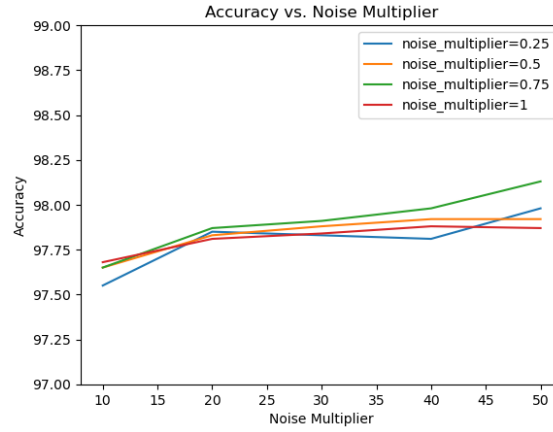


Figure 5.8. Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=5$

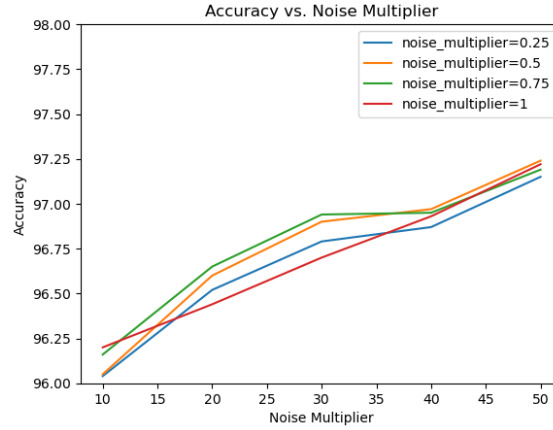


Figure 5.9. Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=15$

than the others. For noise multiplier 0.5 the accuracy is also good, but it becomes stable after 30 communication rounds. For a noise multiplier of 0.25 the accuracy curve seems to be unstable. Though the curves are stable for all the noise multipliers in the case of 15 clients, a noise multiplier of 0.75 maintains the highest accuracy. In the case of 25 clients, a noise multiplier of 0.5 acts in a stable way with a good accuracy. Noise multiplier of 0.75 has an unstable accuracy curve for 25 clients. The overall analysis summarizes that a noise multiplier of 1 sacrifices accuracy as it makes the data more noisy, noise multiplier of 0.75 provides a good accuracy curve for a small number of clients but as the number of clients increases, the curve becomes unstable.

In Fig. 5.11, 5.12 and 5.13 we can observe detailed illustration with different noise multipliers ranging from 0 to 1 for BoT-IoT dataset using FedProx for 5, 15 and 25 clients

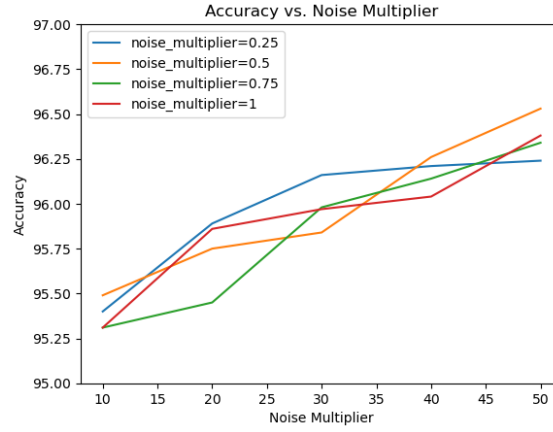


Figure 5.10. Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=25$

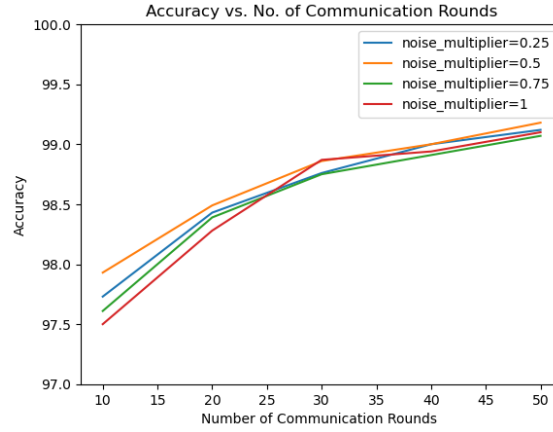


Figure 5.11. Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=5$

respectively. In case of FedProx, When the client number is 5, we get the highest and most stable accuracy changes for noise multiplier 0.5. For noise multipliers 0.25 and 0.75, the result was progressing well but started to deteriorate when the communication rounds increased further. For noise multiplier 1, the accuracy increased for 30 communication rounds, then decreased with the increasing communication rounds, which makes the graph unstable. Assessing 15 clients, we observe the curves are quite stable for all the noise multipliers and we get the highest accuracies for the noise multiplier of 0.5. For 25 clients, noise multiplier 0.1 gave us the lowest accuracies with the increasing communication rounds. We got stable curves for 0.5 and 0.75, whereas highest accuracies are obtained for noise multiplier of 0.5.

