UNIVERSITY OF TURKU

Smart Home Anomaly Recognition and Prevention (SHARP)

Network Intrusion Detection system utilising Unsupervised Learning and Reinforcement Learning

Cyber Security
Master's Degree Programme in Information and Communication Technology
Department of Computing, Faculty of Technology
Master of Science in Technology Thesis

Author:
Irene Chua

Supervisors:
Petri Sainio
Tahir Mohammad

May 2025

Master of Science in Technology Thesis
Department of Computing, Faculty of Technology
University of Turku

Subject: Cyber Security
Programme: Master's Degree Programme in Information and Communication Technology
Author: Irene Chua
Title: Smart Home Anomaly Recognition and Prevention (SHARP)
Number of pages: 92 pages, 0 appendix pages
Date: May 2025

The proliferation of smart home networks consisting of the Internet of Things (IoT) and smart devices has led to a rapidly expanding heterogeneous ecosystem, thereby increasing the attack surface and security vulnerabilities. Whereas traditional supervised Machine Learning (ML) Anomaly-based Network Intrusion Detection Systems (ANIDS) addressed the evolving threat landscape of smart home networks, they face constraints in identifying unknown attacks and lack adaptability in dynamic environments due to the absence of labelled data.

This study reveals a lack of literature concerning ML-based ANIDS utilising unlabelled data while adapting to a dynamic environment. This study introduces a Smart Home Anomaly Recognition and Prevention framework (SHARP), which integrates Unsupervised Learning (UL) and Reinforcement Learning (RL) to identify Distributed Denial of Service (DDoS) cyber-attacks in smart home networks.

SHARP employs an Unsupervised Ensemble (UE) consisting of three unsupervised classifiers to integrate UL and utilises RL's Exponential weight for Exploration and Exploitation with Experts (EXP4) algorithm to dynamically optimise the UE anomaly score based on varying environmental states. The findings show that SHARP outperforms existing UE methods, achieving an increase of 7.46% in accuracy, 11.91% in precision, 34.27% in recall and a 22.85% increase in the F1-score. SHARP continuously optimises itself by observing the environment without the necessity of labelled data, presenting a promising solution that addresses security vulnerabilities in the rapidly evolving threat landscape of smart home networks.

Keywords: Cybersecurity, Cyber-attacks, Internet of Things, Smart home, Machine Learning, Unsupervised Learning, Reinforcement Learning, Network Security, Network Intrusion Detection System, Anomaly Detection, DDoS, Unsupervised Ensemble, EXP4.

Acknowledgement

First and foremost, I would like to express my gratitude to my supervisors, Petri Sainio and Tahir Mohammad, for their guidance, support, and insightful feedback throughout this research.

I am profoundly thankful to my family and partner for their unconditional love and encouragement during this journey. Their faith in my capabilities encouraged me to persevere through challenging situations. Furthermore, I extend my sincere appreciation to my friends, whose moral support has been a source of comfort.

Additionally, I acknowledge the use of QuillBot and Grammarly, which were employed to enhance sentence structures and grammatical accuracy in this research. These tools were utilised exclusively to enhance linguistic clarity, while all intellectual content and interpretation remain my own, ensuring the authenticity and integrity of this research.

Table of contents

List of Figures

List of Tables

List of Acronyms

AMQP	Advanced Message Queuing Protocol
ANIDS	Anomaly-based Network Intrusion Detection System
ANOVA	Analysis of Variance
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
CSV	Comma Separated Values
DDoS	Distributed Denial of Service
DL	Deep Learning
DoS	Denial of Service
DTLS	Datagram Transport Layer Security
ECG	Electrocardiography
EL	Ensemble Learning
ERL	Ensemble Reinforcement Learning
EXP4	Exponential weight for Exploration and Exploitation with Experts
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
HSD	Honest Significant Difference
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
iForest	Isolation Forest
IoT	Internet of Things
IP	Internet Protocol
LAN	Local Area Networks
LOF	Local Outlier Factor
MAB	Multi Arm Bandit
MitM	Man In the Middle
ML	Machine Learning
MQTT	Message Queue Telemetry Transport
NIDS	Network Intrusion Detection System
OCSVM	One Class Support Vector Machine
OSI	Open Systems Interconnection
PAN	Personal Area Networks
PCAP	Packet Capture

RL	Reinforcement Learning
RPL	Routing Protocol For Low-Power Lossy Networks
SHARP	Smart Home Anomaly Recognition and Prevention
SL	Supervised Learning
SSL	Semi-Supervised Learning
TCP	Transmission Control Protocol
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TTL	Time To Live
UDP	User Datagram Protocol
UE	Unsupervised Ensemble
UL	Unsupervised Learning
WLAN	Wireless Local Area Networks
WPAN	Wireless Personal Area Networks
XMPP	eXtensible Messaging and Presence Protocol
6LowPAN	Ipv6 Over Low-Power Wireless Personal Area Networks

1 Introduction

The Internet of Things (IoT) was quickly adopted in our digital society by numerous industries, including the commercial sector, the medical industry, and smart cities. Consequently, the development and popularity of smart homes allowed IoT to become part of our daily lives. Smart homes became trending due to the lifestyle-enhancing capabilities of IoT and smart devices. They quickly expanded in size due to the fast development of new IoT and smart devices driven by the commercial sector. As the landscape of smart home environments rapidly expanded, smart home networks increasingly became vulnerable to cyber-attacks. When compromised, smart home networks can become digital jackpots that may expose personal information and daily routines. Additionally, compromised smart home networks can enable criminals to perform illegal activities such as e-espionage, voyeurism, and physical burglary.

1.1 Problem statement

To prevent smart home networks from getting compromised, the academic community has extensively examined the efficiency of Machine Learning (ML)-based Anomaly Network Intrusion Detection Systems (ANIDS) to detect cyber-attacks within smart home networks. Prior studies have achieved excellent performance in detecting cyber-attacks in smart home networks using ANIDS, which apply supervised ML methods. However, this approach faces constraints due to the scarcity of labelled data in a real smart home network and the inability to process data in a dynamic environment. This study reveals a gap in the existing literature on ML-based ANIDS that can identify cyber-attacks while incorporating unlabelled data and facilitating adaptability to a dynamic smart home environment. Consequently, this study addresses the gap by introducing a Smart Home Anomaly Recognition and Prevention (SHARP) framework, which incorporates elements of Unsupervised Learning (UL) as well as Reinforcement Learning (RL) to identify Distributed Denial of Service (DDoS) cyber-attacks in smart home networks. SHARP combines an Unsupervised Ensemble (UE) consisting of three unsupervised classifiers with the Exponential weight for Exploration and Exploitation with Experts (EXP4) algorithm to create a dynamic weighing system that produces an ensemble anomaly score. The ensemble anomaly score represents the UE's confidence in whether the data is considered anomalous. SHARP leverages the strengths of every individual unsupervised classifier in the UE by employing an iterative weight adjustment mechanism using the RL algorithm EXP4, which adjusts the importance of the unsupervised classifier evaluation results by executing an action and observing the response of the smart home network.

1.2 Research questions

The primary research question for this study is formulated as follows:

"How can unsupervised learning and reinforcement learning be incorporated in an ML-based ANIDS to detect and respond to cyber-attacks in a smart home network?"

Three sub-questions are formulated that address the information required to answer the primary research question:

1. Which anomaly detection techniques are applied in ANIDS to detect cyber-attacks in a smart home network?
2. How can an ML-based ANIDS utilise unsupervised learning and reinforcement learning to enhance the detection of anomalies in a smart home network?
3. How effective is the ML-based ANIDS framework that utilises UL and RL?

1.3 Research objectives

The research objectives for this study are as follows:

1. To review the existing literature for state-of-the-art ANIDS for the smart home environment.
2. To design and implement a framework that combines the strengths of UL and RL to detect and respond to DDoS attacks in a smart home network.
3. To evaluate the performance of the proposed framework and benchmark it against the existing literature.

1.4 Thesis organisation

The rest of the thesis is organised as follows: Chapter 2 provides the literature analysis for the three sub-questions. Chapter 3 discusses the methodology and design of the proposed framework for the ML-based ANIDS, referred to as SHARP. Chapter 4 elaborates on SHARP's practical implementation and validation. Next, the results of this research are examined in Chapter 5, and a discussion is presented in Chapter 6. Finally, Chapter 7 presents the conclusion alongside the limitations and recommendations of this research, followed by suggestions for future work.

2 Literature Review

This chapter offers a comprehensive literature review concerning anomaly detection, the architecture of smart home networks, cyber-attacks specifically targeting smart homes, and the application of UE and RL for anomaly detection.

2.1 Literature review methodology

The literature review of this study is constructed by gathering and analysing a collection of scientific articles. The study leveraged databases, including ScienceDirect, SpringerLink, ACM, IEEE, arXiv and Google Scholar, to find articles relevant to this study. Search terminologies included anomaly detection, anomaly detection techniques, anomaly network detection, intrusion detection, anomaly detection and machine learning, smart home networks, IoT and smart home security, (cyber) attacks in smart home networks, ensemble learning, reinforcement learning, unsupervised ensemble learning and unsupervised ensemble reinforcement learning. Most of the scientific articles used in this study were published between 2015 and 2025. However, twelve older articles dated between 2000 and 2015 were still used in this study due to the information deemed relevant to the topic.

2.2 Anomaly detection systems

This subchapter reviews the existing literature concerning anomalies, activities and methodologies used for anomaly detection, and the application of ML techniques for network anomaly detection. The findings gathered from the literature review serve as the foundation for a generic ML-based ANIDS.

2.2.1 Anomaly types

The term “anomaly” is used across various research domains and generally refers to anything that deviates from a particular set of norms (Ahmed et al., 2016; Chandola et al., 2009; Mane et al., 2022). In cybersecurity, identifying anomalies in a system can shed light on issues such as misconfigurations or malicious cyber-attacks that can disrupt a system’s functionality (Ahmed et al., 2016). Therefore, it is essential to comprehend, analyse and detect anomalies to control anomalous behaviour and improve a system’s security.

Literature suggests that anomalies can be categorised into three types (Ahmed et al., 2016; Alabadi & Çelik, 2020; Chandola et al., 2009; Fahim & Sillitti, 2019; Nassif et al., 2021; Rokhade et al., 2023):

1. *Point anomalies* refer to the abnormal behaviour of a single data entry compared to the entire dataset. For instance, if a CPU maintains a temperature of approximately 40°C throughout its lifecycle but suddenly experiences a spike to 80°C one day, this abrupt increase would be considered a point anomaly.

2. *Contextual anomalies (also referred to as conditional anomalies)* refer to data entries that show anomalous behaviour based on a particular context. For instance, a backup software runs a weekly backup on Monday mornings at 08:00. One day, the software creates a backup on Thursday at 02:00 A.M. This situation suggests that the activity shows anomalous behaviour because it is an unusual time for the software to create a backup.

3. *Collective anomalies* refer to data entries that show anomalous behaviour compared to an entire dataset but may not appear anomalous when observed individually. For instance, a corporate email portal experiences multiple login attempts in a short period from different geolocations. While the individual login attempts may not seem anomalous, the simultaneous attempts indicate a collective anomaly.

2.2.2 Anomaly detection activities

Anomaly detection revolves around performing data analysis activities using anomaly detection techniques to identify anomalous patterns in the data (Ahmed et al., 2016; Chandola et al., 2009). Table 1 examines the key activities for anomaly detection by comparing the anomaly detection flowchart of Manimurugan et al. (2020) and the generic network anomaly detection framework of Ahmed et al. (2016).

Table 1. Comparing key activities in anomaly detection presented in literature.

Step	Ahmed et al. (2016)	Manimurugan et al. (2020)
1	Obtain input data.	Collect and understand data flow.
2	Data processing.	Identify anomaly type.
3	Anomaly detection technique.	Identify training data type.
4	Output (score or label).	Apply anomaly detection technique.
5	Evaluation.	Identify normal and anomalous behaviour.

The first activity in anomaly detection involves gathering data and understanding its characteristics. To spot potential anomalous data, one must first understand normal data behaviour. Next, data cleaning activities, such as removing unnecessary data, transform the data into training data. The third step decides which anomaly detection technique is suitable based on the available training data. The literature suggests that anomaly detection techniques can be categorised based on the nature of available training data (Alabadi & Çelik, 2020; Chandola et al., 2009; Manimurugan et al., 2020; Nassif et al., 2021):

- *Supervised anomaly detection*. Utilises training data consisting of labelled normal and anomalous data.
- *Semi-supervised anomaly detection*. Utilises training data that only contains labelled normal data.
- *Unsupervised anomaly detection*. Utilises training data that does not contain labels.

The fourth activity implements the anomaly detection technique in practice. The last activity produces an output that distinguishes anomalous behaviour from normal behaviour. Depending on the type of training data, the output format may differ. The output format can be presented in binary notation or as an anomaly score ranging from 0 to 1.

2.2.3 Anomaly detection application

While various domains apply anomaly detection, intrusion detection is a prominent research topic in the field of cybersecurity (Bhuyan et al., 2014; Nassif et al., 2021). Intrusion detection aims to detect unauthorised activities in a system or network to prevent compromise on data availability, confidentiality, and integrity (Bhuyan et al., 2014). Initially, Network Intrusion Detection Systems (NIDS) were developed to mitigate network intrusions by utilising signature-based detection and have shown excellent performance in identifying known attacks (Khraisat & Alazab, 2021). However, literature observed shortcomings of NIDS, such as its inability to identify unknown attacks due to the signature of unknown attacks not being registered in the signature system. Consequently, the academic community extensively explored the feasibility and optimisation of Anomaly Network Intrusion Detection Systems (ANIDS) (Bhuyan et al., 2014).

An ANIDS is engineered to learn a system's normal behaviour and recognise anomalous data when it shows deviating behaviour, suggesting that its ability to detect unknown attacks is correlated to its comprehension of normal network behaviour (Bhuyan et al., 2014; Khraisat & Alazab, 2021; Nassif et al., 2021). Furthermore, ANIDS are often developed and customised to integrate into a specific environment, such as a network or system. Although a lack of adaptability can be observed in ANIDS, it also makes it more challenging for attackers to target a system without drawing attention to themselves (Khraisat & Alazab, 2021; Patcha & Park, 2007). While ANIDS can be effectively employed to detect anomalies in systems, it comes with potential disadvantages, such as requiring high resources, producing high false positive rates, and facing deployment issues in different environments (García-Teodoro et al., 2009; Jyothsna, 2011; Khraisat & Alazab, 2021; Lundin & Jonsson, 2000; Patcha & Park, 2007).

Table 2. Comparing key activities in ANIDS presented in literature.

Step	**Bhuyan et al. (2014)**	**Khraisat & Alazab (2021)**
1	Collect network data.	Collect network data.
2	Pre-process data.	Pre-process data.
3	Anomaly detection technique.	Feature extraction.
4	Reference data.	Apply ML.
5	Comparing mechanism.	Anomaly detection technique.
6	Alert generation.	Alert generation.
7	Human analyst.	Response mechanism.

Table 2 presents the key activities in an ANIDS by comparing a general ANIDS architecture discussed by Bhuyan et al. (2014) with the features and components outlined by Khraisat and Alazab (2021). The literature shows that the key activities of ANIDS (Table 2) are similar to those of anomaly detection (Table 1).

The initial activity focuses on collecting network data to gain insight into the available data types. Additionally, the anomaly detection technique is selected depending on the network data. Next, network data is pre-processed and normalised to remove unwanted information. The third step implements the anomaly detection technique and identifies anomalous data points. This step necessitates the integration of a comparison mechanism, as the ANIDS must compare the potential anomalous data with normal data. Finally, an alert is generated that can activate a response mechanism, which either requires human intervention or enforces automatically executed actions.

2.2.4 Network anomaly detection and ML

Various anomaly detection techniques use different approaches to identify anomalous behaviour within a system. Literature analysis shows that anomaly detection techniques utilise various approaches, from conventional techniques like statistics and information theory to modern methods like ML (Chandola et al., 2009; Patcha & Park, 2007). This study investigates which anomaly detection techniques are applied in ANIDS for smart home networks. Thus, literature analysis has been performed on network anomaly detection techniques, and the findings are presented in Table 3.

Table 3. Comparing literature regarding commonly used network anomaly detection techniques.

		Network anomaly detection techniques									
		Statistical	Classification	Cluster	Soft computing	Knowledge based	Ensemble	Information theory	Nearest Neighbour	Spectral	Machine Learning
Authors	Zhang et al. (2009)	x	x								x
	Bhuyan et al. (2014)	x	x	x	x	x	x				x
	Ahmed et al. (2016)	x	x	x				x			x
	Fahim and Sillitti (2019)	x									x
	Chatterjee and Ahmed (2022)	x	x	x			x	x	x		x
	Araya and Rifà-Pous (2023)	x	x	x	x		x	x	x	x	x
	Rokhade et al. (2023)		x	x			x				x

Table 3 reveals that ML is a dominating technique used for network anomaly detection. The findings of Mahesh (2020), Nassif et al. (2021) and Zhang et al. (2009) suggest that employing ML to detect network anomalies is preferred because implementing automated operations autonomously using various algorithms enhances their performance. Three studies were compared to observe which ML methodologies are prevalent for network anomaly detection activities.

Six ML methodologies can be derived from the observations of Araya and Rifà-Pous (2023), Fahim and Sillitti (2019) and Jha et al. (2021) and are divided by their learning style:

1. *Supervised Learning (SL):* ML algorithms are trained and evaluated on labelled datasets requiring human intervention for labelling. SL aims to solve classification or regression-related issues (Jha et al., 2021).

2. *Unsupervised Learning (UL):* ML algorithms are trained and evaluated using unlabelled datasets and do not require human intervention, as no labels are required. UL typically learns autonomously by identifying patterns and observing correlations in data that address cluster-based problems (Jha et al., 2021).

3. *Semi-Supervised Learning (SSL):* ML algorithms are trained and evaluated using both labelled and unlabelled data to solve constrained clustering or semi-supervised classification issues (Jha et al., 2021).

4. *Ensemble Learning (EL):* Multiple ML algorithms are used to solve problems related to their performance (Mahesh, 2020). Several ML algorithms with varying parameter values produce different predictions, which are combined and evaluated. Depending on the availability of labels in the train and test data, supervised EL or unsupervised EL can be deployed (Das et al., 2020).

5. *Reinforcement Learning (RL):* ML algorithms learn to make decisions by interacting with an environment and utilising a learning agent to select actions that maximise the reward returned from the environment (Mahesh, 2020). RL's performance can be optimised using a reward-penalty-based system to evaluate the learning agent's decisions.

6. *Deep Learning (DL):* Aims to solve issues by identifying data patterns via multiple artificial neural network layers. These layers iteratively transform data and adjust according to the output of the previous layers (Alzubaidi et al., 2021).

Table 4. Comparing three workflows to establish ML-based network anomaly detection system.