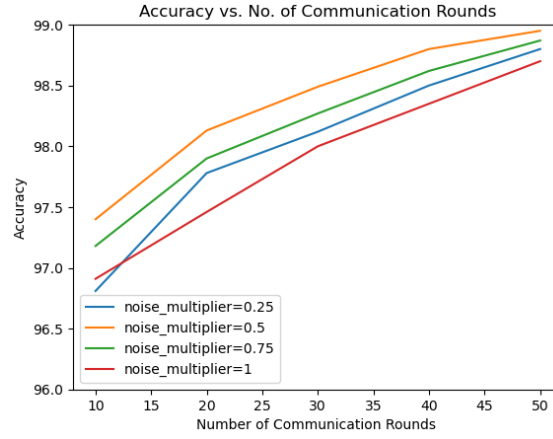


Figure 5.12. Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=15$

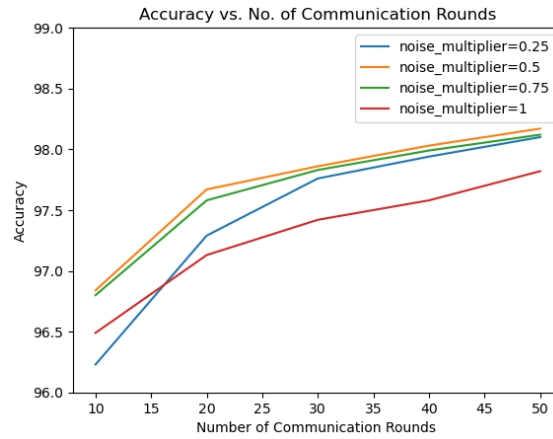


Figure 5.13. Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=25$

Table 5.5 and 5.2 describe the complete statistics of change of accuracy for different values of noise multiplier for varying number of clients using FedAvg and FedProx on BoT-IoT dataset respectively.

Table 5.1. Change of Accuracy for different noise multipliers for varying number of clients using FedAvg on BoT-IoT dataset

Number of Clients	Noise Multiplier	Accuracy for Communication Rounds (FedAvg) (%)				
		10	20	30	40	50
5	0.25	97.55	97.85	97.83	97.81	97.98
	0.5	97.65	97.83	97.88	97.92	97.92
	0.75	97.65	97.87	97.91	97.98	98.13
	1.00	97.68	97.81	97.84	97.88	97.87
15	0.25	96.04	96.52	96.79	96.87	97.15
	0.5	96.05	96.60	96.90	96.97	97.24
	0.75	96.16	96.65	96.94	96.95	97.19
	1.00	96.20	96.44	96.70	96.93	97.22
25	0.25	95.40	95.89	96.16	96.21	96.24
	0.5	95.49	95.75	95.84	96.26	96.53
	0.75	95.31	95.45	95.98	96.14	96.34
	1.00	95.31	95.86	95.97	96.04	96.38

Table 5.2. Change of Accuracy for different noise multipliers for varying number of clients using FedProx on BoT-IoT dataset

Number of Clients	Noise Multiplier	Accuracy for Communication Rounds (FedProx) (%)				
		10	20	30	40	50
5	0.25	97.73	98.43	98.76	99.00	99.12
	0.5	97.93	98.49	98.86	99.00	99.18
	0.75	97.61	98.39	98.75	98.91	99.07
	1.00	97.50	98.28	98.87	98.94	99.1
15	0.25	96.81	97.78	98.12	98.50	98.80
	0.5	97.40	98.13	98.49	98.80	98.95
	0.75	97.18	97.90	98.27	98.62	98.87
	1.00	96.91	97.46	98.00	98.35	98.70
25	0.25	96.23	97.29	97.76	97.94	98.10
	0.5	96.84	97.67	97.86	98.03	98.17
	0.75	96.80	97.58	97.83	97.99	98.12
	1.00	96.49	97.13	97.42	97.58	97.82

5.4.4 Variation of Accuracy with respect to Epsilon for varying delta

In Fig. 5.14 and 5.15 the change of epsilon and accuracy for different values of delta is observed using BoT-IoT dataset for both the aggregation methods respectively.

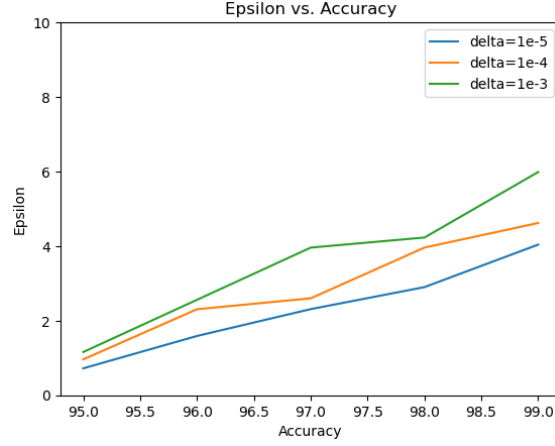


Figure 5.14. Change of Epsilon with respect to Accuracy for different values of delta for FedAvg

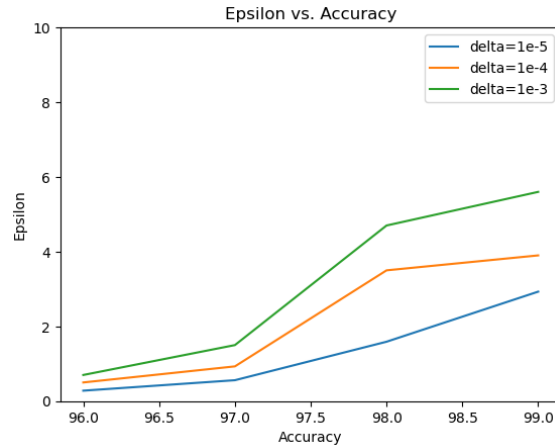


Figure 5.15. Change of Epsilon with respect to Accuracy for different values of delta for FedProx

For a smaller value of delta, epsilon is smaller for accuracy, which indicates lower privacy loss. As the value of delta increases, epsilon also increases indicating a higher privacy loss.

5.5 Result Analysis for N-BaIoT dataset

5.5.1 Variation of Accuracy with respect to Communication Rounds

Variation of Accuracy with varying number of communication rounds using N-BaIoT dataset for FedAvg and FedProx can be visualized from Fig 5.16 and Fig 5.17 respectively. It is observed that the accuracy of the model increases gradually with the increasing number of communication rounds.

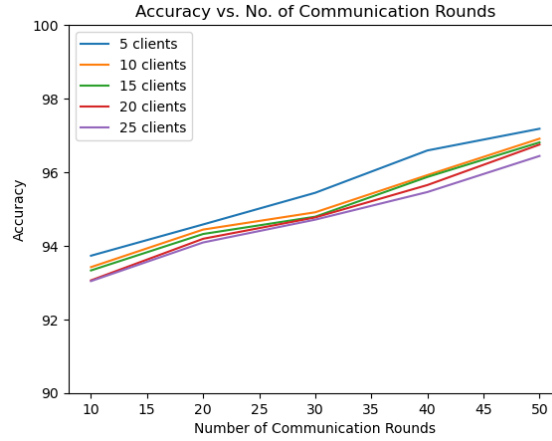


Figure 5.16. Change of Accuracy vs. No. of Communication Rounds for FedAvg

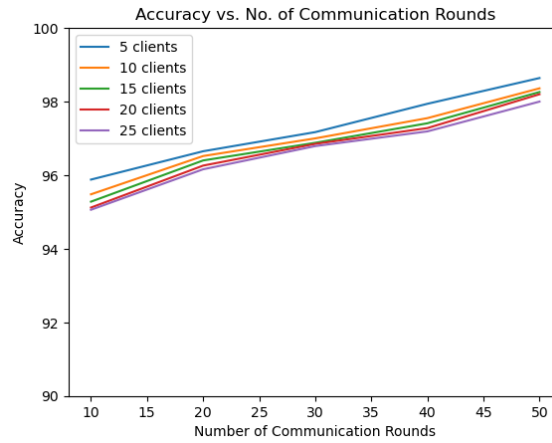


Figure 5.17. Change of Accuracy vs. No. of Communication Rounds for FedProx

Fig 5.18 and 5.19 depict the change of accuracy with varying number of clients for both the aggregation methods. It is observed that the performance is quite similar with the graphs provided for BoT-IoT dataset.

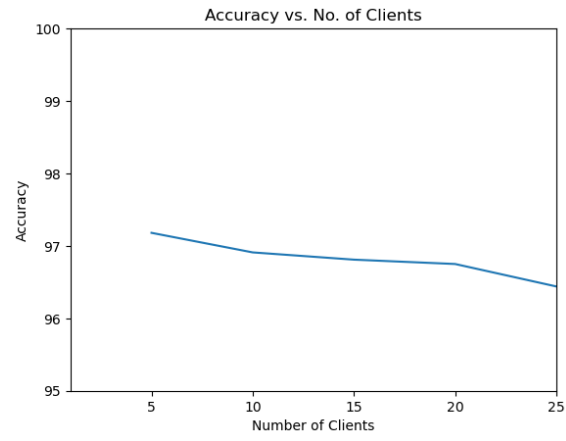


Figure 5.18. Change of Accuracy with respect to Number of Clients for FedAvg

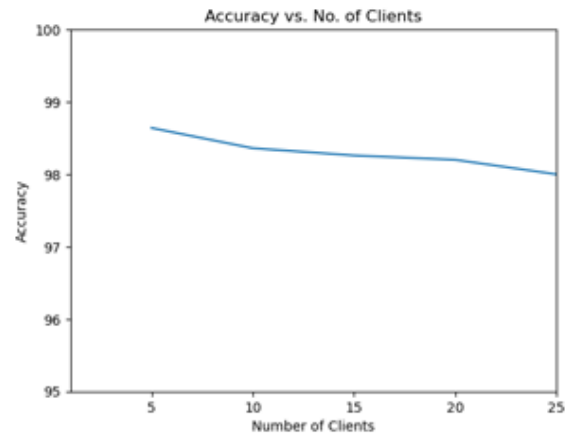


Figure 5.19. Change of Accuracy with respect to Number of Clients for FedProx

5.5.2 Variation of Loss with respect to Communication Rounds

Fig 5.20 and 5.21 illustrate the variation of Loss with varying number of communication rounds for both the aggregation methods using N-BaIoT dataset respectively. Using FedAvg, loss has decreased gradually with increasing number of communication rounds.

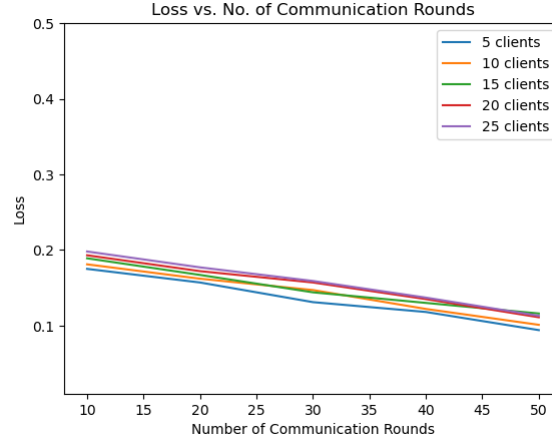


Figure 5.20. Change of Loss with respect to No. of Communication Rounds for FedAvg

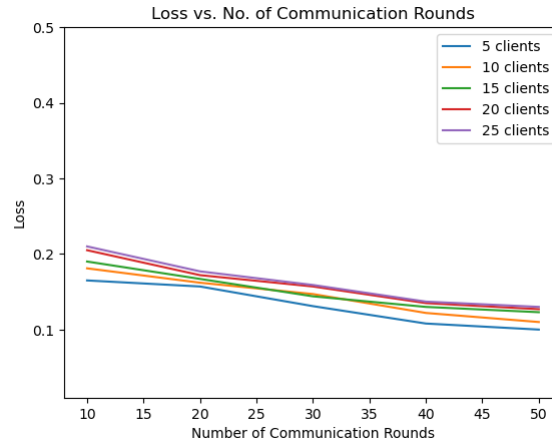


Figure 5.21. Change of Loss with respect to No. of Communication Rounds for FedProx

5.5.3 Variation of Accuracy with respect to Communication Rounds for varying noise multipliers

In Fig. 5.22, 5.23 and 5.24 the change of accuracy for varying noise multiplier ranging from 0.25 to 1 is illustrated using 5,15 and 25 clients respectively.