Step	Gupta et al. (2022)	Rabbani et al. (2021)	Manhas and Kotwal (2021)
1	Problem conceptualization.	Obtain raw data.	Data collection.
2	Data aggregation.	Pre-processing.	Remove duplicate samples.
3	Data preparation.	ML algorithm.	Data preprocessing.
4	Preprocess data.	Detection and recognition.	Apply ML model.
5	Model development.		Detect intrusion on test data.
6	Development and evaluation.		Evaluate ML model.

Table 4 presents a literature comparison of the key activities of ML-based network anomaly detection systems. The first step is to collect and identify the available data. Determining the data characteristics and type facilitates the selection of a suitable ML learning category. Next, the data is transformed in the pre-processing step, which involves cleaning activities such as removing unnecessary information, normalising data types and extracting features (Kotsiantis et al., 2006). In addition, Manhas and Kotwal (2021) suggest dividing the dataset into 70% training data and 30% testing data after it has been pre-processed. The third step applies the chosen ML algorithm in practice, wherein the model is trained. This step produces an output as a score or label, which is evaluated in the final step, where the ML algorithm is evaluated by computing performance metrics such as accuracy, recall, precision or F1-score (Manhas & Kotwal, 2021).

The systematic literature review of Araya and Rifà-Pous (2023) observed that the effectiveness of the six ML methods of detecting cyber-attack anomalies in a smart home network differs. Their observations reveal that supervised EL produced the best performance results, followed by DL and SL. However, the dominating methods in terms of performance do not necessarily reflect the results in a practical, real-world scenario due to the dynamic nature of real-world environments and the fact that ML methods are usually evaluated in a controlled environment. Ensemble Reinforcement Learning (ERL) methods have emerged to address the gap in adaptability and demonstrated significant performance and reliability in solving complex problems in real-world environments (Song et al., 2023). The literature review by Song et al. (2023) illustrated that ERL has been widely adopted across various domains, including IoT computing and malware detection.

Furthermore, the literature review highlighted previous studies that effectively integrated multiple models for data classification and employed RL to dynamically adjust the learning agent to facilitate adaptability to real-world scenarios. Examining the studies in the literature analysis of Song et al. (2023) reveals that ERL approaches rely on labelled datasets, which present challenges when deploying ERL in real-world environments due to the scarcity of labelled data for the training and testing of the individual models. Unsupervised ERL can tackle the issue concerning the scarcity of labelled data by integrating unsupervised classifiers that leverage unlabelled data. However, a lack of literature regarding unsupervised ERL can be observed. Consequently, this study assesses the viability of combining unsupervised EL and RL to detect anomalies in a smart home network, promote the use of unlabelled data, and facilitate adaptability within dynamic environments.

2.2.5 ML-based ANIDS

The literature analysis reveals similarities between anomaly detection activities, ANIDS activities and ML-based network anomaly detection activities. This study combined the findings to create a general ML-based ANIDS workflow incorporating the elements of the previously observed activities into one framework (Figure 1).

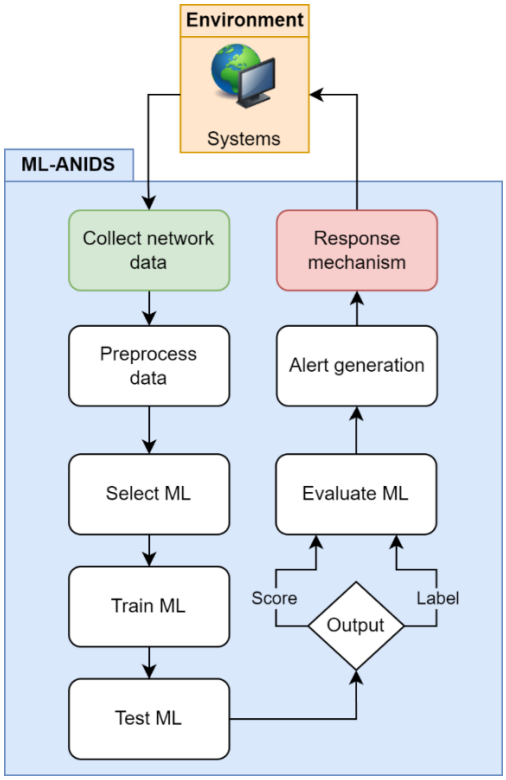

Figure 1. ML-based ANIDS workflow.

Figure 1 illustrates the activities of the ML-based ANIDS. The initial step in an ML-based ANIDS is collecting network data from the environment. It includes data analysis activities such as understanding data types flowing through the network and comprehending normal network behaviour. Next, data pre-processing is performed by cleaning and normalising the data, ensuring that unnecessary information is removed. Additionally, the data should be divided into a training and a testing dataset. Following this, one of the six ML methodologies and their corresponding algorithm is determined depending on the type of available training data. The anomaly detection mechanism is incorporated into the ML-based ANIDS workflow by training and testing the ML algorithms, leading to the identification of anomalous data. The results produced by the ML algorithms, presented as a score or label, are passed to the evaluation step, which determines whether anomalous data is present and examines the ML algorithm's performance using performance metrics. Once the result of the anomalous data is observed, an alert is generated that triggers an appropriate response from the response mechanism. The response can vary based on whether the reaction mechanism requires human intervention or executes actions automatically.

2.3 Smart homes

This subchapter presents a literature review concerning the characteristics of smart home environments and discusses their components.

2.3.1 IoT and Smart home characteristics

IoT significantly influenced Industry 4.0 as it enhanced and automated tasks in people's daily lives to raise the quality of life (HaddadPajouh et al., 2021; Lesch et al., 2023). Although there is no standard definition of IoT, Lesch et al.'s (2023) findings show that various pieces of literature share common elements in their definition of IoT. In the core concept of IoT, any physical object that can recognise, interact and communicate with itself and other objects across a network is considered a "thing" (Lesch et al., 2023). Things connected to the Internet essentially create a bridge that enhances communication between the Internet and physical objects, facilitating human interaction.

The presence of IoT emerged in domestic environments as they raise the quality of living for the home's occupants by managing and controlling devices within the household, contributing to the popularity of "smart homes" (Araya & Rifà-Pous, 2023; Geneiatakis et al., 2017; Khawla & Mazri, 2018; Mohanta et al., 2020; Yahya & Mohammed, 2023).

While literature analysis reveals diverse interpretations of smart home definitions, similarities in smart home components such as internet access, interconnected wired or wireless devices, and the ability to operate autonomously can be observed (Araya & Rifà-Pous, 2023; Geneiatakis et al., 2017; Khawla & Mazri, 2018; Mohanta et al., 2020; Yahya & Mohammed, 2023). Rather than analysing the definitions, Araya and Rifà-Pous (2023) and Mocrii et al. (2018) propose that analysing their characteristics provides more insight into smart homes.

Four characteristics of smart homes can be observed from these studies:

1. *Recognition of a smart home's condition:* The smart home can analyse its present condition. For instance, data gathered from motion sensors indicates that the occupant is physically present.

2. *Simultaneous analysis of factors*: The smart home can assess various factors simultaneously to conclude its current state. For instance, the smart home may detect that the occupant is departing when the motion sensor at the entrance detects movement and can visually confirm the scenario through the video footage from a camera.

3. *Behavioural pattern analysis:* The smart home can forecast the residents' condition by analysing the occupant's behavioural patterns. For example, if the occupant typically returns home by 17:00 on a weekday, the smart home may anticipate the occupant's return time.

4. *Autonomous actions:* Based on the smart home's observations and predictive capabilities, the smart home can perform (preventative) actions. For instance, if the occupant typically turns off all lights before leaving for work but fails to do so one day, the smart home should autonomously turn off the lights.

Observing the four characteristics shows that the smart home is an environment that requires data to be processed, stored and evaluated to understand the occupant's behavioural activities and make predictions to initiate specific actions (Araya & Rifà-Pous, 2023). Based on the findings of the literature, this study considers smart homes as environments composed of wired or wireless devices with Internet access that aim to enhance the occupants' living conditions by managing data, analysing behavioural patterns, and executing appropriate actions based on their predictive capabilities.

2.3.2 Smart home components

According to Geneiatakis et al. (2017), there is a lack of standardisation in smart home architectures primarily due to the heterogeneity of smart home components and the lack of standardisation regarding device compatibility. To gain insight into the components within a smart home environment, various studies have been examined to identify its components. Table 5 presents the smart home components observed across six different studies. The comparison reveals that studies address similar components with different terminology.

Table 5. Comparing six smart home architecture components.

Yang et al. (2016)	**Geneiatakis et al. (2017)**	**Khawla and Mazri (2018)**	**Tao et al. (2018)**	**Mocrii et al. (2018)**	**Shouran et al. (2019)**
Sensor layer	IoT devices	Connected objects	IoT devices	Edge: containing end-devices.	Smart devices
Home logic layer	Smart hub	Smart home gateway	Communication technology	Communication protocols	Home network
Cloud logic layer	Home router	Cloud	Middleware	Gateway	Home gateway
Data access layer	Internet	Applications	Cloud	Cloud	Cloud
	Cloud	Communication protocols			Service platforms

Yang et al. (2016) proposed a four-layered architecture consisting of a sensor layer, home logic layer, Cloud logic layer and data access layer. The sensor layer contains sensor devices that monitor and control the home environment; the home logic layer serves as the bridge between the sensor devices and the home occupant; the Cloud logic layer is responsible for processing and distributing data; and the data access layer serves as data storage. Furthermore, Internet connectivity facilitates data transmission between the home and the Cloud logic layer.

The smart home architecture proposed by Geneiatakis et al. (2017) encompasses five components: IoT devices, smart hubs, home routers, the Internet and the Cloud. IoT devices connect to smart hubs using the Internet and wireless communication protocols, allowing the occupant to interact with the IoT devices either directly via the smart hub using the smart hub's device management services or through Cloud services.

According to Khawla and Mazri (2018), a smart home architecture comprises five main components: connected objects, a smart home gateway, the Cloud, applications, and communication protocols. The components communicate using communication protocols that vary depending on the layer of the Open Systems Interconnection (OSI) model. Their findings emphasise the importance of the Cloud, illustrating the difference in Cloud deployment approaches, including private Cloud, hybrid Cloud, public Cloud and community Cloud.

Tao et al. (2018) illustrated a multi-layered smart home architecture consisting of IoT devices, communication technology, middleware and the Cloud. IoT devices, such as sensors, actuators, controllers, and smartphones, communicate through communication protocols such as Bluetooth, Zigbee, WiFi, or middleware. The authors do not explicitly define middleware; instead, they use this term to refer to technology that supports communication and integration between applications and the Cloud. The final component is the Cloud, which the authors classify as private, public or a combination.

Mocrii et al. (2018) propose a general architecture for a smart home that utilises the Cloud to collect and process data travelling within the smart home architecture. Their architecture contains four components: the edge, communication protocols, the gateway and the Cloud. The edge contains a collection of end devices such as appliances, sensors and actuators. The gateway connects the end-devices to the Cloud through communication protocols, enabling communication within the smart home and the outside environment. After the gateway, the data arrives at the Cloud, which stores and processes data. The Cloud allows for integrations with other third-party services, such as applications for mobile phones that offer device management services.

Shouran et al. (2019) discuss a smart home architecture using three network layers: perception, network and application. The perception layer consists of smart devices. The network layer refers to the collection of the home network, the home gateway and the Cloud. Furthermore, the application layer facilitates services to users through platform services.

This comparative analysis of smart home architectures reveals the lack of standardisation in smart home architectures. This may lead to a misunderstanding of smart home architectures, which is crucial to address, as some components were used interchangeably, but they inherently have distinct meanings. For instance, various terms are used in literature to indicate the presence of wired or wireless devices within the smart home architecture, such as “connected objects”, “IoT devices”, “end-devices”, and “smart home devices”.

Terms such as “IoT devices” and “smart devices” are used interchangeably in the literature, but it is important to distinguish these terms as their definitions vary. IoT devices refer to electronic devices with internet access, whereas smart devices are context-aware and are capable of performing autonomous activities while facilitating data transmission between other devices (Silverio-Fernández et al., 2018). For instance, a smoke detector with internet connectivity is considered an IoT device. On the other hand, robot vacuums, which identify various floor conditions, adjust their cleaning parameters accordingly and use internet connectivity to notify the owner when the cleaning is finished, are classified as smart devices.

According to Mocrii et al. (2018), IoT devices are not considered intelligent by themselves. However, the cumulative data gathered from IoT devices, combined with autonomous actions being performed within a home, contribute to the definition of a smart home environment. Based on this observation, this study defines the components of a smart home architecture as follows:

- *IoT and/or smart devices:* Refers to devices such as sensors, actuators or appliances that collect and share data with other devices.
- *Smart home gateway:* Refers to a system that serves as a bridge between the smart home environment and the Cloud. It facilitates data transfer between the smart home and the Cloud, enabling access to IoT device management services. In addition, it acts as a central hub with internet connectivity and connectivity for IoT smart devices.
- *Communication protocols:* Refer to protocols that facilitate communication within the smart home and the Cloud.
- *Cloud:* Refers to servers storing and processing data. It can be private, public, hybrid or community-based.
- *Applications:* Refers to software applications designed to deliver services to the users using the Cloud.

2.4 Smart home network

This subchapter delves into the existing literature regarding critical components of a smart home network, including the network topology, network structure and communication protocols. The findings of the literature shed light on how communication between various smart home components is facilitated.

2.4.1 Smart home network topology

Literature reveals four architectural topologies that are commonly used when creating a smart home network (Mansour et al., 2023; Marksteiner et al., 2017; Stojkoska & Trivodaliev, 2017; Tournier et al., 2021):

- *Star topology*. A single central home gateway manages the communication between nodes and networks.
- *Tree topology*. The communication in the tree topology is hierarchical, with a single home gateway at the top connected to several nodes underneath.
- *Mesh topology*. Facilitates direct connection between routing and neighbouring nodes, allowing for non-central communication.
- *Peer to Peer*. Several connections between two nodes throughout the network allow nodes to communicate without a central home gateway.

2.4.2 Smart home network structure

Depending on the connectivity requirements and operational range, various network structures can be applied in smart homes. Local Area Networks (LAN) and Personal Area Networks (PAN) are commonly observed as smart home network structures as they facilitate the connection of a limited number of devices while efficiently maintaining network resources (Hafeez et al., 2014; Holguin & Errapotu, 2023; Tournier et al., 2021). However, with the increase of wireless IoT and smart devices, the network structures necessitated versatility, leading to the popularity of Wireless PAN (WPAN) and Wireless LAN (WLAN) used in smart home networks. While WPAN supports a limited number of devices within a range of 10 to 50 meters, it enhances energy efficiency and excels in providing short-distance wireless communication (Chen et al., 2021; Kim, 2022; Orfanos et al., 2023; Tournier et al., 2021).

Due to the short distance, WPAN supports flexible topologies such as the star or mesh topology (Holguin & Errapotu, 2023). On the other hand, WLAN covers a broader range of 50 to 100 meters, which accommodates more devices but at the cost of higher resource consumption than WPAN (Chen et al., 2021; Tournier et al., 2021).

2.4.3 Smart home network communication protocols

Network communication protocols facilitate data transmission in smart home networks. Communication protocols possess different functionalities and operate on different layers of the IoT protocols stack (Mansour et al., 2023). Figure 2 illustrates the IoT protocol stack layers alongside their corresponding smart home network communication protocols. Table 6 compares 14 studies to highlight common smart home network communication protocols. The IoT protocol stack framework facilitates the explanation of how data is transmitted within a smart home network. The IoT protocol stack consists of five layers: physical, data-link, network, transport and application layer (Mansour et al., 2023; Marksteiner et al., 2017; Tournier et al., 2021).

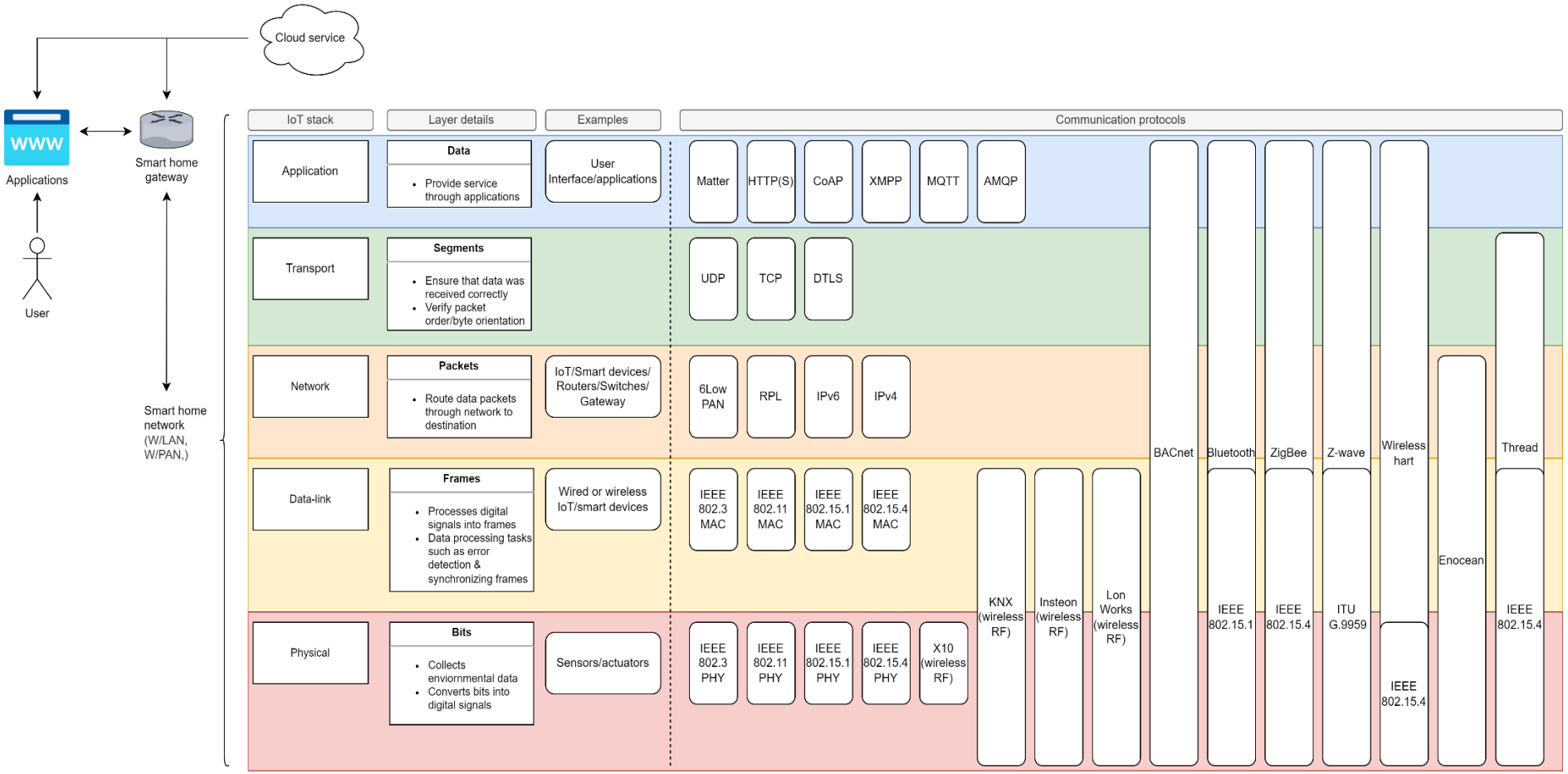

Figure 2. Smart home network communication protocols using the IoT protocol stack framework.