The change of accuracy is observed with respect to communication rounds for FedAvg

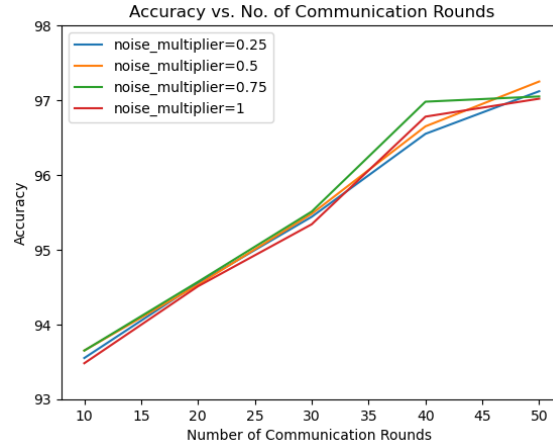


Figure 5.22. Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=5$

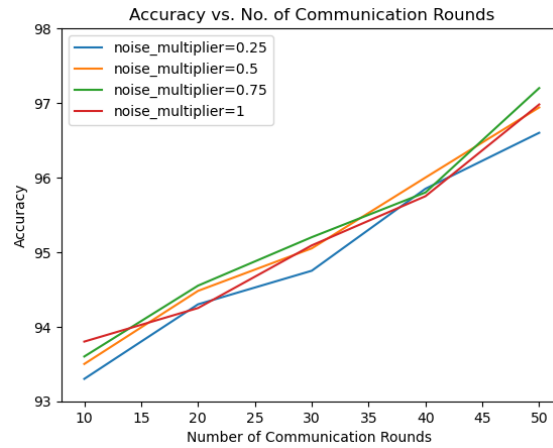


Figure 5.23. Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=15$

and FedProx. For FedAvg, when the client number is 5, accuracy for noise multipliers 0.75 and 1 start to decrease after 40 communication rounds whereas the accuracy curve is quite stable for noise multipliers 0.25 and 0.5. Accuracy curves are fluctuating for noise multiplier values of 0.25, 0.5 and 1 when the number of clients is 15. For 25 clients, Accuracy for noise multipliers 0.25 and 1 decrease after a certain number of communication rounds, but the accuracy curves remain stable for 0.5 and 0.75.

Fig. 5.25, 5.26 and 5.27 describe the change of accuracy for varying noise multiplier

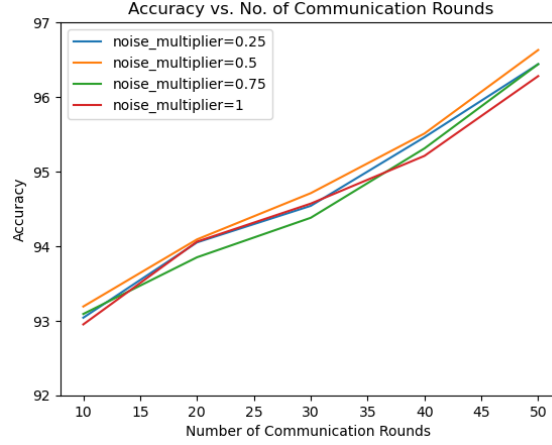


Figure 5.24. Change of Accuracy vs. No. of Communication Rounds using FedAvg for different values of noise multipliers when clients, $k=25$

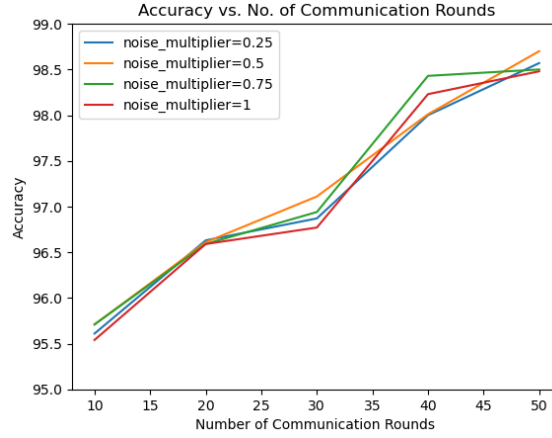


Figure 5.25. Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=5$

ranging from 0.25 to 1 is illustrated using 5, 15 and 25 clients for the case of FedProx, respectively. Using FedProx, when client number is 5, graphs for noise multiplier 0.75 and 1 are fluctuating and unstable. For 0.25 and 0.5, accuracy changes with communication rounds in a stable manner but we get better accuracy for noise multiplier of 0.5. For 15 and 25 clients respectively, we get unstable and fluctuating curves for noise multipliers 0.25, 0.75 and 1. Noise multiplier of 0.5 gives a better performance compared to the others. It is observed that, For the number of clients varying from 5 to 25 in both the datasets, noise multiplier of 0.5 provides a stable accuracy curve which sequentially increases with the number of communication rounds.