Table 6. Comparing 14 studies regarding common smart home communication protocols.

	Wired protocol						Wireless protocol										Dual protocol		
	Ethernet	BACnet	KNX	X10	Insteon	LonWorks	Wi-Fi	Bluetooth	Zigbee	Z-wave	6LOWPAN	Enocean	Thread	Wirelesshart	Matter	KNX RF	X10 RF	Insteon RF	LonWorks RF
Grabowski and Dziwoki (2009)			x	x			x	x	x										
Lee et al. (2014)							x	x	x		x								
Mendes et al. (2015)	x		x	x	x	x	x	x	x	x		x		x		x	x	x	x
Gunawan et al. (2017)			x	x	x		x	x	x	x						x	x		
Marksteiner et al. (2017)			x	x		x			x	x	x	x	x			x			
Stojkoska and Trivodaliev (2017)		x		x	x		x	x	x	x	x	x					x	x	
Khawla and Mazri (2018)							x	x	x		x								
Mocrii et al. (2018)	x		x	x	x		x	x	x	x	x					x	x	x	
Anwar et al. (2020)							x	x	x		x			x					

	Wired protocol						Wireless protocol										Dual protocol		
	Ethernet	BACnet	KNX	X10	Insteon	LonWorks	Wi-Fi	Bluetooth	Zigbee	Z-wave	6LOWPAN	Enocean	Thread	Wirelesshart	Matter	KNX RF	X10 RF	Insteon RF	LonWorks RF
Tournier et al. (2021)								x	x	x	x								
Gerodimos et al. (2023)							x	x	x	x	x								
Holguin and Errapotu (2023)								x	x	x			x		x				
Mansour et al. (2023)	x		x	x			x	x	x	x	x								
Orfanos et al. (2023)	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x	x	x

The physical layer contains hardware that utilises sensors to collect data and transform it into electronic signals known as bits. The subsequent data-link layer processes the electronic signal into frames. This layer ensures the data processing is carried out correctly by performing error and collision detection tasks. The Physical and Media Access Control standards of IEEE, such as Ethernet (802.3), WLAN (802.11), Bluetooth (802.15.1), and Low power/data rate wireless communication (802.15.4) reside on both the physical and data-link layer (Mendes et al., 2015). However, communication protocols that do not conform to the IEEE standard and are also present at this layer, such as X10, KNX, Insteon, LonWorks, BACnet, Enocean and Z-wave, use the ITU G.9959 specification. The following network layer facilitates routing data packets over the network to their destination. Common network layer protocols include IPv4, IPv6, Ipv6 Routing Protocol For Low Power Lossy Networks (RPL) and Ipv6 Over Low Power Wireless Personal Area Networks (6LowPAN) (Gerodimos et al., 2023; Holguin & Errapotu, 2023; Khawla & Mazri, 2018; Lee et al., 2014; Mansour et al., 2023; Marksteiner et al., 2017; Orfanos et al., 2023; Stojkoska & Trivodaliev, 2017; Tournier et al., 2021). Next, the transport layer assesses the packet order and byte orientation to verify that the data, now referred to as segments, were received correctly. UDP, TCP, and DTLS are common protocols used in this layer (Gerodimos et al., 2023; Khawla & Mazri, 2018; Lee et al., 2014; Mansour et al., 2023; Marksteiner et al., 2017; Tournier et al., 2021). Finally, the application layer delivers services to users via software applications. Common communication protocols on the application layer include HyperText Transfer Protocol (HTTP), Constrained Application Protocol (CoAP), eXtensible Messaging and Presence Protocol (XMPP), Message Queue Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP) and recently also Matter (Gerodimos et al., 2023; Holguin & Errapotu, 2023; Khawla & Mazri; Lee et al., 2014; Mansour et al., 2023; Mocrii et al., 2018; Tournier et al., 2021).

The analysis of smart home network communication protocols, as presented in Table 6, reveals that the protocols can be categorised into wired, wireless and hybrid/dual (Mocrii et al., 2018; Orfanos et al., 2023). Although wired protocols like Ethernet formed the original communication standard in smart home networks, wireless protocols emerged and dominated in popularity due to their cost-effectiveness and efficient connectivity (Orfanos et al., 2023).

Furthermore, the literature showed that some wired smart home network communication protocols like KNX, X10, Intseon, and LonWorks adjusted to the growing popularity of wireless smart home network communication protocols by incorporating radio frequency to facilitate wireless communication. As a result, these smart home network communication protocols fall under the hybrid/dual communication category. The dataset used in this study contains various protocols with TCP, UDP, and ICMP, accounting for most protocol types observed in the network data.

2.5 Smart home network attacks

This subchapter presents a literature review on cyber-attacks targeting smart home networks, examining their impact and prevalent attack types. Smart home networks have become a significant point of interest for attackers, as they may reveal occupants' personal information and habits (Denning et al., 2013). The increasing number of vulnerabilities in smart home networks is primarily due to the rapid development of insecure IoT and smart devices, the popularity of wireless smart home network communication protocols and the absence of security professionals in a smart home environment (Denning et al., 2013). As a result, Victor et al. (2023) reported that cyber-attacks targeting smart home networks have sharply increased over recent years.

According to Denning et al. (2013), attacks on smart home networks can negatively impact the occupant's privacy, financial stability, emotional well-being and physical safety. Compromised smart home networks can alter critical data attributes, including the confidentiality, integrity, availability, authorisation, and authentication of data (Hammi et al., 2022; Geneiatakis et al., 2017; Khawla & Mazri, 2018; Sivapriyan et al., 2021). Consequently, attackers exploit private and sensitive information for illicit purposes such as voyeurism, espionage, or blackmailing (Denning et al., 2013). Furthermore, financial repercussions can arise when a smart home network is compromised. For instance, when smart devices that regulate daily utilities are compromised, like lights or thermostats, the occupant may experience increased energy consumption (Denning et al., 2013). Moreover, Denning et al. (2013) suggested that cyber-attacks targeting smart home networks can psychologically impact the occupant, causing emotional distress and disrupting personal relationships. Additionally, compromised smart home networks pose a risk to the occupant's physical safety. Access and control to smart security systems such as smart cameras, smart door locks, and smart motion sensors can facilitate criminal activities such as burglary (Denning et al., 2013; Hammi et al., 2022).

Furthermore, malfunctioning smart devices due to a compromised smart home network can create hazardous scenarios, such as unattended smart ovens overheating. Finally, the unavailability of smart healthcare devices, like ECG and blood pressure monitoring systems, can pose health risks to occupants.

Literature analysis reveals two types of cyber-attacks that smart home networks experience: passive and active attacks (Geneiatakis et al., 2017; Khawla & Mazri, 2018). Whereas passive cyber-attacks aim to gather information without altering the network's data, active cyber-attacks, on the other hand, focus on altering network data to intrude or disrupt the network (Khawla & Mazri, 2018).

This study focuses on active cyber-attacks, as these attacks can be detected by observing the data, due to the adversary modifying data within the network. Attackers commonly deploy resource-centric cyber-attacks due to limited resources within smart home networks, such as constrained energy, memory or bandwidth (Çakir et al., 2020). According to Çakir et al. (2020), active resource-related cyber-attacks aimed at smart home networks can be divided into Denial of Service (DoS)/Distributed DoS attacks (DDoS), routing attacks, and other attacks. Figure 3 illustrates the prevalent cyber-attack types observed in smart home networks. It is based on a comparison of 16 studies concerning smart home network cyber-attacks, which is presented in Table 7.

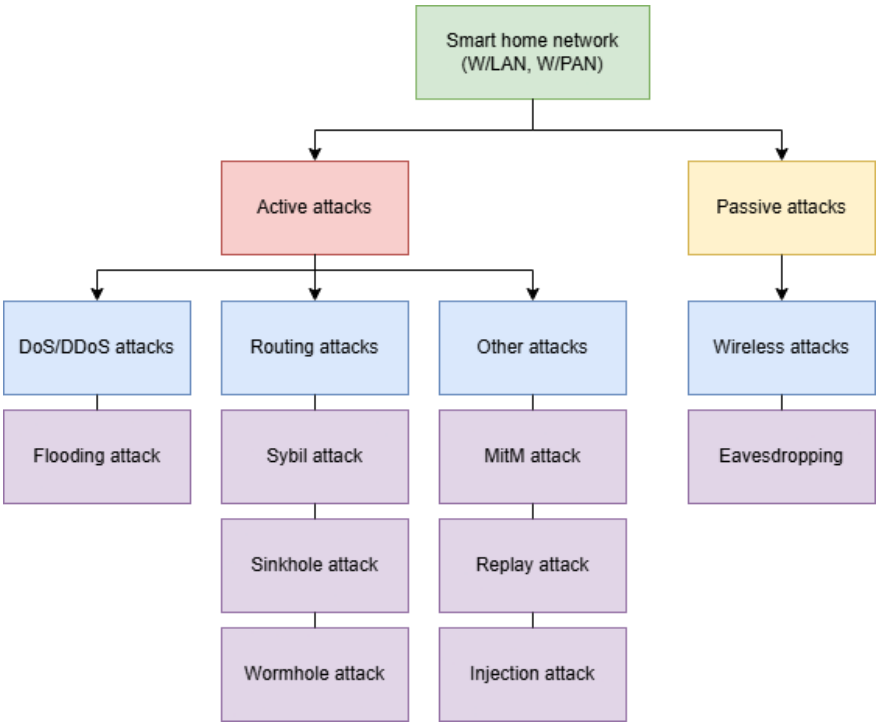

Figure 3. Overview of common smart home network cyber-attacks.

Table 7. Comparing 16 studies regarding active cyber-attacks observed in smart home networks.

		Attack type								
		DoS	DDoS	Sybil attack	Sinkhole attack	Wormhole attack	RPL attack	MITM attack	Replay attack	Injection attack
Authors	Geneiatakis et al. (2017)	x	x						x	
	Saxena et al. (2017)	x		x	x	x		x	x	x
	Ali & Awad (2018)	x		x	x	x		x	x	
	Gai et al. (2018)	x	x						x	x
	Kamel & Hegazi (2018)	x		x	x	x		x		x
	Khawla & Mazri (2018)	x					x	x	x	
	Anthi et al. (2019)	x	x	x	x	x	x	x	x	
	Mohanta et al. (2020)	x	x	x		x	x	x		
	Gupta & Sharma (2021)	x	x	x	x	x		x		
	HaddadPajouh et al. (2021)	x	x	x	x		x		x	
	Sivapriyan et al. (2021)	x	x							
	Touqeer et al. (2021)	x					x	x		x
	Hammi et al. (2022)	x	x	x				x	x	x
	Araya & Rifà-Pous (2023)	x	x		x			x	x	x
	Liu et al. (2023)		x						x	
	Victor et al. (2023)		x							x

2.5.1 DoS/DDoS

Observing Table 7 reveals that DoS and DDoS are most prevalent in smart home networks. DoS attacks aim to disrupt a target's service by exhausting network resources by transmitting large data packets (Araya & Rifà-Pous, 2023; Hammi et al., 2022; Kamel & Hegazi, 2018). Similarly, DDoS attacks share the same objective as DoS attacks but instead employ multiple nodes spread across various locations to overwhelm the target (Araya & Rifà-Pous, 2023). DoS and DDoS attacks can occur by flooding the target or exploiting a network protocol flaw (Hammi et al., 2022). Flooding assaults such as TCP, UDP or Hello flood attacks tend to abuse lightweight routing protocols such as RPL to overwhelm the target (Anthi et al., 2019; Çakir et al., 2020; Touqeer et al., 2021). DoS and DDoS attacks can be considered collective anomalies; while individual data packets may appear normal, the collective transmission behaviour indicates a significant deviation from the expected normal network behaviour. The impact of unavailable smart home networks and components differs depending on the type of IoT or smart devices in the network (Hammi et al., 2022). Whilst some smart home occupants may experience minor inconvenience, others may be in a catastrophic situation when smart healthcare devices become unavailable. Due to the relevance of this attack, this study aims to detect DDoS attacks using an ML-based ANIDS in smart home networks.

2.5.2 Routing attacks

Routing attacks represent the second category of active attacks that smart home networks encounter. Routing attacks target routing protocols, particularly those with constrained resource capacity such as RPL, to gain, alter or remove routing information (Anthi et al., 2019; Gupta & Sharma, 2021). The most common routing attacks observed in the literature included Sybil, sinkhole and wormhole attacks. Sybil attacks use Sybil nodes that forge multiple identities to manipulate the network and perform malicious activities (Mishra et al., 2019). The sinkhole attack takes a different approach by compromising nodes or creating a malicious node in the network that deceives other nodes into connecting with the malicious node to collect network traffic (Saxena et al., 2017). The wormhole attack deceives other nodes in the network traffic by combining two malicious nodes and creating an efficient tunnel that provides significantly better communication resources than other network nodes (Saxena et al., 2017). Traffic passing through this tunnel enables attackers to intercept, redirect or delay network traffic (Mohanta et al., 2020; Saxena et al., 2017).

2.5.3 Other attacks

This category refers to any other active cyber-attack that does not belong in either DoS/DDoS or routing attacks. The first commonly observed cyber-attack in this category is the Man-in-the-Middle (MitM) attack. MitM attacks aim to intercept network communication between a sending and receiving node by placing itself in between their connection while masking its presence in order to analyse or alter the traffic unnoticeably (Araya & Rifà-Pous, 2023; Saxena et al., 2017). Similar to MitM attacks, replay attacks intercept network traffic between two nodes to facilitate unauthorised access. In replay attacks, the attackers impersonate themselves as the original sending node by resubmitting network packets that were previously intercepted (Araya & Rifà-Pous, 2023; Saxena et al., 2017). Replay attacks typically exploit the lack of authentication measures in smart home network protocols to gain access to the network, such as with 6LoWPAN (Khawla & Mazri, 2018). The third type of attack is injection attacks. Injection attacks modify network packets by injecting deceptive data to compromise a target or force a target to become unavailable (Araya & Rifà-Pous, 2023; Hammi et al., 2022). For instance, attackers may inject malicious data into a network packet, causing the target node to malfunction and potentially granting the attacker control of the node or network (Araya & Rifà-Pous, 2023; Kamel & Hegazi, 2018).

2.6 Unsupervised Ensemble classifiers

This subchapter reviews the literature on supervised and unsupervised EL classifiers for anomaly detection, emphasising three effective unsupervised classifiers. Furthermore, the performance evaluation metrics for the unsupervised classifiers are examined, and prevalent datasets employed for ML-based ANIDS in smart home environments are discussed.

Existing literature highlights the effectiveness of employing traditional EL methods to identify cyber-attack anomalies in smart home environments (Araya & Rifà-Pous, 2023). Subsequently, Araya and Rifà-Pous (2023) observed that most well-performing ensembles utilise supervised EL, suggesting that supervised ML classifiers effectively detect cyber-attack anomalies within a smart home network. While supervised EL produces promising results, the performance of the supervised ML classifier is tied to a controlled environment and does not represent the results for a dynamic environment.

Labelled data is typically absent in dynamic environments such as smart home networks, which supervised ML classifiers require for training and evaluation activities (Araya & Rifà-Pous, 2023; Das et al., 2020). Subsequently, academic research regarding Unsupervised Ensemble (UE) gained attention to investigate the feasibility and effectiveness of combining unsupervised classifiers trained and tested on unlabelled data for anomaly detection (Das et al., 2020). Inherent to UE is that unsupervised classifiers utilise unlabelled data to detect anomalies, suggesting no human intervention is required to label the data. In addition, Das et al. (2024) observed that UE can identify unknown and evolving attacks by employing novelty-based or outlier detection methods, which can discover unusual patterns in data (Das et al., 2024). Novelty-based and outlier-based detection methods use clustering to group inlier data, creating a boundary to identify outlier data (Das et al., 2024). The two methods differ in training, as novelty-based detection uses training data that does not contain anomalies, and outlier-based detection uses a combination of anomalous and normal data in its training data.

Both Agyemang (2024) and Das et al. (2020) observed that the unsupervised classifiers One Class Support Vector Machine (OCSVM), Local Outlier Factor (LOF), and Isolation Forest (iForest) showed robust performance in identifying DDoS anomalies. While numerous unsupervised classifiers exist, this study employs OCSVM, LOF, and iForest to establish the UE due to their efficiency and performance demonstrated in prior research, forming the first core component of SHARP. SHARP incorporates OCSVM and LOF using the novelty-based method and applies iForest to integrate outlier-based detection.

2.6.1 OCSVM

OCSVM identifies outliers by learning the characteristics of normal data, thereby establishing a boundary which creates a barrier between normal and anomalous data (Agyemang, 2024). Where data within the boundary is considered normal, data outside the boundaries is identified as anomalous. OSCVM uses a pre-determined kernel matrix to observe the distance between data, using various kernels such as polynomial, linear or Radial Basis Functions (Agyemang, 2024). The OCSVM training process creates a boundary using only one data type, either normal or anomalous (Das et al., 2020).

2.6.2 LOF

LOF identifies anomalies using the local density and observing the difference between neighbouring data points (Agyemang, 2024). The K-distance is used to define the area around a single data point. This area, also known as the neighbourhood, is then used to gain insight into the reachability of one data point to another. The neighbourhood's density and reachability of data points are then used to establish a final score (LOF score), which measures the isolation between a data point and its neighbouring data points (Das et al., 2020; Agyemang, 2024). LOF scores closer to 1 show similar density results and imply normal data, whereas LOF scores exceeding 1 indicate anomalous data.

2.6.3 iForest

iForest detects anomalies by leveraging a hierarchical tree mechanism, where the path length from the starting node to the final node indicates whether the data is anomalous (Das et al., 2020). iForest utilises a tree structure with the root node at the top and branches leading downwards by randomly splitting the data as long as it is within a predefined maximum feature space. Analysing the frequency and depth of the splits is the key activity that iForest uses to identify anomalous data. Paths with fewer splits imply anomalous data due to anomalous data typically occurring less frequently than normal data (Das et al., 2020; Agyemang, 2024).

2.6.4 Performance evaluation metrics

Performance evaluation metrics can be applied to measure the effectiveness of the three unsupervised classifiers that identify DDoS anomalies. (Araya & Rifà-Pous, 2023; Das et al., 2024). The confusion matrix, consisting of four categories, is used to compute equations that form the foundation of different performance metrics (Abdulla & Hasoun, 2022; Das et al., 2024). The four categories include True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN).

Table 8. Confusion matrix.

	Predicted normal data	**Predicted anomalous data**
Actual normal data	TN – Correctly classifies normal data as normal.	FP – Incorrectly classifies normal data as anomalous.
Actual anomalous data	FN – Incorrectly classifies anomalous data as normal.	TP – Correctly classifies anomalous data as anomalous.

Table 8 provides an overview of the categories within the confusion matrix. Literature analysis reveals the prevalence of computing performance metrics, including the accuracy, precision, recall, and F1-score, when assessing the performance of unsupervised classifiers (Agyemang, 2024; Araya & Rifà-Pous, 2023; Das et al., 2024; Liao et al., 2021; Singh & Kane, 2022). Furthermore, error metrics such as False Positive Rate (FPR), False Negative Rate (FNR), and True Negative Rate (TNR) provide insight into the competence of unsupervised classifiers in distinguishing normal and anomalous data.

Table 9. Overview of seven performance evaluation metric equations.

Performance metric	Formula	Description
Accuracy	$\frac{(TP+TN)}{(TP+FP+FN+TN)}$	Measures DDoS attacks that are accurately classified out of the whole dataset.
Precision	$\frac{TP}{(FP+TP)}$	Measures how frequently DDoS attacks are accurately classified.
Recall	$\frac{TP}{(FN+TP)}$	Measures the number of DDoS attacks that are classified accurately.
F1-score	$2 * \frac{(Precision * recall)}{(Precision + recall)}$	Measures the balance of precision and recall.
FPR	$\frac{FP}{(FP+TN)}$	Incorrect prediction of benign data classified as DDoS attack.
FNR	$\frac{FN}{(TP+FN)}$	Incorrect prediction of DDoS attack classified as benign data.
TNR	$\frac{TN}{(TN+FP)}$	Correct prediction of DDoS attack, also known as specificity.

Table 9 outlines the mathematical equations for each performance evaluation metric using the four categories of the confusion matrix (Das et al., 2024; Liao et al., 2021).

Comparing the differences in performance metrics across the three unsupervised classifiers can be achieved by utilising statistical approaches such as the one-way Analysis of Variance (ANOVA) (Khampuengson & Wang, 2022). The one-way ANOVA helps to identify whether a difference in the performance metrics among three unsupervised classifiers is present. A post-hoc analysis test, such as Tukey's Honest Significant Difference (HSD), can be applied if a significant difference is observed. Tukey's HSD test provides insights into which unsupervised classifiers differ from each other (Khampuengson & Wang, 2022).

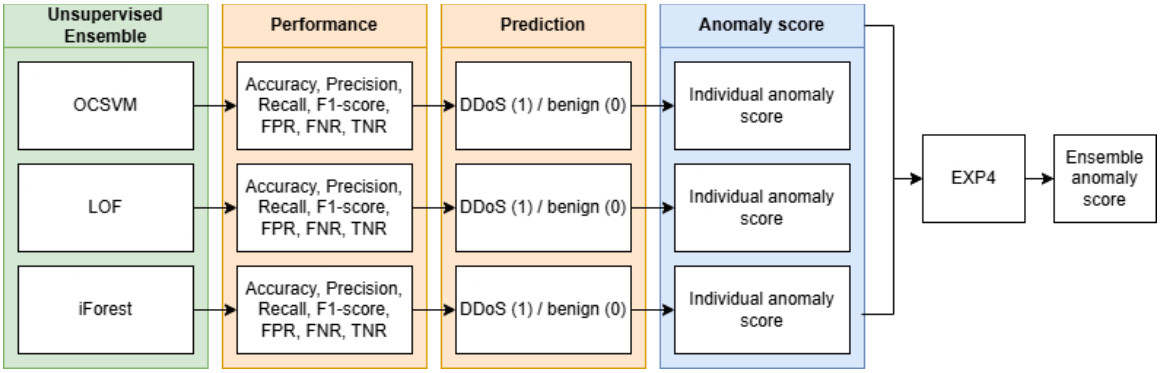

Figure 4. UE model and their performance metrics.

Figure 4 illustrates SHARP's UE model, which consists of three unsupervised classifiers: OCSVM, LOF, and iForest. The performance of every unsupervised classifier is assessed using the seven performance metrics: accuracy, precision, recall, F1-score, TNR, FPR, and FNR. These performance metrics are also utilised to establish the ensemble anomaly score. The data is labelled as DDoS (1) or benign (0), depending on the unsupervised classifier's performance. The performance and prediction values are combined to generate an anomaly score for each unsupervised classifier. After applying the EXP4 algorithm, the individual anomaly scores are combined into a final ensemble anomaly score.

2.6.5 Smart home network datasets

To train and evaluate SHARP's UE model, datasets that contain DDoS data and normal smart home network traffic data are required. The comprehensive literature analysis of Araya and Rifà-Pous (2023) reveals that academic research concerning the detection of cyber-attacks in smart home environments uses datasets originating from software simulations, a controlled laboratory experiment, or an actual smart home setting.

Prevalent public datasets used for the detection of cyber-attacks in a smart home network setting in academic literature include IoT-23, N-BaIoT, CSE-CIC-IDS2018, UNSW-NB15, NSL-KDD and CICIDS2017 (Araya & Rifà-Pous, 2023; Das et al., 2024; Saha et al., 2022). Araya and Rifà-Pous (2023) highlighted the importance of datasets incorporating diverse smart devices to enhance cyber-attack anomaly detection in smart home environments. The authors argue that some datasets used for anomaly detection in smart home environments lack the complexity of smart home network traffic, as the data contains conventional LAN traffic from devices such as servers, limiting the applicability to smart home environments, as observed with the CSE-CIC-IDS2018 dataset.

Therefore, the authors emphasise the necessity of using network data containing normal and anomalous traffic of diverse smart devices to reflect a realistic smart home environment.

This study aims to identify DDoS anomalies within a smart home network context. Therefore, it is essential to assess SHARP's UE model under conditions that reflect a realistic smart home environment. To achieve this objective, this study employs the CIC IoT Dataset 2023, developed by the University of New Brunswick Centre for Cybersecurity. The CIC IoT Dataset 2023 was specifically developed to address complexities and security challenges within an IoT network. The dataset contains real-time captured traffic data from a realistic smart home network, encapsulating a heterogeneous IoT topology using 105 IoT devices alongside 33 different cyber-attacks, including DDoS attacks. Moreover, the dataset captured traffic through various communication protocols, such as TCP and UDP, and specific smart home network communication protocols, such as Zigbee and Z-wave.

2.7 Reinforcement Learning with EXP4

RL has gained significant relevance in the academic literature across various domains to perform tasks that involve dynamic learning environments. The core principle of RL enables learning agents to iteratively optimise themselves by assessing the state of the environment and selecting the optimal action (Mahesh, 2020). Learning agents receive feedback through positive or negative rewards depending on the outcome of the selected action, allowing for a dynamic decision-making system (Araya & Rifà-Pous, 2023; Mahesh, 2020; Tariq et al., 2022). Since smart home environments contain dynamic network traffic, anomaly detection systems must facilitate and respond to dynamic changes (Araya & Rifà-Pous, 2023). Therefore, Araya and Rifà-Pous (2023) proposed to explore the integration of RL methodologies to enhance anomaly detection in smart home environments. This subchapter explores the potential of RL in an ML-based ANIDS to identify DDoS anomalies.

2.7.1 RL with Multi-Arm Bandit (MAB)

According to Araya and Rifà-Pous (2023) and Tariq et al. (2022), incorporating RL gives the UE the ability to dynamically adjust itself by performing iterative cycles, allowing for flexibility in the decision-making process of the UE and enhancing the final ensemble anomaly score over time.

Establishing an ML-based ANIDS to detect DDoS anomalies in a smart home network necessitates comprehending the underlying issue. Identifying DDoS anomalies within a smart home network can be conceptualised as a Multi-Armed Bandit problem (MAB). In MAB problems, a learning agent must discover which action yields the most significant reward when presented with several actions (often called arms). The learning agent selects and executes an action based on a probability distribution and receives the corresponding reward for the selected action (Lattimore & Szepesvári, 2020).

The SHARP design utilises the MAB approach to facilitate a dynamic decision-making process by creating a weight-based decision mechanism. SHARP's learning agent takes the individual unsupervised classifier anomaly scores into account to select and carry out an action that it deems to be optimal. After carrying out the action, the status of the environment is observed, and the learning agent receives a corresponding reward, which informs whether the selected action was deemed beneficial.

A vital challenge that learning agents face with MAB problems is that they must maintain a balance between selecting the action that offers the best reward given the current state (exploitation) and experimenting with other actions to discover the association between the actions and their rewards (exploration) (Lattimore & Szepesvári, 2020). Maintaining the balance between the exploration-exploitation trade-off can be achieved through the use of the Exponential weight for Exploration and Exploitation (EXP4) algorithm (Tariq et al., 2022). Tariq et al. (2022) demonstrated how EXP4 can optimise a single unsupervised classifier by integrating an iterative weight-based decision-making system. There is a lack of literature exploring the application of EXP4, particularly in combination with UE models. Therefore, this study explores the integration of UE with EXP4 to gain insight into whether EXP4 can enhance a UE model consisting of multiple unsupervised classifiers. This study aims to create a dynamic weight-based decision system that combines the strengths of the three individual unsupervised classifiers in the UE model to identify DDoS attacks. This approach necessitates comprehension of the fundamentals of the EXP4 algorithm, which is elaborated in the following section.

2.7.2 EXP4 fundamentals

EXP4 is an algorithm that solves MAB problems by introducing the concept of "experts". Experts are systems offering advice on the exploration-exploitation trade-off to select the action that yields the highest reward (Tariq et al., 2022). A collective decision-making system is established by combining the advice of multiple experts, allowing the system to optimise the individual experts in a dynamic environment. To evaluate the expert's performance, "weights" are assigned to every expert to indicate the importance of the expert's advice. Weights take the form of a numerical value. They can be viewed as an accuracy metric that is initially equal among every expert but is updated with new values after every iteration, depending on whether the expert provided valuable advice. Experts with a high weight value imply that the learning agent is confident that the expert provides accurate advice. In contrast, experts with lower weight values suggest that their advice is less significant during decision-making. Utilising this concept allows for an adaptive and continuous optimising decision-making process. Based on the information from Lattimore and Szepesvári (2020) and Tariq et al. (2022), the fundamental elements and their equations are discussed through a 7-step approach.

1. Gather information and initialize variables.

Table 10. Initialisation of EXP4 variables.

Parameter	**Description**
i	Expert.
N	Total number of experts.
K	Total number of actions.
A	Sum of actions $a_i(t)$.
$w_i(t)$	Expert weight. Assign initial weight value $w_i(0) = 1$ for every expert i.
$W(t) = \sum_{i=1}^{N} w_i(t)$	Sum of expert weight.
$\gamma \in [0,1]$	Egalitarian factor; where a high value increases randomness emphasizing exploration.

Before delving into the equations of EXP4, it is recommended to identify and assign variables as presented in Table 10.

2. Obtain advice vector for every expert i.

The expert's advice vector, noted as $e_i(t)$, represents its advice on every action in the form of a vector. The advice of an action can be determined through various methods. SHARP determines the advice vector using the unsupervised classifiers' performance metrics and compares their results to a predefined threshold.

3. Calculate the probability value $p_i(a)$ for every action based on expert's advice.

$$p_i(a) = (1-\gamma)\sum_{i=1}^{N}\left(\frac{w_i(t) * e_{i,a}(t)}{W(t)}\right) + \frac{\gamma}{K} \quad \text{for } a = \{1, \dots, \mathrm{K}\} \tag{1}$$

Equation 1 calculates the probability of expert i selecting action a as long as the action is within the number of possible actions K. Actions with a higher probability value suggest that the algorithm will exploit these actions to gain high rewards. In contrast, actions with a lower probability value imply that these actions must be further explored. Equation 1 contains three parts: the exploration factor, weighted advice and uniform action probability. The exploration factor $(1-\gamma)$ scales the influence of the expert's advice. A high exploration factor value indicates that the algorithm will exploit the action, whereas a smaller value suggests exploring. The expert's weighted advice is calculated in the fraction's nominator by factoring the expert's weight $w_i(t)$ with the advice vector denoted as $e_{i,a}(t)$. The result is then divided by $W(t)$ in the denominator for normalisation purposes. Finally, to avoid bias towards a certain action, a baseline denoted as $\frac{\gamma}{K}$ is added to the equation.

4. Select an action based on the calculated probabilities.

Based on the calculated probabilities, select one action. Use a reward system to obtain the reward for the selected action. To facilitate the exponential increase and decrease in the new weight calculation, the range of reward values is between -1 and 1, where a negative one indicates inaccurate actions and one is assigned for accurate actions. Rewards are denoted as $r_{at} \in [-1,1]$.

5. Determine estimated reward for selected action.

$$a(t) = \begin{cases} \frac{r_a(t)}{p_a(t)}, for\ a = a(t) \\ 0, for\ a \neq a(t) \end{cases} \qquad for\ \hat{r}_i(t) = \{\hat{r}_1(t), \dots . \hat{r}_K(t)\} \tag{2}$$

Equation 2 computes the estimated reward $\hat{r}_i(t)$ solely for experts recommending the selected action to understand its performance relative to its probability value. The estimated reward is computed by dividing the obtained reward by the action's corresponding probability value.

6. Calculate the contribution of every expert.

$$y_i(t) = e_i(t) * \hat{r}_i(t), \quad for\ every\ i,t\ \epsilon [N] \tag{3}$$

Equation 3 computes the contribution of every expert that shows the correspondence between the expert's advice and the estimated reward for the selected action. The contribution is computed by factoring the expert advice vector value for the selected action with the calculated estimated reward.

7. Update weights for next iteration.

$$w_i(t+1) = w_i(t) * e^{\gamma \frac{y(t)}{K}} \tag{4}$$

The expert's weight is updated for the next iteration through Equation 4. The weight value at the next iteration is exponentially adjusted by factoring the expert's weight at the current iteration with the exponential factor e. The equation considers the exploration-exploitation trade-off by factoring the egalitarian factor γ with the expert's normalised contribution value $\gamma^{\frac{y(t)}{K}}$. Increased weight will be assigned to experts who provide accurate advice, whereas less weight will be assigned to experts who provide less significant advice.

3 SHARP Methodology

This chapter discusses the methodology of SHARP's framework by elaborating on the architectural workflow and exemplifying the application of UL and RL. Furthermore, it provides an illustrative example using fabricated data to demonstrate the practical implementation of the SHARP framework.

3.1 SHARP design

This subchapter elaborates on SHARP's architectural design. SHARP is designed to explore the feasibility of combining UL with RL by incorporating a UE model comprising three unsupervised classifiers with the EXP4 algorithm to perform ML-based ANIDS activities. This study uses the SHARP framework to identify and respond to DDoS attacks in smart home networks. By leveraging the strength of three unsupervised classifiers and simultaneously adjusting the weights, which indicate a degree of importance with EXP4, the decision-making process in SHARP facilitates dynamic adjustments and the use of unlabelled data.

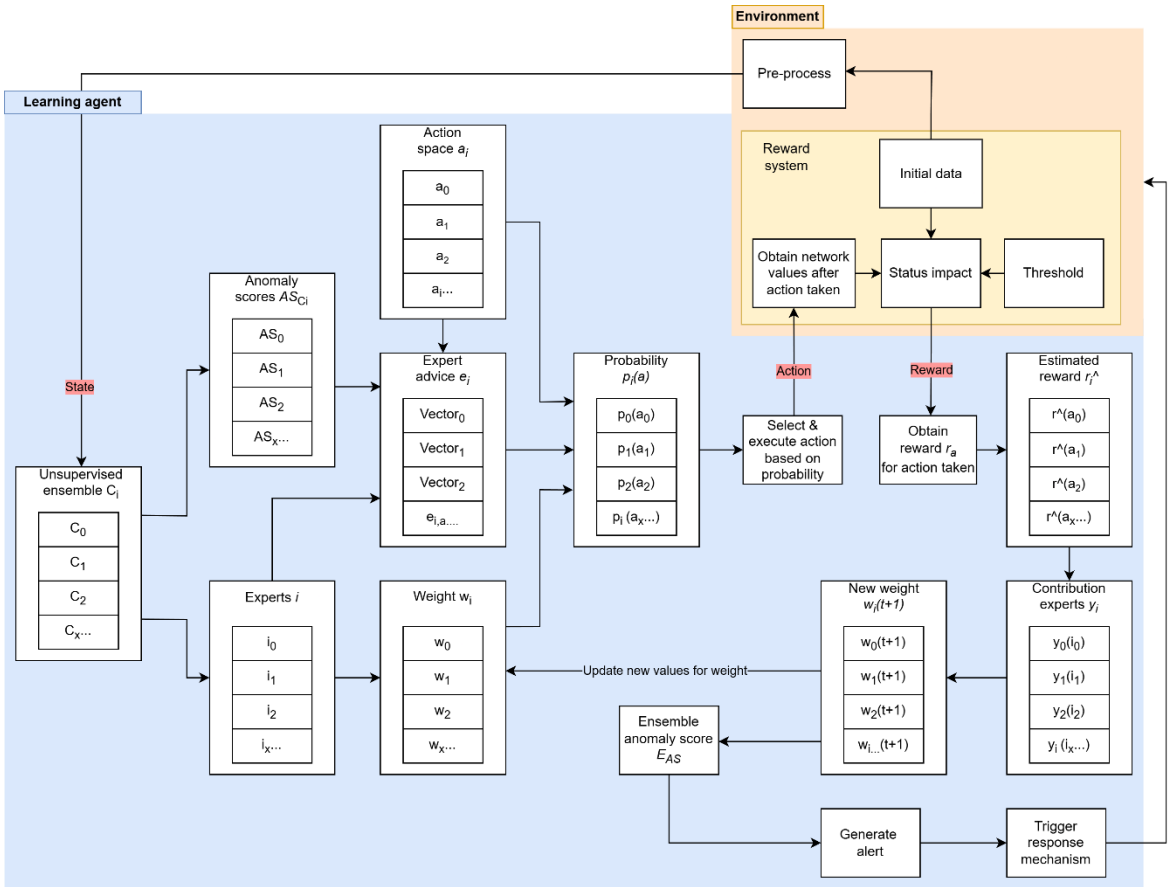

Figure 5. SHARP architecture.

Figure 5 illustrates SHARP's proposed architectural design, which contains two components: the ANIDS learning agent and the environment. The process is initiated in the environment that collects network traffic, pre-processes data, and assigns rewards in EXP4's reward system.