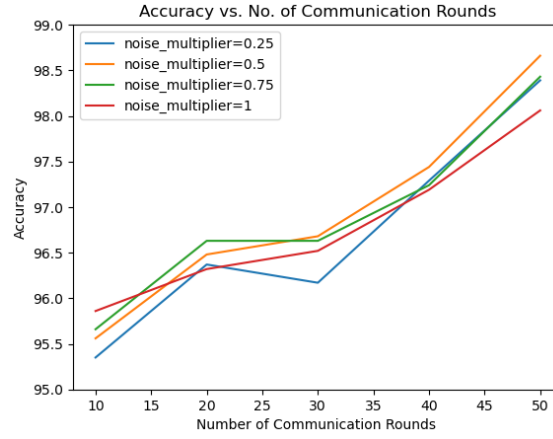


Figure 5.26. Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=15$

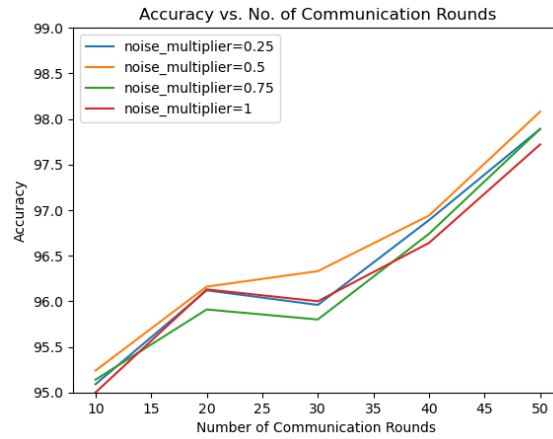


Figure 5.27. Change of Accuracy vs. No. of Communication Rounds using FedProx for different values of noise multipliers when clients, $k=25$

Table 5.3 and 5.4 describe the complete statistics of change of accuracy for different values of noise multiplier for varying number of clients using FedAvg and FedProx on N-BaIoT dataset respectively.

Table 5.3. Change of Accuracy for different noise multipliers for varying number of clients using FedAvg on N-BaIoT dataset

Number of Clients	Noise Multiplier	Accuracy for Communication Rounds (FedAvg) (%)				
		10	20	30	40	50
5	0.25	93.55	94.55	95.44	96.55	97.12
	0.5	93.65	94.53	95.48	96.65	97.25
	0.75	93.65	94.57	95.51	96.98	97.05
	1.00	93.48	94.51	95.34	96.78	97.02
15	0.25	93.30	94.30	94.75	95.85	96.60
	0.5	93.50	94.48	95.05	96.00	96.94
	0.75	93.60	94.55	95.20	95.80	97.20
	1.00	93.80	94.25	95.09	95.75	96.98
25	0.25	93.04	94.05	94.54	95.46	96.44
	0.5	93.19	94.09	94.71	95.51	96.63
	0.75	93.09	93.85	94.38	95.31	96.44
	1.00	92.95	94.06	94.57	95.21	96.28

Table 5.4. Change of Accuracy for different noise multipliers for varying number of clients using FedProx on N-BaIoT dataset

Number of Clients	Noise Multiplier	Accuracy for Communication Rounds (FedProx) (%)				
		10	20	30	40	50
5	0.25	95.61	96.63	96.87	98.00	98.57
	0.5	95.71	96.61	97.11	98.01	98.70
	0.75	95.71	96.59	96.94	98.43	98.50
	1.00	95.54	96.59	96.77	98.23	98.48
15	0.25	95.35	96.37	96.17	97.29	98.39
	0.5	95.56	96.48	96.68	97.44	98.66
	0.75	95.66	96.63	96.63	97.24	98.43
	1.00	95.86	96.32	96.52	97.19	98.06
25	0.25	95.09	96.12	95.96	96.89	97.89
	0.5	95.24	96.16	96.33	96.94	98.08
	0.75	95.14	95.91	95.80	96.74	97.89
	1.00	95.00	96.13	96.00	96.64	97.72

5.5.4 Variation of Accuracy with respect to Epsilon for varying delta

The N-BaIoT dataset is used in Figures 5.28 and 5.29 to show how the epsilon and accuracy change for various delta values for each of the two aggregation methods. Similar to the BoT-IoT dataset, Epsilon is smaller for accuracy for a smaller value of delta, indicating a lower loss of privacy. Epsilon rises in tandem with delta, signifying a greater loss of privacy.

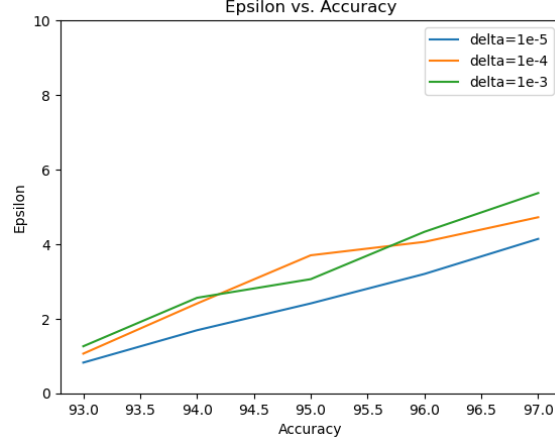


Figure 5.28. Change of Epsilon with respect to Accuracy for different values of delta for FedAvg