The pre-processed data is sent to the UE, denoted as C_i, which consists of three unsupervised classifiers: OCSVM, LOF and iForest. The unsupervised classifiers produce anomaly scores AS_i that consist of the prediction label and seven performance metrics: accuracy, precision, recall, F1-score, TNR, FPR, and FNR. An expert i and the expert's initial weight value w_i are assigned to each unsupervised classifier. Given SHARP's objective to detect DDoS anomalies, three actions are initialised, in which the learning agent must select one: block IP address, limit traffic rate, and ignore traffic.

The advice vector e_i is the first EXP4 element, which determines the advice for every action by comparing the unsupervised classifier's anomaly scores with predetermined conditions and assigning the corresponding advice vector. The predetermined conditions are presented in Table 11. Following advice vectors, a probability distribution $p_i(a)$ must be calculated for every action. The probability scores are used to facilitate the selection and execution of an action.

Upon executing the action, the environment assigns a reward using its reward system. The reward system assigns rewards by comparing the initial network state and the state after the action has been executed. The difference between the two states is called the "status impact" and is evaluated to understand whether the executed action was beneficial. Key network traffic components for DDoS attacks that can be observed to identify the status impact include the packet count, rate, source IP and port, destination IP and port, protocols and TTL (Ozer & Iskefiyeli, 2017; Sakong & Kim, 2023). Positive rewards indicate an outcome of the executed action, which is deemed beneficial. For instance, if the action was to block network traffic and the status impact shows a significant decrease in network traffic, then a positive reward should be assigned. On the contrary, when the outcome of the executed action results in a similar or worse observed status impact, a negative reward should be recorded. Network monitoring systems can be employed to analyse the network traffic to facilitate observing the status impact. This study implements a conditional logic mechanism (Tables 12, 13 and 14) that assigns rewards by comparing status impact values with predefined conditions. Due to the scope of this study, predetermined status impact values, which include accurate, partially accurate and inaccurate, are used to simulate the status impact observed by the environment.

After obtaining a reward, the estimated reward $\hat{r}_i$ is computed, which is required to compute the contribution y_i of every expert. The contribution reflects the effectiveness of the expert's advice concerning the executed action. Subsequently, the new weight value for each expert is calculated using the expert's contributions, respectively.

The updated expert weight values are used to compute the ensemble anomaly score to leverage the strengths of the unsupervised classifiers, respectively. The ensemble anomaly score includes a majority-voted label and a weighted ensemble anomaly score comprising seven performance metrics. The majority-voted label yields the outcome of the label that was most prevalent across the three unsupervised classifiers. The ensemble anomaly score is computed by combining the weighted averages across the three unsupervised classifiers for the performance metrics accuracy, precision, recall, F1-score, TNR, FPR, and FNR. Based on the ensemble anomaly score outcome, an alert is generated that can trigger a response mechanism that points back to the environment.

3.2 SHARP example

This subchapter discusses the steps for SHARP's framework using simulated numbers to apprehend the architectural flow. Assume three unsupervised learning classifiers are part of the UE model. An expert is attached to each unsupervised classifier. The anomaly scores for each unsupervised classifier consist of a label, where label 1 implies anomalous data and label 0 suggests normal data, as well as the performance metrics. Furthermore, the action space consists of three actions: block IP, limit or ignore traffic.

Assume the following anomaly scores for the three unsupervised classifiers:

- $C_0(AS)$ = [label: 1, accuracy: 99%, precision: 99%, recall: 99%, F1-score: 99%, TNR: 99%, FPR: 1%, FNR:1%]
- $C_1(AS)$ = [label: 1, accuracy: 86%, precision: 86%, recall: 86%, F1-score: 86%, TNR: 86%, FPR: 6%, FNR:6%]
- $C_2(AS)$ = [label: 1, accuracy: 95%, precision: 95%, recall: 95%, F1-score: 95%, TNR: 95%, FPR: 3%, FNR:3%]

Step 1) Initialisation

- $N = 3$
- $K = 3$
- $\gamma = 0.1$
- $w_0(0) = 1, w_1(0) = 1, w_2(0) = 1$
- $W(t) = w_0 + w_1 + w_2 = 1+1+1 = 3$
- $A = [a_0, a_1, a_2]$ where a_0: block, a_1: limit and a_2: ignore

Step 2) Compute expert advice vectors

Using the rules in Table 11, the following expert advice vector values are assigned:

	a_0(block)	a_1 (limit)	a_2 (ignore)
i_0	0.90	0.08	0.02
i_1	0.10	0.70	0.20
i_2	0.70	0.20	0.10

Summary of advice vectors:

$e_0(t) = [0.90, 0.08, 0.02]$

$e_1(t) = [0.10, 0.70, 0.20]$

$e_2(t) = [0.70, 0.20, 0.10]$

Table 11. Action condition rules to assign advice vector values.

Action	Rule	Condition	$e_{i,a}$: Block	$e_{i,a}$: Limit	$e_{i,a}$: Ignore	Description
Block	0	Accuracy ≥ 98% Precision ≥ 99% Recall ≥ 99% F1-score ≥ 99% TNR ≥ 99% FPR ≤ 2% FNR ≤ 2%	0.90	0.08	0.02	High confidence in correctly identifying DDoS due to high performance values.
	1	Accuracy ≥ 95% Precision ≥ 97% Recall ≥ 97% F1-score ≥ 97% TNR ≥ 97% FPR ≤ 5% FNR ≤ 5%	0.80	0.15	0.05	Still high confidence of DDoS anomaly, but metrics are slightly lower. Overall performance is still considered reliable.
	2	Accuracy ≥ 90% Precision ≥ 95% Recall ≥ 95% F1-score ≥ 95% TNR ≥ 95% FPR ≤ 5% FNR ≤ 5%	0.70	0.20	0.10	Minimum values to initiate block action. Overall performance is acceptable but not optimal.
Limit	3	Accuracy ≥ 88% Precision ≥ 92% Recall ≥ 92% F1-score ≥ 92% TNR ≥ 92% 5 ≤ FPR ≤ 10% 5 ≤ FNR ≤ 10%	0.08	0.90	0.02	Moderate performance, further evaluation is needed before deciding to block traffic.
	4	Accuracy ≥ 85% Precision ≥ 90% Recall ≥ 90% F1-score ≥ 90% TNR ≥ 90% 5 ≤ FPR ≤ 10% 5 ≤ FNR ≤ 10%	0.15	0.80	0.05	Moderate performance, lower performance metrics and FPR & FNR are higher possibly resulting in more false positives.
	5	Accuracy ≥ 85% Precision ≥ 88% Recall ≥ 88% F1-score ≥ 88% TNR ≥ 88% 5 ≤ FPR ≤ 10% 5 ≤ FNR ≤ 10%	0.10	0.70	0.20	Moderate performance, lower values indicating minimum threshold for limiting traffic.

Ignore	6	Accuracy ≥ 85% Precision ≥ 85% Recall ≥ 85% F1-score ≥ 85% TNR ≥ 85% FPR ≥ 10% FNR ≥10%	0.10	0.20	0.70	Classifier performance is subpar to decide to actually limit the network traffic.
	7	Accuracy ≥ 80% Precision ≥ 82% Recall ≥ 82% F1-score ≥ 82% TNR ≥ 82% FPR ≥ 10% FNR ≥ 10%	0.05	0.15	0.80	The model is performing worse, suggesting the classifier cannot make reliable predictions.
	8	Accuracy ≤ 80% Precision ≤ 80% Recall ≤ 80% F1-score ≤ 80% TNR ≤ 80% FPR ≥ 10% FNR ≥ 10%	0.02	0.08	0.90	Classifier's overall performance cannot be trusted. Low metrics and high FPR/FNR.

Step 3) Compute the probability distribution over actions

Probability calculation a_0:

$$e_0(t) = 0.90 \Rightarrow p_0(0) = (1 - 0.1)\left(\frac{((1*0.90) + (1*0.10) + (1*0.70)}{3}\right) + \frac{0.1}{3}$$
$$e_1(t) = 0.10$$
$$e_2(t) = 0.70$$
$$= 0.9 * \frac{1.7}{3} + 0.03$$
$$= 0.54$$

Probability calculation a_1:

$$e_0(t) = 0.08 \Rightarrow p_1(1) = (1 - 0.1)\left(\frac{((1*0.08) + (1*0.70) + (1*0.20)}{3}\right) + \frac{0.1}{3}$$
$$e_1(t) = 0.70$$
$$e_2(t) = 0.20$$
$$= 0.9 * \frac{0.98}{3} + 0.03$$
$$= 0.324$$

Probability calculation a_2:

$$e_0(t) = 0.02 \Rightarrow p_2(2) = (1 - 0.1)\left(\frac{((1*0.02) + (1*0.20) + (1*0.10)}{3}\right) + \frac{0.1}{3}$$
$$e_1(t) = 0.20$$
$$e_2(t) = 0.10$$
$$= 0.9 * \frac{0.32}{3} + 0.03$$
$$= 0.126$$

Summary of probabilities for every action:

The probability of action a_0 gives approximately $p_0(0) = 54\%$
The probability of action a_1 gives approximately $p_1(1) = 32.4\%$
The probability of action a_2 gives approximately $p_2(2) = 12.6\%$

Step 4) Select action based on probabilities

Select an action based on the calculated probabilities. An example of pseudo-code that selects an action using the probability distribution is demonstrated as follows:

```
Probabilities = [0.54, 0.324, 0.126]

def select_action(probabilities):
   return np.random.choice(len(probabilities),  p=probabilities)
```

Assume action a_0 is selected. Execute the selected action in the environment.

Step 5) Status impact and reward calculation

The status impact necessitates the observation of the initial status against the status after executing the action. SHARP defines three status impact scenarios- accurate, partially accurate and inaccurate- each holding a specific reward value. This study provides an example of a conditional reward system for each action, as shown in Tables 12, 13, and 14. However, these reward conditions may vary depending on predetermined thresholds. Assume that after executing the action a_0, the network traffic was reduced by 88%, no data from blocking the IP address was further received, and the protocols used in the network were more evenly distributed. This status impact suggests an accurate scenario as observed from the condition in Table 12, thus a reward of +1 is obtained for executing the action a_0.

Table 12. Conditional reward rules for the block traffic action.

Block condition	**Accurate +1**	**Inaccurate +0**
Total packets	Reduced > 85%.	Similar to the initial network value or increased.
Source IP & ports	No data from the blocked IP address.	Data from the blocked IP is still received.
Destination IP & ports	No data from the blocked IP address.	Data from the blocked IP is still received.
Protocol	Distributed 2 or more protocols, no dominant protocol that accounts for 80% of traffic.	The majority of traffic uses a dominant protocol.
TTL	Average range 64-128.	Lower than average value.

Table 13. Conditional reward rules for the limit traffic action.

Limit condition	**Accurate +1**	**Partially accurate +0.5**	**Inaccurate -1**
Total packets	Reduced by 30-50%.	Reduced by 20-30%.	Similar to the initial network value or increased.
Source IP& ports (one source)	Reduced number of source IPs originating from same source to acceptable amount.	Slight reduction in the number of source IPs originating from the same source.	No significant reduction originating from the same source.
Source IP & ports (many sources)	Reduced number of source IP to acceptable number of source IP (100-500 unique IP).	Slight reduced number of source IP to acceptable number of source IP (100-500 unique IP).	No significant reduction originating from the same source.
Destination IP & ports	Similar destination.	Similar destination.	Variation in destination.
Protocol	Distributed 2 or more protocols, no dominant protocol that accounts for 80% of traffic.	Majority of traffic uses a dominant protocol.	Majority of traffic uses a dominant protocol.
TTL	Average range 64-128.	Average range 64-128.	Above or below the average range.

Table 14. Conditional reward rules for the ignoring traffic action.

Ignore condition	**Accurate +1**	**Partially accurate +0.5**	**Partially accurate +0.5**	**Inaccurate -1**	**Inaccurate -1**
Total packets	Similar as initial or 10% deviation.	Reduced by >25%.	Increased by >25%.	Reduced by >30%.	Increased by >30%.
Source IP& ports (one source)	Similar number.	Reduced number of source IPs originating from same source by >25%.	Increased number of source IPs originating from same source by >25%.	Reduced number of source IPs originating from same source by >30%.	Increased number of source IPs originating from same source by >30%.
Source IP & ports (many sources)	Similar number.	Reduced number of source IP by >25%.	Increased number of source IP by >25%.	Reduced number of source IP by >30%.	Increased number of source IP by >30%.
Destination IP & ports	Similar destination.	Similar destination.	Similar destination.	Similar destination.	Similar destination.
Protocol	Distributed 2 or more protocols, no dominant protocol that accounts for 80% of traffic.	Dominant protocol accounts for 60% of the protocols.	Dominant protocol accounts for 60% of the protocols.	Dominant protocol accounts for 70% of the protocols.	Dominant protocol accounts for 70% of the protocols.
TTL	Average range 64-128.	Average range 64-128.	Average range 64-128.	Above or below average range.	Above or below average range.

Summary of rewards for selecting action a_0:

- $r_0 = 1.0 \rightarrow$ because i_0 provided highest advice value on a_0
- $r_1 = 0 \rightarrow$ because i_1 did not provide highest advice value a_0
- $r_2 = 1.0 \rightarrow$ because i_2 provided highest advice value on a_0

Step 6) Determine the estimated reward for the selected action

For $i = 0$ and (a_0): $\hat{r}_0(t) = \frac{1}{0.54} = 1.85$

For $i = 1$ and (a_1): $\hat{r}_1(t) = \frac{0}{0.324} = 0$

For $i = 2$ and (a_2): $\hat{r}_2(t) = \frac{1.0}{0.54} = 1.85$

Step 7) Calculate contribution $y_i(t)$ for every expert $i \in N$

$y_0(t) = e_0(t) * \hat{r}_0(t) = 0.90 * 1.85 = 1.66$

$y_1(t) = e_1(t) * \hat{r}_1(t) = 0.10 * 0 = 0$

$y_2(t) = e_2(t) * \hat{r}_2(t) = 0.70 * 1.85 = 1.29$

Step 8) Calculate the new weight value for the next iteration

Expert 1: $w_0(t+1) = w_0(t) * e^{\gamma \frac{y(t)}{K}} = 1 * e^{\frac{0.1*1.66}{3}} = 1 * e^{0.055} = 1.056$

Expert 2: $w_1(t+1) = w_1(t) * e^{\gamma \frac{y(t)}{K}} = 1 * e^{\frac{0.1*0}{3}} = 1 * e^{0} = 1$

Expert 3: $w_2(t+1) = w_2(t) * e^{\gamma \frac{y(t)}{K}} = 1 * e^{\frac{0.1*1.29}{3}} = 1 * e^{0.043} = 1.044$

Step 9) Calculate ensemble anomaly score

Two out of three unsupervised classifiers predicted label 1; therefore, with majority voting, the final label gives us 1. Next, the performance metrics for the ensemble anomaly score are computed. First, the new weight value is normalised by dividing the new weight value by the sum of all weight values, as shown in Equation 5.

$$w_i' = \frac{w_i}{W_{total}} \quad (5)$$

Summary of the normalised weight calculation for every unsupervised classifier:

$W_{total} = w_0 + w_1 + w_2 = 1.056 + 1 + 1.044 = 3.1$

$w_0' = \left(\frac{w_0}{w_{total}}\right) = \left(\frac{1.056}{3.1}\right) = 0.34$

$w_1' = \left(\frac{w_1}{w_{total}}\right) = \left(\frac{1}{3.1}\right) = 0.32$

$w_2' = \left(\frac{w_2}{w_{total}}\right) = \left(\frac{1.044}{3.1}\right) = 0.33$

For every unsupervised classifier, the weighted anomaly score is calculated by factoring the normalised weight value with each performance metric in the anomaly score, as shown in Equation 6.

$$Weighted\ anomaly\ score_{Cx} = (w_i' * C_{AS}) \quad (6)$$

Summary of the weighted anomaly scores for the first unsupervised classifier C_0:

$Weighted\ score_{C_{0accuracy}} = (w_0' * C_{AS_{0accuracy}}) = 0.34 * 0.99 = 0.336$

$Weighted\ score_{C_{0precision}} = (w_0' * C_{AS_{0precision}}) = 0.34 * 0.99 = 0.336$

$Weighted\ score_{C_{0recall}} = (w_0' * C_{AS_{0recall}}) = 0.34 * 0.99 = 0.336$

$Weighted\ score_{C_{0F1-score}} = (w_0' * C_{AS_0 F1-score}) = 0.34 * 0.99 = 0.336$

$Weighted\ score_{C_{0TNR}} = (w_0' * C_{AS_{0TNR}}) = 0.34 * 0.99 = 0.336$

$Weighted\ score_{C_{0FPR}} = (w_0' * C_{AS_{0FPR}}) = 0.34 * 0.01 = 0.0034$

$Weighted\ score_{C_{0FNR}} = (w_0' * C_{AS_{0FNR}}) = 0.34 * 0.01 = 0.0034$

Summary of the weighted anomaly scores for the second unsupervised classifier C_1:

$Weighted\ score_{C_{1accuracy}} = (w_1' * C_{AS_{1accuracy}}) = 0.32 * 0.86 = 0.27$

$Weighted\ score_{C_{1precision}} = (w_1' * C_{AS_{1precision}}) = 0.32 * 0.86 = 0.27$

$Weighted\ score_{C_{1recall}} = (w_1' * C_{AS_{1recall}}) = 0.32 * 0.86 = 0.27$

$Weighted\ score_{C_{1F1-score}} = (w_1' * C_{AS_{1F1-score}}) = 0.32 * 0.86 = 0.27$

$Weighted\ score_{C_{1TNR}} = (w_1' * C_{AS_{1TNR}}) = 0.32 * 0.86 = 0.27$

$Weighted\ score_{C_{1FPR}} = (w_1' * C_{AS_{1FPR}}) = 0.32 * 0.06 = 0.0192$

$Weighted\ score_{C_{1FNR}} = (w_1' * C_{AS_{1FNR}}) = 0.32 * 0.06 = 0.0192$

Summary of the weighted anomaly scores for the third unsupervised classifier C_2:

$Weighted\ score_{C_{2accuracy}} = (w_2' * C_{AS_{2accuracy}}) = 0.33 * 0.95 = 0.313$

$Weighted\ score_{C_{2precision}} = (w_2' * C_{AS_2 precision}) = 0.33 * 0.95 = 0.313$

$Weighted\ score_{C_{2recall}} = (w_2' * C_{AS_2 recall}) = 0.33 * 0.95 = 0.313$

$Weighted\ score_{C_{2F1-score}} = (w_2' * C_{AS_2 F1-score}) = 0.33 * 0.95 = 0.313$

$Weighted\ score_{C_{2TNR}} = (w_2' * C_{AS_2 TNR}) = 0.33 * 0.95 = 0.313$

$Weighted\ score_{C_{2FPR}} = (w_2' * C_{AS_2 FPR}) = 0.33 * 0.03 = 0.0099$

$Weighted\ score_{C_{2FNR}} = (w_2' * C_{AS_2 FNR}) = 0.33 * 0.03 = 0.0099$

Finally, the performance metrics of the ensemble anomaly score are computed by summing up the weighted anomaly scores for every performance metric, as shown in Equation 7. The final ensemble anomaly score also includes the majority-voted label.

$$Ensemble\ anomaly\ score = \sum_{i=0}^{N} (w_i' * C_{AS}) \quad (7)$$

Summary of ensemble anomaly score:

Ensemble label: 1

$Accuracy\ ensemble\ score$ = 0.336 + 0.27 + 0.313 = 0.919 = 91.9%

$Precision\ ensemble\ score$ = 0.336 + 0.27 + 0.313 = 0.919 = 91.9%

$Recall\ ensemble\ score$ = 0.336 + 0.27 + 0.313 = 0.919 = 91.9%

$F1 - score\ ensmeble\ score$ = 0.336 + 0.27 + 0.313 = 0.919 = 91.9%

$TNR\ ensemble\ score$ = 0.336 + 0.27 + 0.313 = 0.919 = = 91.9%

$FPR\ ensemble\ score$ = 0.0034 + 0.0192 + 0.0099 = 0.0325 = 3.25%

$FNR\ ensemble\ score$ = 0.0034 + 0.0192 + 0.0099 = 0.0325 = 3.25%

4 Implementation and validation

This chapter discusses the implementation methodology to establish the SHARP framework, depicted in Figure 5. The practical implementation is performed on a 64-bit Windows Operating system with Intel(R) Core (TM) i9-10980HK CPU configuration @ 2.40GHz 3.10 GHz, 16 GB RAM. The implementation utilises Python as the programming language and Jupyter Notebook to create and edit files. Furthermore, the implementation relies on Pandas, Scikit-Learn, Matplotlib, Random, Math, Numpy, Scipy and Counter libraries.

The implementation of the SHARP framework is demonstrated by elaborating activities performed in three Jupyter notebooks named "Data_Preprocessing", "SHARP_98_02", and "SHARP_50_50"[1]. The implementation is divided into two sections: data preprocessing and SHARP. The activities in the first section can be found in the Data_Preprocessing notebook, and the activities in the second section covering the SHARP implementation can be found in the other two notebooks.

4.1 Datasets and preprocessing activities

This subchapter delves into the datasets and preprocessing activities performed to establish the SHARP framework. The University of New Brunswick Centre for Cybersecurity produced the CIC IoT dataset 2023, which contains extensive data on cyber-attacks in IoT networks (Neto et al., 2023). The IoT network contains 105 devices and consists of labelled malicious and benign network traffic encapsulated in a collection of PCAP and CSV files. An in-depth analysis, including the configuration of the IoT network for this dataset, can be found on the official UNB website and in the research paper of Neto et al. (2023). Due to resource limitations, a subset CSV file containing 1% of the total CIC IoT dataset 2023 created by Malhotra (2023) was used for the practical implementation. Table 15 provides an overview of the 10 datasets created to train the UE and evaluate the SHARP framework. The table summarises each dataset's number of rows, columns, and class ratio. The purpose of the datasets can be divided into preprocessing datasets, training datasets and testing datasets. Dataset 1 contains 1% of the CIC IoT dataset 2023 data, consisting of eight labels: DDoS, DoS, Mirai, Benign, Spoofing, Recon, Web, and Bruteforce. This research focuses on DDoS anomaly detection. Therefore, the labels DDoS and benign were extracted and stored in dataset number 2.

[1] Jupyter notebooks of SHARP implementation are available at *https://github.com/imchua/SHARP.git*

Table 15. Overview of the train and test datasets used for the SHARP implementation.

#	Dataset file name	Description	Selected rows	Total rows	Total Columns	DDoS rows	Benign rows
1	CIC_IoT_0.1_percent_8_classes.csv	Original 1% CIC IoT dataset containing 7 attack and benign data.	4.668.653	4.668.653	47	3.398.438	109.824
2	CIC_IoT_0.1_percent_2_classes.csv	Original 1% CIC IoT dataset containing DDoS and benign data.	3.508.262	3.508.262	47	3.398.438	109.824
3	CIC_IoT_0.1_percent_2_classes_cleaned.csv	Pre-processed dataset. Contains DDoS and benign data.	3.508.262	3.508.262	64	3.398.438	109.824
4	CIC_IoT_0.1_percent_novelty_train_set.csv	Train novelty models. Contains only benign data.	Benign: 0 - 20.000	20.000	64	20.000	0
5	CIC_IoT_0.1_percent_outlier_train_set.csv	Train outlier model. Contains DDoS and benign data.	DDoS: 0 - 400 Benign: 0 – 19.600	20.000	64	19.600	400
6	CIC_IoT_0.1_percent_2_classes_test_it1_98_02.csv	Used for testing iteration 1 with 98-02 class ratio. Contains DDoS and benign data.	DDoS: 500 - 600 Benign: 21.000 – 25.900	5.000	64	100	4.900
7	CIC_IoT_0.1_percent_2_classes_test_it1_50_50.csv	Used for testing iteration 1 with 50-50 class ratio. Contains DDoS and benign data.	DDoS: 700 – 3.200 Benign: 26.000-28.500	5.000	64	2.500	2.500

#	Dataset file name	Description	Selected rows	Total rows	Total Columns	DDoS rows	Benign rows
8	CIC_IoT_0.1_percent_2_classes_test_it2_98_02.csv	Used for testing iteration 2 with 98-02 class ratio. Contains DDoS and benign data.	DDoS: 3.300 – 3.400 Benign: 29.000 – 33.900	5.000	64	100	4.900
9	CIC_IoT_0.1_percent_2_classes_test_it2_50_50.csv	Used for testing iteration 2 with 50-50 class ratio. Contains DDoS and benign data.	DDoS: 3.500 – 6.000 Benign: 34.000-36.500	5.000	64	2.500	2.500
10	CIC_IoT_0.1_percent_2_classes_test_it4_benign_only.csv	Used for testing iteration 4 only benign data. Contains only benign data.	Benign: 45.000-50.000	5.000	64	0	5.000

Pre-processing activities were carried out on the second dataset to enhance its quality. Subsequently, the refined data was stored in the third dataset to create training and test datasets. The pre-processing activities included observing data characteristics, removing missing and null values, observing duplicate values, and transforming categorical features into numerical values using Scikit-Learn's LabelEncoder and OneHotEncoder. Duplicate values were retained because they can aid the unsupervised classifiers during the learning process, as this repetitive nature may be an important indicator. Feature selection techniques were not applied to keep the features as close to the original dataset. Table 16 compares the dataset's original features against those after pre-processing the data.

Table 16. Feature comparison of original and pre-processed datasets.

Dataset	Features
Original	'flow_duration', 'header_length', 'protocol_type', 'duration', 'rate', 'srate', 'drate', 'fin_flag_number', 'syn_flag_number', 'rst_flag_number', 'psh_flag_number', 'ack_flag_number', 'ece_flag_number', 'cwr_flag_number', 'ack_count', 'syn_count', 'fin_count', 'urg_count', 'rst_count', 'http', 'https', 'dns', 'telnet', 'smtp', 'ssh', 'irc', 'tcp', 'udp', 'dhcp', 'arp', 'icmp', 'ipv', 'llc', 'tot_sum', 'min', 'max', 'avg', 'std', 'tot_size', 'iat', 'number', 'radius', 'covariance', 'variance', 'weight', 'label', 'magnitude'
Pre-processed	'flow_duration', 'header_length', 'duration', 'rate', 'srate', 'drate', 'fin_flag_number', 'syn_flag_number', 'rst_flag_number', 'psh_flag_number', 'ack_flag_number', 'ece_flag_number', 'cwr_flag_number', 'ack_count', 'syn_count', 'fin_count', 'urg_count', 'rst_count', 'http', 'https', 'dns', 'telnet', 'smtp', 'ssh', 'irc', 'tcp', 'udp', 'dhcp', 'arp', 'icmp', 'ipv', 'llc', 'tot_sum', 'min', 'max', 'avg', 'std', 'tot_size', 'iat', 'number', 'radius', 'covariance', 'variance', 'weight', 'label', 'magnitude', 'protocol_type_ARGUS (deprecated)', 'protocol_type_BBN-RCC-MON', 'protocol_type_CBT', 'protocol_type_CHAOS', 'protocol_type_EGP', 'protocol_type_EMCON', 'protocol_type_GGP', 'protocol_type_HOPOPT', 'protocol_type_ICMP', 'protocol_type_IGMP', 'protocol_type_IGP', 'protocol_type_IPv4', 'protocol_type_NVP-II', 'protocol_type_PUP', 'protocol_type_ST', 'protocol_type_TCP', 'protocol_type_UDP', 'protocol_type_XNET'

The third dataset was used to establish the training and testing datasets. The fourth and fifth datasets comprise 20,000 rows of training data and facilitated the training of novelty-based as well as outlier-based unsupervised classifiers. The novelty training dataset exclusively contains benign data and is used to train OCSVM and LOF. iForest, on the other hand, utilised a combination of DDoS and benign data for its training. Hence, the outlier train dataset contains a 98-02 percentage ratio of DDoS and benign data. Another essential pre-processing activity was normalising the data using Scikit-Learn's MinMaxScaler function. Data normalisation was performed on both the training and testing data.

However, while OCSVM and iForest benefited from normalising the data, LOF showed decreased performance when trained on normalised data. Consequently, LOF was trained and tested on non-normalised training and test data.

The SHARP framework was evaluated by examining the outcomes of four iterative cycles. Datasets 6 to 10, each containing 5,000 rows of test data, were used to assess the results of each iteration. The first two iterations use two test datasets, one consisting of a 98-02 percentage ratio of DDoS and benign traffic and the other a 50-50 percentage ratio. Whereas the 98-02 test set portrays a realistic anomaly scenario, the 50-50 test set, on the other hand, was used to evaluate the unsupervised classifier's performance using a balanced dataset. The third iteration required specific performance metric values to evaluate the desired scenario. Therefore, it did not use a test dataset file but used static predetermined performance metrics values to portray the desired scenario instead. The static performance metric values for the three unsupervised classifiers were defined as follows:

- Iteration 3 OCSVM performance evaluation = 88, 93, 92, 92, 93, 6.0, 6.0
- Iteration 3 LOF performance evaluation = 89, 92, 93, 93, 94, 5.5, 5.5
- Iteration 3 iForest performance evaluation = 85, 86, 86, 86, 86, 12, 12

The final iteration examined the SHARP framework's outcome when experiencing normal traffic behaviour. To mimic this scenario, the last test dataset exclusively contains benign data.

4.2 SHARP implementation

This subchapter discusses the implementation of SHARP's foundational components, focusing on the UE model and EXP4. The first section provides the methodology for the UE by discussing the training and testing activities performed on OCSVM, LOF, and iForest. The second section covers the implementation of EXP4.

4.2.1 Implementation UE

The unsupervised classifiers OCSVM, LOF and iForest were trained using the novelty and outlier training datasets to establish the UE. Python's Scikit-Learn fit() and predict() functions were leveraged to train and test the unsupervised classifiers. The parameters used to initialise and train the unsupervised classifiers are listed in Table 17.

Table 17. Overview parameters of unsupervised classifiers in UE model.

Model	Parameter
OCSVM	kernel='rbf', nu=0.001, gamma=1.5152
LOF	n_neighbors=25, metric='euclidean', contamination=0.001, novelty=True
iForest	contamination=0.025, max_features=29, n_estimators=29, random_state=42

The data ratio for the training and test sets follows the experimental configurations of the supervised and unsupervised EL approaches of Das et al. (2024) and Saha et al. (2022) to provide a comparable evaluation scenario for the unsupervised classifier's performance. An overview of the training and testing methods used for each unsupervised classifier is as follows:

- OCSVM: trained on normalised novelty train data and tested on normalised test data, which include the 98-02 and 50-50 datasets.
- LOF: trained on original novelty train data and tested on original test data, which include the 98-02 and 50-50 datasets.
- iForest: trained on normalised outlier train data and tested on normalised test data, which include the 98-02 and 50-50 datasets.

The unsupervised classifiers were evaluated against the 98-02 test ratio datasets, 50-50 test ratio datasets, and benign-only datasets, as outlined in Table 15. Scikit-Learn's Library's metrics were used to implement the performance metrics: accuracy, precision, recall, and F1-score. Additionally, a confusion matrix was used to determine the TP, FP, TN, and FN.

The derived indicators from the confusion matrix were then used to compute the TNR, known as specificity, FPR and FNR using the equations shown in Table 9.

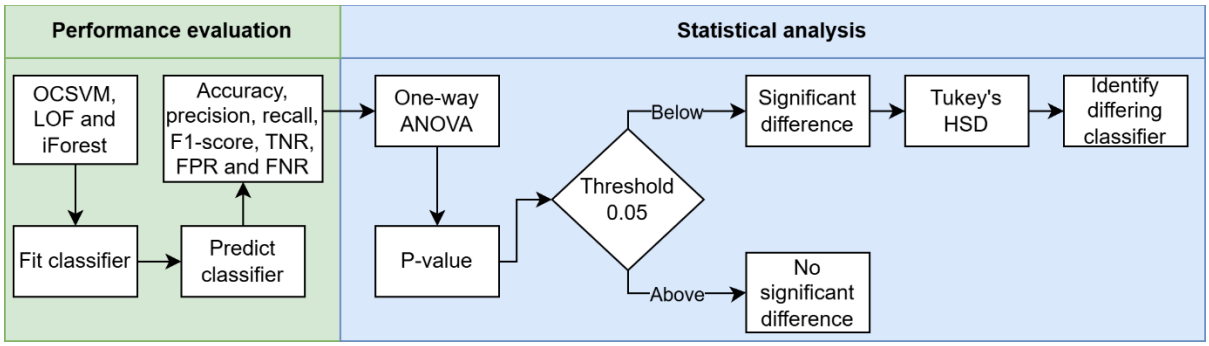

Figure 6. Overview performance evaluation.

An overview of the evaluation process for the UE is shown in Figure 6. To observe whether a significant difference was present in the performance metrics among OCSVM, LOF and iForest, one-way ANOVA tests were performed for each performance metric. The null hypothesis used for the one-way ANOVA test assumed that there was no difference in performance metrics among the three unsupervised classifiers and is denoted in Equation 8.

$$H_0 = \mu_1 = \mu_2 = \mu_3 \tag{8}$$

The significant threshold for the null hypothesis was set to 0.05, where if a p-value is below the threshold, the null hypothesis is rejected, indicating that a statistical difference in the performance metrics exists between at least two of the unsupervised classifiers. When the p-value was above the threshold, the null hypothesis failed to be rejected, indicating that there was not enough information to conclude that there was a significant difference in performance metrics between the unsupervised classifiers. Furthermore, Tukey's HSD was applied using Scipy's statistics library when a significant difference in a performance metric was observed. Tukey's HSD test revealed which unsupervised classifiers differed from each other.

4.2.2 Implementation EXP4

The EXP4 algorithm performed four iterations using the 98-02 and 50-50 percentage ratio datasets to evaluate the SHARP framework. The iterations represented a combination of one of the three actions with a corresponding status impact value. The objective of the four iterations was to observe the output for different scenarios and evaluate the proposed hypotheses, as stated in Table 18. While the actions and status impact in this implementation were static variables, the actions must be connected to an executable activity in the production environment. Additionally, the status impact should be monitored in a live production environment through network monitoring tools or manual assessment.

The deployment of EXP4 in the SHARP framework is elaborated by addressing the practical implementation of the seven fundamental steps of EXP4. Sequentially, the construction of the final ensemble anomaly score, consisting of a majority-voted label and the ensemble anomaly scores of the performance metrics: accuracy, precision, recall, F1-score, TNR, FPR and FNR, is discussed.

Table 18. Four iterative cycles of SHARP.

#	Scenario	Action	Status impact	Hypothesis	Test dataset
1	Experts advise blocking network traffic based on the unsupervised classifier's evaluation. After executing the block action, the network traffic improves.	Block	Accurate	Higher importance for experts who advised the block action. Other experts experience no change in importance.	#6 & #7
2	Experts advise ignoring network traffic based on the unsupervised classifier's evaluation. After executing the ignoring action, the network traffic does not improve.	Ignore	Inaccurate	Lower importance for experts who chose to ignore the action. Other experts experience no change in importance.	#8 & #9
3	Experts advise limiting network traffic based on the unsupervised classifier's evaluation. After executing the limit action, the network traffic experiences a slight improvement but does not fully resolve the high traffic.	Limit	Partially accurate	Slight increase for experts who advised the limit action. Other experts experience no change in importance.	-
4	Experts advise ignoring network traffic based on the unsupervised classifier's evaluation. After executing the ignore action, the network traffic does not experience any difference as intended.	Ignore	Accurate	Higher importance for experts who advised the ignore action. Other experts experience no change in importance.	#10

Step 1) Initialisation

The variables, as outlined in Table 10, were instantiated. A crucial step was initialising three experts, each representing an unsupervised classifier. A Python dictionary was employed to store variables and results of EXP4, where each key represents an expert. At the initial iteration, the weights of every expert were assigned the value 1. Noteworthy is that this implementation instantiated three actions in its action space: block, limit, and ignore. However, the actions must be connected to an executable activity in a live production environment.

Step 2) Obtain label and expert advice vector

The labelling system based on the evaluation of an unsupervised classifier was realised by computing a weighted evaluation score, which was compared against a certain threshold. A range between 0 and 100 was used to compare the weighted evaluation score, where 100 implied a perfect score.

The threshold value for the weighted evaluation score was set to 80, as this number separated well-performing and underperforming unsupervised classifiers. The evaluation was labelled 1 (DDoS) when the threshold was surpassed. Otherwise, the evaluation was identified as benign traffic and is labelled with 0. The weighted evaluation score was computed by factoring a predetermined weight value with the respective performance metrics. The predetermined weight values were allocated based on the importance of the performance metric and were divided as follows:

1. High-importance performance metrics received a weight value of 0.4. This category included precision, recall, F1-score, FPR, and FNR. These metrics were classified as high-importance metrics because they are important for outlier detection and can quantify misclassification.

2. Medium-importance performance metrics received a weight value of 0.15. This category contained TNR. While TNR quantifies the number of benign traffic classified correctly, the primary focus is to identify DDoS attacks.

3. Low-importance performance metrics received a weight value of 0.05. This category contained accuracy. The accuracy metric was assigned low importance as it may give misleading results in imbalanced scenarios.

The advice vector for every expert was assigned using the conditional rules outlined in Table 11. When the evaluation of the unsupervised classifier matched a conditional rule, the corresponding advice vector was assigned to the expert and was stored in the expert's key in the dictionary.