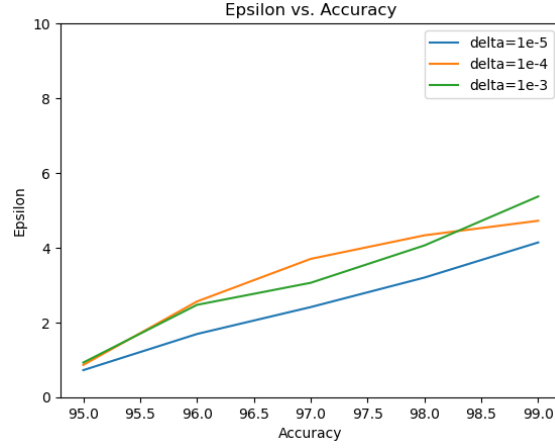


Figure 5.29. Change of Epsilon with respect to Accuracy for different values of delta for FedProx

5.6 Comparison with Existing Research Work

In this research, we discuss our approach for boosting security in the IoT context that makes use of a BiLSTM-CNN model. We effectively incorporated DP principles with Gaussian noise into our design. Then we implemented two aggregation methods, FedAvg and FedProx, to make a fair comparison and choose the better aggregation algorithm. In the context of IoT intrusion detection, our work on the privacy loss with respect to epsilon (ϵ) and delta (δ) has provided insight on the trade-off between privacy and model accuracy. As previously noted, despite the privacy benefits afforded by FL, recent study indicated that multiple attacks are still conceivable during the federated training process, for which privacy protecting measures are in use. We have added a comprehensive comparison of our proposed approach with relevant IDSs that use federated learning and DP in Table 5.5.

For BoT-IoT dataset, We have compared the federated approaches with centralized learning. Centralized system has an accuracy of 98.57% which is a bit higher than FedAvg but lower

Table 5.5. Comparison with existing works using BoT-IoT and N-BaIoT datasets

References	Dataset Used	Algorithm/ Model Implemented	Consideration of Differential Privacy	Privacy Parameters Considered	FL Accuracy
Ferrag et al. [44]	BoT-IoT	DNN, CNN, RNN, FedAvg	No	-	DNN(91.98%), CNN(89.91%), RNN(88.56%)
Rey et al. [29]	N-BaIoT	MLP, Multi-Epoch Avg, Mini-Batch Avg	No	-	99%
Poopola et al. [24]	BoT-IoT, N-BaIoT	LAE-LSTM, FedAvg	No	-	99%
Dong et al. [45]	BoT-IoT	MLP, FedAvg, FedForest	Yes	ϵ	FedAvg(81.36%), FedForest(89.63%)
Our Proposed Model	BoT-IoT, N-BaIoT	BiLSTM-CNN, FedAvg, FedProx	Yes	ϵ, δ	98% (BoT-IoT, FedAvg), 97.20% (N-BaIoT, FedAvg), 99.24% (BoT-IoT, FedProx) 98.60% (N-BaIoT, FedProx)

than FedProx. The way each approach manages data distribution and model aggregation is the leading cause of the accuracy discrepancies between FedAvg, FedProx, and centralized learning. FedAvg does not account for data heterogeneity while aggregating local model updates from customers. Despite its 98% accuracy, it may have trouble processing non-iid data, which is common in IoT environments. As a result, local model divergence and reduced generalization may occur. By including a regularization element that incentivizes local models to remain nearer the global model, FedProx outperforms FedAvg. Even with different data distributions, this improves federated learning performance (99.24%) by reducing the effect of data heterogeneity across IoT devices. Because centralized learning has access to the entire dataset, it can learn from all of the data simultaneously, which results in a little better performance than FedAvg (98.57%). Additionally, because it lacks the benefits of federated approaches considering device-to-device data variety, it performs marginally worse than FedProx. Unlike centralized learning, which has such built-in methods, FedProx's regularization aids the model in federated learning by improving its ability to adapt to dispersed data, resulting in increased accuracy.

5.7 Conclusion

In summary it can be said that, in contrast to other techniques, our research assessed a customized BiLSTM-CNN model merged with Gaussian DP in FL scenario that is applied to the recent BoT-IoT and N-BaIoT datasets, which contain various forms of IoT attacks. We have focused on building up a model which can strike balance between model's accuracy and data privacy. We focused on two different aggregation methods, namely, FedAvg and FedProx, while FedProx has shown a better accuracy proving itself to be more suitable. We checked the change of accuracy for different amount of random noises to integrate differential privacy with the proposed BiLSTM-CNN model. Our approach concludes that a noise multiplier value around 0.5 minimizes the accuracy-privacy tradeoff for our proposed model for both FedAvg and FedProx.

Conclusion

The research offers an in-depth strategy for IoT intrusion detection that protects privacy. It presents a privacy-conscious architecture that keeps fair intrusion detection performance while protecting individual data points with differential privacy protections. The proposed BiLSTM CNN architecture strikes a balance between model complexity and intrusion detection accuracy, making it well suited for resource-constrained IoT devices. This research makes a fair comparison between FedAvg and FedProx, proving FedProx as a better aggregation method for the proposed architecture as it considers the partial contributions of the clients. This study helps practitioners choose acceptable privacy criteria by providing an in-depth examination of privacy loss with varied epsilon and delta values. The viability of federated learning in IoT intrusion detection is confirmed through client assessments comprising 5 to 25 clients across 50 communication cycles, demonstrating scalability and durability.