Step 3) Calculate the probability value for every action

The advice vector for every action was defined by selecting the advice vector values of every expert for the respective action. Using the advice vector of every action, the probability of every action was computed using Equation 1.

Step 4) Select an action and obtain a reward

The action to be selected was determined through a probabilistic system that considered the number of actions and action probability values. Once the action is selected, it should be executed in the live production environment, and its impact on the network should be observed to receive the corresponding reward.

Due to the scope of this study, three static scenarios were defined that simulated the observation of the status impact on the network. The three simulated static status impact scenarios used were accurate, partially accurate, and inaccurate. The reward for each expert was obtained by assessing the rule that aligned with the status impact, using the conditional logic of Tables 12, 13, and 14. The rewards were then stored in the dictionary at the corresponding expert key.

Step 5) Calculate the estimated reward

Equation 2 computed the estimated reward for every expert. Simultaneously, the condition where the estimated reward is solely computed for experts who provided the highest advice vector value on the selected action was enforced. Otherwise, experts received an estimated reward of value 0. The estimated reward of each expert was then stored at the corresponding expert key in the dictionary.

Step 6) Calculate the contribution of every expert

The estimated reward and the expert's advice vectors were used to compute each expert's contribution via Equation 3 and were stored in the dictionary.

Step 7) Calculate and update the new weight value for every expert

The new weight value for every expert was computed using Equation 4, which was updated in the dictionary for the next iteration.

```
{'i_OCSVM': [1.0498, [0.8, 0.15, 0.05], 1.0, 1.821, 1.4568],
 'i_LOF': [1.0561, [0.9, 0.08, 0.02], 1.0, 1.821, 1.6389],
 'i_iForest': [1.0, [0.02, 0.08, 0.9], 0, 0, 0.0]}
```

Figure 7. Results of EXP4 variables shown in a Python dictionary for the iteration.

The EXP4 steps were then considered finished for the corresponding iteration. Every expert key in the dictionary contained the variables in the following sequence: updated weight value, expert advice vector, reward, estimated reward and expert contribution (Figure 7).

Step 8: Calculate ensemble anomaly score

To compute a single ensemble anomaly score, the performance metrics of all unsupervised classifier evaluations, excluding the label, were factored with the newly updated and normalised expert weight value, as illustrated in Equations 5 and 6.

This resulted in seven weighted performance metric values for every unsupervised classifier, which were attached to the corresponding expert key in the dictionary. The ensemble anomaly score combined the weighted performance metrics of the three unsupervised classifiers by summing up the corresponding metrics to establish a single performance metric for the final ensemble anomaly score. Moreover, the most common label among the three experts was the majority-voted label, which was also recorded in the final ensemble anomaly score.

```
ensemble_anomaly_score

[1, 0.9818, 0.9973, 0.9841, 0.9906, 0.868, 0.132, 0.0159]
```

Figure 8. Final ensemble anomaly score.

The SHARP iteration was considered finished, as the final ensemble anomaly score contained all the results, including the majority-voted label and the ensemble anomaly performance scores: accuracy, precision, recall, F1-score, TNR, FPR, and FNR (Figure 8).

5 Results and Discussion

This chapter presents the key findings of the development and implementation of SHARP's framework, derived from the literature review on ML-based ANIDS. Furthermore, this chapter discusses ML-based ANIDS utilising UE and RL, SHARP's UE model, and SHARP's ensemble anomaly scores. SHARP enhances dynamic decision-making and addresses the exploration-exploitation trade-off by integrating an UE model to facilitate UL, while leveraging EXP4 to incorporate RL to detect DDoS anomalies in a smart home network.

5.1 Results UE of SHARP

This subchapter presents the UE model results of the SHARP framework. The UE results present the quantitative performance findings of the three unsupervised classifiers, OCSVM, LOF, and iForest. The results are divided into two sections corresponding to the 98-02 and 50-50 percentage ratio test datasets, derived from prior experimental configurations observed in the literature. Each section presents the findings regarding the performance metrics of the three unsupervised classifiers, accompanied by the ANOVA test for each performance metric. The post-hoc Tukey's HSD test was performed on results that showed p-values below the null hypothesis threshold of 0.05. Following the results of the two test scenarios, the UE results of both test scenarios were compared.

5.1.1 UE results 98-02 test scenario

Tables 19 and 20 present the UE results and the mean average scores for the 98-02 percentage ratio test dataset. Furthermore, Figures 9, 10, 11, and 12 show the results of the Tukey's HSD tests.

Table 19. Results testing UE on 98-02 datasets.

Unsupervised classifier	Iteration	Accuracy	Precision	Recall	F1-score	FPR	TNR	FNR
OCSVM	1	98.74	100.0	98.71	99.35	0.0	100.0	1.29
	2	99.14	99.98	99.14	99.56	1.0	99.0	0.86
	3	-	-	-	-	-	-	-
	4	98.90	100.0	98.90	99.45	0.0	0.0	1.10
OCSVM average it 1&2		98.94	99.99	98.93	99.46	0.50	99.5	0.75
LOF	1	99.78	100.0	99.78	99.89	0.0	100.0	0.22
	2	99.82	100.0	99.82	99.90	0.0	100.0	0.18
	3	-	-	-	-	-	-	-
	4	99.92	100.0	99.92	99.96	0.0	0.0	0.08
LOF average it 1&2		99.80	100.0	99.8	99.9	0.0	100.0	0.20
iForest	1	97.02	99.17	97.78	98.47	40.00	60.0	2.22
	2	96.78	99.00	97.69	98.34	48.0	52.0	2.30
	3	-	-	-	-	-	-	-
	4	97.84	100.0	97.84	98.90	0.0	0.0	2.16
iForest average it 1&2		96.90	99.09	97.73	98.40	44.00	56.00	2.26

Table 20. Mean averages and p-values for 98-02 ratio test dataset of iteration 1, 2, and 4.

Performance metric	Mean	P-value
Accuracy	98.66	0.00024
Precision	97.80	0.08368
Recall	98.42	0.0000052
F1-score	99.32	0.00029
TNR	56.78	0.73008
FPR	9.89	0.08372
FNR	1.16	0.0000052

```
Tukey's HSD Pairwise Group Comparisons (95.0% Confidence Interval)
Comparison  Statistic  p-value  Lower CI  Upper CI
 (0 - 1)     -0.913     0.040    -1.775    -0.052
 (0 - 2)      1.713     0.002     0.852     2.575
 (1 - 0)      0.913     0.040     0.052     1.775
 (1 - 2)      2.627     0.000     1.765     3.488
 (2 - 0)     -1.713     0.002    -2.575    -0.852
 (2 - 1)     -2.627     0.000    -3.488    -1.765
```

Figure 9. Tukey's HSD result of accuracy metric on 98-02 percentage dataset.

Tukey's HSD Pairwise Group Comparisons (95.0% Confidence Interval)

Comparison	Statistic	p-value	Lower CI	Upper CI
(0 - 1)	-0.918	0.000	-1.264	-0.573
(0 - 2)	1.149	0.000	0.804	1.495
(1 - 0)	0.918	0.000	0.573	1.264
(1 - 2)	2.067	0.000	1.722	2.413
(2 - 0)	-1.149	0.000	-1.495	-0.804
(2 - 1)	-2.067	0.000	-2.413	-1.722

Figure 10. Tukey's HSD result of recall metric on 98-02 percentage dataset.

Tukey's HSD Pairwise Group Comparisons (95.0% Confidence Interval)

Comparison	Statistic	p-value	Lower CI	Upper CI
(0 - 1)	-0.465	0.046	-0.921	-0.009
(0 - 2)	0.879	0.003	0.423	1.335
(1 - 0)	0.465	0.046	0.009	0.921
(1 - 2)	1.344	0.000	0.888	1.800
(2 - 0)	-0.879	0.003	-1.335	-0.423
(2 - 1)	-1.344	0.000	-1.800	-0.888

Figure 11. Tukey's HSD result of F1-score metric on 98-02 percentage dataset.

Tukey's HSD Pairwise Group Comparisons (95.0% Confidence Interval)

Comparison	Statistic	p-value	Lower CI	Upper CI
(0 - 1)	0.918	0.000	0.573	1.264
(0 - 2)	-1.149	0.000	-1.495	-0.804
(1 - 0)	-0.918	0.000	-1.264	-0.573
(1 - 2)	-2.067	0.000	-2.413	-1.722
(2 - 0)	1.149	0.000	0.804	1.495
(2 - 1)	2.067	0.000	1.722	2.413

Figure 12. Tukey's HSD result of FNR metric on 98-02 percentage dataset.

The accuracy metric rejected the null hypothesis, with a mean accuracy of 98.66% and a p-value below 0.05, illustrating a statistical difference between the unsupervised classifiers. To discover the difference between the unsupervised classifiers, Tukey's HSD post hoc test was applied, and it revealed that the mean accuracy of OCSVM surpassed the mean accuracy of iForest. Moreover, LOF showed a higher mean accuracy than OCSVM by 0.913 while surpassing the mean accuracy of iForest by 2.627.

The precision reached a mean score of 97.80%. However, the p-value of 0.08368 showed that the precision failed to reject the null hypothesis.

Furthermore, a recall mean score of 98.42% was observed. Additionally, a p-value below 0.05 showed that a statistical difference between the three unsupervised classifiers was present, thus rejecting the null hypothesis. Via Tukey's HSD post hoc test, it was observed that the recall mean of OCSVM surpassed the recall mean of iForest with a difference of 1.149.

Moreover, the mean of LOF's recall score was slightly higher than OCSVM's recall by 0.918 and outperformed iForest's recall mean by 2.067.

The F1-score yielded a mean score of 99.32%, rejecting the null hypothesis as the p-value was below 0.05. Tukey's HSD post hoc test results demonstrated that OCSVM maintained a higher mean F1-score than iForest by 0.879. In addition, LOF had a greater mean average than OCSVM by 0.465. The most significant difference was observed in the F1-scores of LOF and iForest, with a difference of 1.344.

Contrary to the F1-score, the TNR recorded the lowest mean score at 56.78%. Additionally, the TNR failed to reject the null hypothesis, as the p-value was significantly larger than 0.05. While OCSVM and LOF maintained low TNR scores, iForest was the only unsupervised classifier that produced a high TNR score.

Similarly, the FPR score, with a mean of 9.89 and a p-value above 0.05, showed a comparable pattern, as iForest was the only unsupervised classifier to produce higher FPR scores than OCSVM and LOF.

The mean of FNR was 1.16 and yielded a p-value of below 0.05. Tukey's HSD post hoc test results revealed that OCSVM performed slightly better than the FNR mean of LOF by 0.918. The results also revealed that the FNR mean of iForest surpassed the FNR of OCSVM and LOF mean by 1.149 and 2.067, respectively.

5.1.2 UE results 50-50 test scenario

Tables 21 and 22 present the UE results and the mean average scores for the 50-50 percentage ratio test dataset. Moreover, Figures 13 and 14 present the results of the Tukey's HSD tests.

Table 21. Results testing UE on 50-50 datasets.

Unsupervised classifier	Iteration	Accuracy	Precision	Recall	F1-score	FPR	TNR	FNR
OCSVM	1	98.50	98.09	98.92	98.50	1.92	98.08	1.08
	2	98.38	98.09	98.68	98.38	1.92	98.08	1.32
	3	-	-	-	-	-	-	-
	4	98.90	100.0	98.9	99.44	0.0	0.0	1.09
OCSVM average it 1&2		98.44	98.09	98.80	98.44	1.92	98.08	1.2
LOF	1	99.96	100.0	99.92	99.95	0.0	100.0	0.08
	2	100.0	100.0	100.0	100.0	0.0	100.0	0.0
	3	-	-	-	-	-	-	-
	4	99.92	100.0	99.92	99.95	0.0	0.0	0.08
LOF average it 1&2		99.98	100.0	99.96	99.97	0.0	100.0	0.04
iForest	1	75.20	67.53	97.04	79.65	46.64	53.36	2.96
	2	75.42	67.55	97.84	79.92	47.0	53.0	2.16
	3	-	-	-	-	-	-	-
	4	97.84	100.0	97.84	98.90	0.0	0.0	2.16
iForest average it 1&2		75.31	67.54	97.44	79.78	46.82	53.18	2.56

Table 22. Mean averages and p-values for 50-50 ratio test dataset of iteration 1, 2, and 4.

Performance metric	Mean	P-value
Accuracy	97.80	0.05637
Precision	92.36	0.08717
Recall	98.78	0.00014
F1-score	94.97	0.06904
TNR	55.84	0.70217
FPR	10.83	0.08451
FNR	1.21	0.00014

```
Tukey's HSD Pairwise Group Comparisons (95.0% Confidence Interval)
Comparison  Statistic  p-value  Lower CI  Upper CI
 (0 - 1)     -1.113     0.007    -1.812    -0.415
 (0 - 2)      1.260     0.004     0.562     1.958
 (1 - 0)      1.113     0.007     0.415     1.812
 (1 - 2)      2.373     0.000     1.675     3.072
 (2 - 0)     -1.260     0.004    -1.958    -0.562
 (2 - 1)     -2.373     0.000    -3.072    -1.675
```

Figure 13. Tukey's HSD result of recall metric on 50-50 percentage dataset.

Tukey's HSD Pairwise Group Comparisons (95.0% Confidence Interval)

Comparison	Statistic	p-value	Lower CI	Upper CI
(0 - 1)	1.113	0.007	0.415	1.812
(0 - 2)	-1.260	0.004	-1.958	-0.562
(1 - 0)	-1.113	0.007	-1.812	-0.415
(1 - 2)	-2.373	0.000	-3.072	-1.675
(2 - 0)	1.260	0.004	0.562	1.958
(2 - 1)	2.373	0.000	1.675	3.072

Figure 14. Tukey's HSD result of FNR metric on 50-50 percentage dataset.

The accuracy mean of the 50-50 percentage ratio test dataset was 97.80% and failed to reject the null hypothesis due to the p-value slightly exceeding the 0.05 threshold. A similar observation can be seen in the precision score. While the precision mean reached 92.36%, the p-value exceeded 0.05, which indicated a failure to reject the null hypothesis.

The results revealed that the recall metric reached the highest mean score at 98.78%, and with the corresponding p-value below 0.05, a difference between the unsupervised classifier's recall was observed. Tukey's HSD post hoc test revealed that OCSVM's mean surpassed the mean of iForest by 1.260. Furthermore, LOF's recall mean was superior to OCSVM's by 1.113 and iForest's by 2.373.

Regarding the F1-scores, the mean reached 94.97%, accompanied by a p-value slightly exceeding the 0.05 threshold. Therefore, the F1-score metric failed to reject the null hypothesis.

The TNR mean was 55.84%, and iForest was the sole unsupervised classifier that produced high TNR values. Additionally, the TNR p-value was significantly above the 0.05 threshold, thus indicating the failure to reject the null hypothesis.

Similarly to iForest's TNR, the FPR results of iForest were significantly higher than OCSVM and LOF, resulting in a mean of 10.83%. The p-value was found to have slightly exceeded the 0.05 threshold. Thus, it was observed that FPR failed to reject the null hypothesis.

Finally, the mean average of FNR reached 1.21, and the p-value was significantly below the 0.05 threshold. The Tukey's HSD post hoc test results demonstrated that OCSVM's FNR surpassed LOF's by 1.113. Moreover, iForest's FNR mean surpassed the mean outcomes of OCSVM and LOF by 1.260 and 2.373.

5.1.3 Comparison UE results 98-02 and 50-50 test scenarios

Based on the results of the two test scenarios, LOF outperformed the other two models, followed by OCSVM and iForest. A slight marginal difference of 1-2% was observed for OCSVM when comparing its results of the two test scenarios. Comparing the performance metrics values of LOF for both test scenarios revealed a minimal difference of less than 1%. On the other hand, iForest demonstrated the lowest performance metrics across the results of both test scenarios.

Comparing the mean differences between the two test scenarios showed that the 98-02 test scenario achieved higher mean values than the 50-50 test scenario. Figure 15 provides an overview of the comparison of performance metrics means for both test scenarios of iterations 1, 2, and 4.

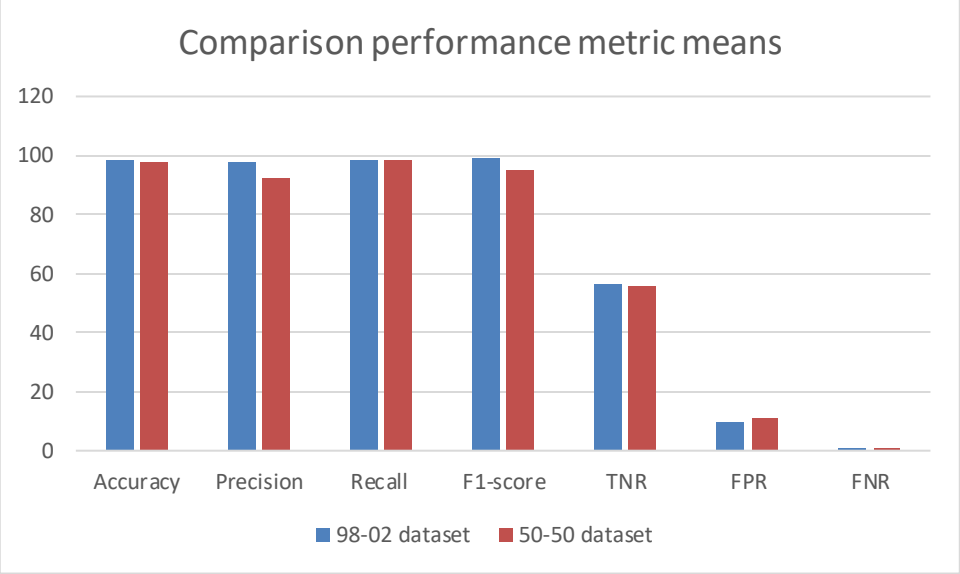

Figure 15. Comparison UE performance means between 98-02 and 50-50 percentage datasets.

The most significant marginal disparity was observed in the mean precision, with a difference of 5.44%. The 98-02 test scenario difference surpassed the mean accuracy by 0.86% and the F1-score mean by 4.35%. Additionally, the 98-02 test scenario revealed an increase of 0.94% in the TNR mean compared to the 50-50 test scenario. Notably, the 50-50 test scenario yielded a slightly higher mean for the recall by 0.36% when compared to the 98-02 test scenario. Finally, a slight difference was demonstrated in the mean of FPR and FNR, as the 98-02 test scenario illustrated slightly lower values at 0.94% and 0.05%.

5.2 Results SHARP Framework

This subchapter presents the results of the SHARP framework using two test scenarios: 98-02 and 50-50 percentage ratio test datasets. The findings included the outcome of the EXP4 application and the ensemble anomaly scores across four iterations. After demonstrating the results of both test scenarios, a comparative analysis was conducted to observe the differences.