The key findings of this study can be summarized as follows:

- **Privacy-Preserving Intrusion Detection:** Our suggested architecture ensures IoT data privacy while maintaining competitive intrusion detection performance. We secure individual data points from inference assaults by introducing differential privacy protections, even as the model learns and adapts.
- **Exploring Different Aggregation Algorithms:** We have explored two different aggregation methods in our research to make a fair comparison and to pick the more suitable one for our proposed architecture. FedProx and FedAvg, both are stable for the same range of noise multiplier but FedProx has shown a higher accuracy.
- **Privacy Loss Analysis:** We conducted an in-depth investigation of privacy loss with varied epsilon (ϵ) and delta (δ) values, demonstrating how these factors affect the trade-off between privacy and accuracy. This research assists practitioners and system designers in selecting privacy criteria that are appropriate for their individual use cases.
- **Client Evaluation:** We tested the architecture with 5 to 25 clients over 50 communication cycles for both the dataset to demonstrate its scalability and resilience.

The results show that federated learning is feasible in the context of IoT intrusion detection, even with a high number of participating clients.

For future work, several avenues are worth exploring:

- **Advanced Differential Privacy Techniques:** To further enhance privacy assurances while reducing the negative impact on model accuracy, look into more sophisticated differential privacy strategies.
- **Optimization Strategies:** Investigate optimization methods like model compression, quantization, and adaptive learning rate tuning to boost the effectiveness of federated learning in an IoT setting.
- **Real-World Deployment:** Assess the viability and performance of our architecture in various IoT contexts and deployment scenarios using real-world experiments and simulations.

In conclusion, by combining differential privacy with a customized BiLSTM CNN model in FL architecture, our research offers a promising step toward intrusion detection with privacy preservation in the IoT. Addressing privacy issues will remain a crucial area of research as the IoT landscape changes, and our work lays a solid platform for future efforts in this regard.

Bibliography

- [1] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, “Security and privacy challenges in industrial internet of things,” in *Proceedings of the 52nd annual design automation conference*, 2015, pp. 1–6.
- [2] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, “Taxonomy and survey of collaborative intrusion detection,” *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, pp. 1–33, 2015.
- [3] I. H. Sarker, “Cyberlearning: Effectiveness analysis of machine learning security modeling to detect cyber-anomalies and multi-attacks,” *Internet of Things*, vol. 14, p. 100393, 2021.
- [4] Z. Ji, Z. C. Lipton, and C. Elkan, “Differential privacy and machine learning: A survey and review. corr abs/1412.7584 (2014),” *arXiv preprint arxiv:1412.7584*, 2014.
- [5] I. H. Sarker, *AI-driven cybersecurity and threat intelligence: cyber automation, intelligent decision-making and explainability*. Springer, 2024.
- [6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial intelligence and statistics*, PMLR, 2017, pp. 1273–1282.
- [7] P. Ruzafa-Alcázar, P. Fernández-Saura, E. Mármol-Campos, *et al.*, “Intrusion detection based on privacy-preserving federated learning for the industrial iot,” *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1145–1154, 2021.
- [8] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, “Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset,” *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.
- [9] Y. Meidan, M. Bohadana, Y. Mathov, *et al.*, “N-baiot—network-based detection of iot botnet attacks using deep autoencoders,” *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.
- [10] P. Liu, X. Xu, and W. Wang, “Threats, attacks and defenses to federated learning: Issues, taxonomy and perspectives,” *Cybersecurity*, vol. 5, no. 1, pp. 1–19, 2022.
- [11] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

- [12] G. Andrew, O. Thakkar, B. McMahan, and S. Ramaswamy, “Differentially private learning with adaptive clipping,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 455–17 466, 2021.
- [13] R. C. Geyer, T. Klein, and M. Nabi, “Differentially private federated learning: A client level perspective,” *arXiv preprint arXiv:1712.07557*, 2017.
- [14] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, “Learning differentially private recurrent language models,” *arXiv preprint arXiv:1710.06963*, 2017.
- [15] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our data, ourselves: Privacy via distributed noise generation,” in *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*, Springer, 2006, pp. 486–503.
- [16] Z. Chuanxin, S. Yi, and W. Degang, “Federated learning with gaussian differential privacy,” in *Proceedings of the 2020 2nd international conference on robotics, intelligent control and artificial intelligence*, 2020, pp. 296–301.
- [17] C. Dwork, A. Roth, *et al.*, “The algorithmic foundations of differential privacy,” *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.
- [18] Z. Cai and Q. Chen, “Latency-and-coverage aware data aggregation scheduling for multihop battery-free wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1770–1784, 2020.
- [19] R. Zhao, G. Gui, Z. Xue, *et al.*, “A novel intrusion detection method based on lightweight neural network for internet of things,” *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9960–9972, 2021.
- [20] C. Yin, Y. Zhu, J. Fei, and X. He, “A deep learning approach for intrusion detection using recurrent neural networks,” *Ieee Access*, vol. 5, pp. 21 954–21 961, 2017.
- [21] H. He, X. Sun, H. He, G. Zhao, L. He, and J. Ren, “A novel multimodal-sequential approach based on multi-view features for network intrusion detection,” *IEEE Access*, vol. 7, pp. 183 207–183 221, 2019.
- [22] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, “A hybrid deep learning-based model for anomaly detection in cloud datacenter networks,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 924–935, 2019.
- [23] X. Xu, J. Li, Y. Yang, and F. Shen, “Toward effective intrusion detection using log-cosh conditional variational autoencoder,” *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6187–6196, 2020.

- [24] S. I. Popoola, B. Adebisi, M. Hammoudeh, G. Gui, and H. Gacanin, “Hybrid deep learning for botnet attack detection in the internet-of-things networks,” *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4944–4956, 2020.
- [25] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: An ensemble of autoencoders for online network intrusion detection,” *arXiv preprint arXiv:1802.09089*, 2018.
- [26] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, “A hierarchical hybrid intrusion detection approach in iot scenarios,” in *GLOBECOM 2020-2020 IEEE global communications conference*, IEEE, 2020, pp. 1–7.
- [27] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: Techniques, datasets and challenges,” *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.
- [28] K. A. Da Costa, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. de Albuquerque, “Internet of things: A survey on machine learning-based intrusion detection approaches,” *Computer Networks*, vol. 151, pp. 147–157, 2019.
- [29] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, “Federated learning for malware detection in iot devices,” *Computer Networks*, vol. 204, p. 108 693, 2022.
- [30] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, K.-K. R. Choo, and M. Nafaa, “Felids: Federated learning-based intrusion detection system for agricultural internet of things,” *Journal of Parallel and Distributed Computing*, vol. 165, pp. 17–31, 2022.
- [31] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, “Diot: A federated self-learning anomaly detection system for iot,” in *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*, IEEE, 2019, pp. 756–767.
- [32] Y. Liu, S. Garg, J. Nie, *et al.*, “Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach,” *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2020.
- [33] T. V. Khoa, Y. M. Saputra, D. T. Hoang, *et al.*, “Collaborative learning model for cyberattack detection systems in iot industry 4.0,” in *2020 IEEE wireless communications and networking conference (WCNC)*, IEEE, 2020, pp. 1–6.
- [34] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, “Chained anomaly detection models for federated learning: An intrusion detection case study,” *Applied Sciences*, vol. 8, no. 12, p. 2663, 2018.
- [35] R. Zhao, Y. Yin, Y. Shi, and Z. Xue, “Intelligent intrusion detection based on federated learning aided long short-term memory,” *Physical Communication*, vol. 42, p. 101 157, 2020.

- [36] I. Siachos, K. Kaltakis, K. Papachristopoulou, I. Giannoulakis, and E. Kafetzakis, "Comparison of machine learning algorithms trained under differential privacy for intrusion detection systems," in *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*, IEEE, 2023, pp. 654–658.
- [37] A. Alazab, A. Khraisat, S. Singh, and T. Jan, "Enhancing privacy-preserving intrusion detection through federated learning," *Electronics*, vol. 12, no. 16, p. 3382, 2023.
- [38] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2019.
- [39] B. Zhao, K. Fan, K. Yang, Z. Wang, H. Li, and Y. Yang, "Anonymous and privacy-preserving federated learning with industrial big data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 6314–6323, 2021.
- [40] J. Li, L. Lyu, X. Liu, X. Zhang, and X. Lyu, "Fleam: A federated learning empowered architecture to mitigate ddos in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4059–4068, 2021.
- [41] D. C. Attota, V. Mothukuri, R. M. Parizi, and S. Pouriyeh, "An ensemble multi-view federated learning intrusion detection for iot," *IEEE Access*, vol. 9, pp. 117 734–117 745, 2021.
- [42] D. Man, F. Zeng, W. Yang, M. Yu, J. Lv, and Y. Wang, "Intelligent intrusion detection based on federated learning for edge-assisted internet of things," *Security and Communication Networks*, vol. 2021, no. 1, p. 9 361 348, 2021.
- [43] C. He, M. Annavaram, and S. Avestimehr, "Fednas: Federated deep learning via neural architecture search, 2020," *Cited on page*, vol. 34,
- [44] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis," *IEEE Access*, vol. 9, pp. 138 509–138 542, 2021.
- [45] T. Dong, S. Li, H. Qiu, and J. Lu, "An interpretable federated learning-based network intrusion detection framework," *arXiv preprint arXiv:2201.03134*, 2022.
- [46] Y. Zhao, J. Zhao, L. Jiang, *et al.*, "Privacy-preserving blockchain-based federated learning for iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1817–1829, 2020.
- [47] M. Abadi, A. Chu, I. Goodfellow, *et al.*, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

- [48] M. M. Rashid, S. U. Khan, F. Eusufzai, M. A. Redwan, S. R. Sabuj, and M. Elsharief, “A federated learning-based approach for improving intrusion detection in industrial internet of things networks,” *Network*, vol. 3, no. 1, pp. 158–179, 2023.
- [49] O. Friha, M. A. Ferrag, M. Benbouzid, T. Berghout, B. Kantarci, and K.-K. R. Choo, “2df-ids: Decentralized and differentially private federated learning-based intrusion detection system for industrial iot,” *Computers & Security*, vol. 127, p. 103 097, 2023.
- [50] Y. Zhao, W. Xiao, L. Shuai, J. Luo, S. Yao, and M. Zhang, “A differential privacy-enhanced federated learning method for short-term household load forecasting in smart grid,” in *2021 7th International Conference on Computer and Communications (ICCC)*, IEEE, 2021, pp. 1399–1404.
- [51] J. Li, Y. Meng, L. Ma, *et al.*, “A federated learning based privacy-preserving smart healthcare system,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, 2021.
- [52] D. Hwang, H. Mun, and Y. Lee, “Improving response time of home iot services in federated learning,” in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 2022, pp. 157–163.

A Differential Privacy Aided Federated Learning Model for Intrusion Detection in IoT Networks.pdf

ORIGINALITY REPORT

19%	15%	15%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dokumen.pub Internet Source	1%
2	Yihan Xiao, Cheng Xing, Taining Zhang, Zhongkai Zhao. "An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks", IEEE Access, 2019 Publication	1%
3	courses.cs.ut.ee Internet Source	1%
4	doaj.org Internet Source	1%
5	"Innovations in Computational Intelligence and Computer Vision", Springer Science and Business Media LLC, 2024 Publication	1%
6	arxiv.org Internet Source	1%
7	file.techscience.com	