5.2.1 SHARP results 98-02 test scenario

Tables 23 and 24 present an overview of the EXP4 variables and the ensemble anomaly score for all four iterations using the 98-02 percentage ratio test datasets.

Table 23. Results EXP4 for 98-02 percentage ratio test dataset.

Model	It.	Weight start	Weight updated	Advice vector	Reward	Estimated reward	Expert contribution
OCSVM	1	1.0	1.053	0.9, 0.08, 0.02	1.0	1.727	1.554
	2	1.053	1.053	0.9, 0.08, 0.02	0.0	0.0	0.0
	3	1.053	1.078	0.08, 0.9, 0.02	0.5	0.761	0.685
	4	1.078	1.116	0.02, 0.08, 0.9	1.0	1.161	1.045
LOF	1	1.0	1.053	0.9, 0.08, 0.02	1.0	1.727	1.554
	2	1.053	1.053	0.9, 0.08, 0.02	0.0	0.0	0.0
	3	1.053	1.078	0.08, 0.9, 0.02	0.5	0.761	0.685
	4	1.078	1.116	0.02, 0.08, 0.9	1.0	1.161	1.045
iForest	1	1.0	1.0	0.02, 0.08, 0.9	0.0	0.0	0.0
	2	1.0	0.909	0.02, 0.08, 0.9	-1.0	-3.165	-2.848
	3	0.909	0.909	0.1, 0.2, 0.7	0.0	0.0	0.0
	4	0.909	0.942	0.02, 0.08, 0.9	1.0	1.161	1.045

Table 24. Results ensemble anomaly score for 98-02 percentage ratio test dataset

It.	Label	Accuracy	Precision	Recall	F1-score	FPR	TNR	FNR
1	1.0	98.54	99.73	98.77	99.25	12.88	87.12	1.23
2	1.0	98.66	99.69	98.94	99.31	14.85	85.12	1.06
3	1.0	87.55	90.66	90.66	90.66	7.61	91.37	7.61
4	0.0	99.04	100	99.04	99.57	0.0	0.0	1.06
Average it. 1 & 2	1.0	98.6	99.71	98.86	99.28	13.87	86.12	1.15

The results demonstrated the similarity between OCSVM and LOFS' EXP4 outcomes, as both unsupervised classifiers present identical EXP4 variable values. The experts of OCSVM and LOF obtained neutral and positive reward values across the four iterations. These rewards affected the expert's weight value as a slow increment was observed during the iterations. On the other hand, iForest demonstrated a decrement in the corresponding expert weight value due to the expert receiving a combination of neutral, negative and positive rewards. Despite this, the expert's weight in iForest recovered and experienced a slight improvement after the fourth iteration, as it received a positive reward.

The results of the ensemble anomaly scores showed that in three out of four iterations, the test data was labelled with 1.0 (DDoS). The first two iterations revealed the ensemble anomaly scores for tests containing DDoS data, and the mean scores of these iterations were computed as these iterations reflect actual DDoS scenarios. The first and second iterations revealed that the ensemble accuracy, precision, recall, and F1-score exceeded 98% and had an FNR below 2%. Notably, the ensemble TNR scored low, between 85% and 87%. Additionally, the ensemble FPR score exceeded 10%. The third iteration presented the ensemble anomaly scores, where the test data may indicate the presence of DDoS. This was reflected as the ensemble FPR and FNR scores were below 10%, and other performance metrics ranged between 87% and 91%. The final iteration used test data without any DDoS data. The results of the fourth iteration showed the label assignment of 0.0 and the ensemble scores of 0.0 for both TNR and FPR, indicating that no DDoS data was identified. Furthermore, this iteration achieved the highest ensemble accuracy, precision, recall and F1-score performance.

5.2.2 SHARP results 50-50 test scenario

Tables 25 and 26 summarise the EXP4 results and the ensemble anomaly score for all four iterations utilising the 50-50 percentage ratio test datasets.

Table 25. Results EXP4 for 50-50 percentage ratio test dataset.

Model	It.	Weight start	Weight updated	Advice vector	Reward	Estimated reward	Expert contribution
OCSVM	1	1.0	1.053	0.9, 0.08, 0.02	1.0	1.727	1.554
	2	1.053	1.053	0.9, 0.08, 0.02	0.0	0.0	0.0
	3	1.053	1.078	0.08, 0.9, 0.02	0.5	0.761	0.685
	4	1.078	1.116	0.02, 0.08, 0.9	1.0	1.161	1.045
LOF	1	1.0	1.053	0.9, 0.08, 0.02	1.0	1.727	1.554
	2	1.053	1.053	0.9, 0.08, 0.02	0.0	0.0	0.0
	3	1.053	1.078	0.08, 0.9, 0.02	0.5	0.761	0.685
	4	1.078	1.116	0.02, 0.08, 0.9	1.0	1.161	1.045
iForest	1	1.0	1.0	0.02, 0.08, 0.9	0.0	0.0	0.0
	2	1.0	0.909	0.02, 0.08, 0.9	-1.0	-3.165	-2.848
	3	0.909	0.909	0.1, 0.2, 0.7	0.0	0.0	0.0
	4	0.909	0.942	0.02, 0.08, 0.9	1.0	1.161	1.045

Table 26. Results ensemble anomaly score for 50-50 percentage ratio test dataset.

It.	Label	Accuracy	Precision	Recall	F1-score	FPR	TNR	FNR
1	1.0	91.49	88.90	98.65	92.93	15.67	84.33	1.35
2	1.0	92.02	89.53	98.89	93.37	14.86	85.14	1.11
3	1.0	87.55	90.66	90.66	90.66	7.61	91.37	7.61
4	0.0	99.04	100	99.04	99.57	0.0	0.0	1.06
Average it. 1 & 2	1.0	91.76	89.21	98.77	93.15	15.27	84.74	1.23

The experts of OCSVM and LOF showed identical results when assessing the EXP4 iteration across the four iterations. While the expert evaluations of OCSVM and LOF obtained neutral and positive rewards, it was observed that the expert of iForest received a collection of negative, neutral and positive rewards. Consequently, OCSVM and LOF experienced an increase in weight. At the same time, iForest demonstrated a fluctuation as it first experienced a decrease in weight value but recovered as soon as the expert obtained a positive reward.

The first and second iterations used DDoS test data, which is reflected in the ensemble anomaly scores of the first and second iterations, as they labelled the test data as 1.0 (DDoS). The results of the first two iterations also revealed fluctuation in the ensemble accuracy, precision, recall and F1-score, ranging between 88% and 98%. Furthermore, the TNR was recorded at 84-85%, whereas the FPR ranged between 14-15% for the first two iterations. The findings revealed that the FNR remained between 1.1% and 1.35% during these iterations. The third iteration provided the results of a test scenario where the presence of DDoS data was possible, and thus, the ensemble anomaly score was assigned the label 1.0. The ensemble anomaly scores of the third iteration experienced a decrease, where scores ranged between 87 and 91%. The TNR in the third iteration achieved the highest value overall at 91%. Moreover, the FPR and FNR were below 10% and shared the same values at 7.61%. The fourth iteration used test data consisting of only benign data. The ensemble anomaly scores classified the data as benign. This iteration produced the highest ensemble anomaly score with the ensemble accuracy, precision, recall and F1-scores ranging between 99-100% and the ensemble FNR rate of 1.06%. Noteworthy is that the TNR and FPR in this iteration were 0.0, indicating no DDoS data was observed.

5.2.3 Comparison of SHARP results 98-02 and 50-50 test scenarios

Figures 16 and 17 visualise the ensemble anomaly score results of the 98-02 and 50-50 percentage ratio test datasets without the label results to compare the baseline performance metrics.

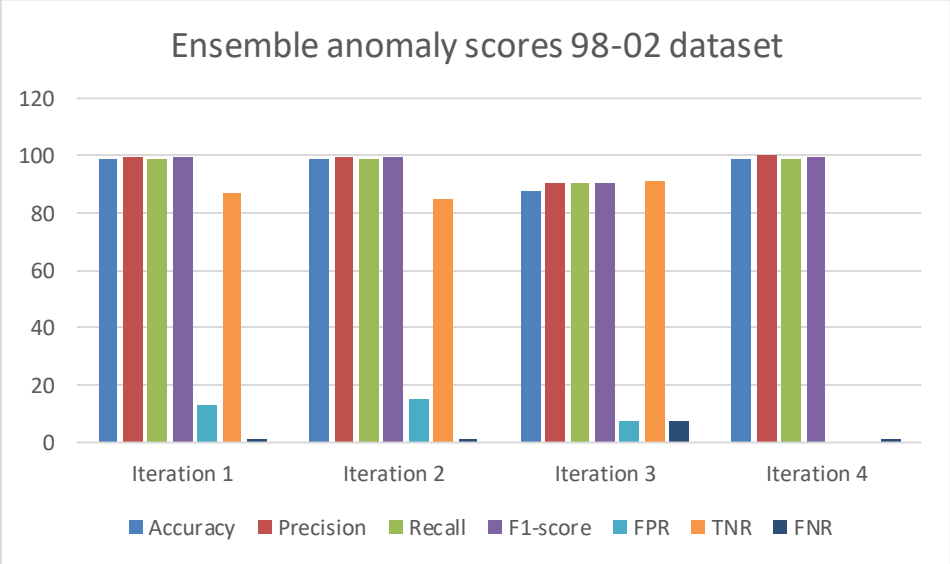

Figure 16. Overview ensemble anomaly scores of each iteration for the 98-02 percentage dataset.

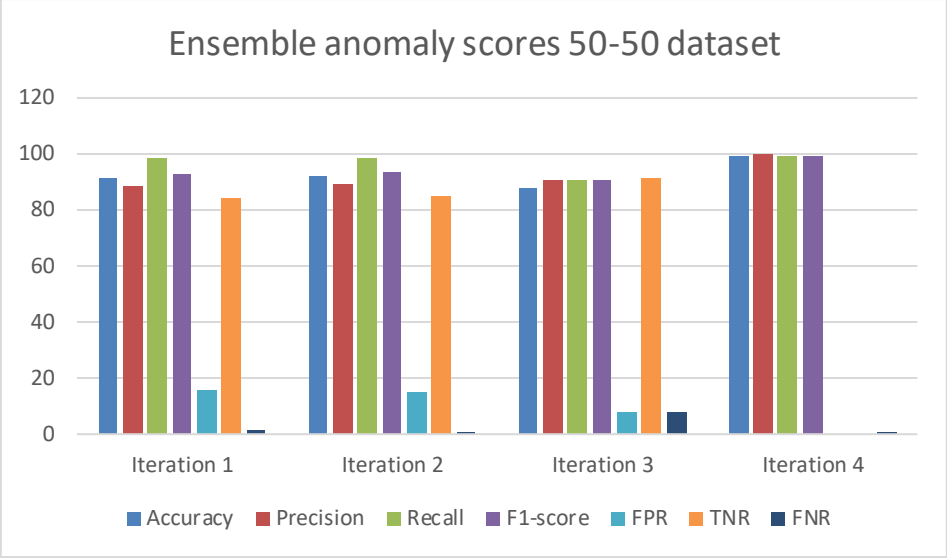

Figure 17. Overview ensemble anomaly scores of each iteration for the 50-50 percentage dataset.

The 98-02 and 50-50 percentage ratio test datasets were analysed across four iterations, revealing similar EXP4 performance. Both test scenarios share comparable results in areas including the advice vector values, rewards, estimated rewards and expert contribution. Consequently, the updated expert's weight values are identical for both test scenarios. Furthermore, no differences emerged between iterations three and four concerning the ensemble anomaly scores. However, significant differences in the first and second iterations concerning the ensemble anomaly scores were observed. In the first iteration, the ensemble anomaly scores for the 98-02 test scenario surpassed those of the 50-50 test scenario. Specifically, the accuracy was higher by 7.05%, the precision differed by 10.83%, the recall by 0.12%, the F1 score by 6.32%, and TNR by 2.79%. Moreover, the FPR of the 98-02 test scenario achieved a lower value than the 50-50 test scenario with a difference of 2.79%. Additionally, the FNR was also lower by 0.12%. The second iteration showed similar patterns in the differences in ensemble anomaly scores. Whereas the accuracy of the 98-02 test scenario surpassed the accuracy of the 50-50 test scenario by 6.64%, the precision achieved the most significant marginal difference with a percentage of 10.16%. While the recall of the 98-02 test scenario was slightly higher by 0.05%, the F1-score of the 50-50 test scenario, on the other hand, surpassed the F1-scores of the 98-02 test scenario by 5.94%. Furthermore, the TNR of the 50-50 test scenario showed a slight increase with a difference of 0.02%. Finally, the FPR and FNR of the 98-02 test scenario achieved lower values than the 50-50 test scenario, with FPR differing by 0.01% and FNR by 0.05%.

5.3 Discussion

The rapid development in the smart home industry and the lack of security focus motivated this study to contribute to the research fields of IoT and smart home security. This study examined the feasibility of integrating an ML-based ANIDS utilising UL and RL for dynamic environments, like smart home networks, to detect and respond to cyber-attacks. This subchapter presents a discussion centred around the three sub-questions that interpret literature findings and the results of the practical implementation of SHARP.

5.3.1 ML-based ANIDS using UE and RL for smart home networks

Central to the first sub-question were anomalies and anomaly detection foundations. Anomalies in terms of network security were defined as deviations from normal network behaviour. Regarding anomaly detection for smart home environments, ANIDS were commonly employed to detect cyber-attack anomalies in a smart home network. The literature identified supervised EL using supervised ML as an effective method for performing ML-based ANIDS activities within a smart home network. However, a gap in existing literature was observed regarding the utilisation and practical implementation of UE for ML-based ANIDS activities.

Furthermore, the literature highlighted that vulnerabilities inherent to smart home environments are driven by the heterogeneity of smart home components and the absence of security measures, risking the smart homeowners' privacy, financial stability and physical safety. Due to the landscape size of smart home environments and their dynamic network behaviour, the application of supervised EL utilising supervised ML to perform ML-based ANIDS activities is deemed challenging due to its inability to identify new cyber-attacks, inadaptability to the environment, and the lack of labelled data. Therefore, this study proposed combining UE and RL to tackle this issue.

Existing literature has shown the effectiveness of implementing the EXP4 algorithm to incorporate RL and dynamically enhance the performance of a single unsupervised classifier. Building upon this idea, this study introduced SHARP, an ML-based ANIDS that combines UE and EXP4. SHARP facilitates a dynamic weight-based decision-making system that leverages the strengths of individual unsupervised classifiers and optimises its UE based on the environmental state to produce an ensemble anomaly score that detects DDoS attacks in smart home networks.

5.3.2 SHARP's unsupervised classifiers

Four iterative cycles, each simulating a different scenario, were performed to evaluate SHARP. The results derived from the first and second iterations were used to compare SHARP's individual unsupervised classifiers' performance with the findings of existing studies, as this data contained actual DDoS anomalies in its test scenarios. Table 27 compares the performance of the three unsupervised classifiers, OCSVM, LOF, and iForest, used for the UE in SHARP against the findings of Das et al. (2020).

Table 27. Comparing SHARP's UE with existing unsupervised classifiers.

	Classifier	Accuracy	Precision	Recall	F1-score	FPR	TNR	FNR
Das et al., (2020)	OCSVM	86.3	86.6	78.0	83.0	7.5	-	-
	LOF	57.0	49.6	43.8	46.5	33.2	-	-
	iForest	89.3	89.0	85.5	87.2	7.9	-	-
This study	OCSVM-98-02	98.94	99.99	98.93	99.46	0.50	99.50	0.75
	OCSVM-50-50	98.44	98.09	98.80	98.44	1.92	98.08	1.2
	LOF-98-02	99.80	100.0	99.8	99.9	0.0	100.0	0.20
	LOF-50-50	99.98	100.0	99.96	99.97	0.0	100.0	0.04
	iForest-98-02	96.90	99.09	97.73	98.40	44.00	56.00	2.26
	iForest-50-50	75.31	67.54	97.44	79.78	46.82	53.18	2.56

This study's results indicate that novelty-based classifiers like OCSVM and LOF show superior results relative to iForest in both test scenarios. The results of this study indicate that the performance of LOF was superior to the other unsupervised classifiers across both test scenarios. This can be observed through its high precision, F1-score, and low FPR and FNR, suggesting that LOF could identify rare DDoS anomalies with the occurrence of 2% while also being able to identify benign traffic correctly. Despite LOF presenting excellent performance in this study, the findings of Das et al. (2020) show that it was the worst-performing model among OCSVM and iForest. Interestingly, both studies employ LOF using novelty-based detection.

Moreover, this study observed similar performance between OCSVM and LOF, contrasting with Das et al. (2020), which presents a significant difference between the two classifiers. Furthermore, it was evident that iForest showed subpar performance due to its high FPR values in both test scenarios, suggesting the model creates many false positives and experiences difficulties with accurately identifying DDoS attacks.

In contrast to this study, iForest outperformed the other models in the findings of Das et al. (2020), while both studies employed iForest using the outlier detection method. Das et al. (2020) used test datasets consisting of 22.544 rows of data, where approximately 33% were identified as DDoS and 67% were benign. This study used fewer data instances for its test set but included two test ratios. Albeit iForest's FPR and TNR scores, iForest demonstrated a high accuracy, precision, recall and F1-score for the 98-02 test scenario. However, iForest most likely showed a bias towards the majority class, as the results for the 98-02 test scenario significantly differed from those of the 50-50 test scenario.

While this study suggests that novelty-based classifiers outperform outlier-based classifiers when handling imbalanced and balanced datasets, comparing the results with Das et al.(2020) illustrated that this observation is not universally applicable, as factors including the train and test dataset ratio can affect the performance of the unsupervised classifiers.

5.3.3 SHARP's ensemble anomaly scores

The comparative analysis of SHARP's ensemble anomaly scores against existing supervised and UE leverages the first and second iteration findings, as these iterations represented actual DDoS anomalies within their test scenarios.

SHARP creates an ensemble anomaly score that harnesses each unsupervised classifier's strengths by incorporating a dynamic weighing system. The weighing system assigns a degree of importance to every expert, which is linked to an unsupervised classifier based on their performance evaluation. Experts who select proper actions based on the unsupervised classifier's evaluation are assigned greater weights, and their evaluation carries more importance during the computation of the collective ensemble anomaly score.

Table 28 compares SHARP's ensemble anomaly scores with the findings of Das et al. (2024) regarding DDoS anomaly detection using supervised ensemble and UE. In addition, the anomaly detection results utilising a single iForest unsupervised classifier with EXP4 of Tariq et al. (2022) are also demonstrated. Furthermore, Figure 18 presents a visualisation derived from the data in Table 28 that compares SHARP's ensemble anomaly scores against existing UE findings.

Table 28. Comparing SHARP's ensemble anomaly score with other UE performances.

	Method	**Dataset**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**TNR**	**FPR**	**FNR**
Das et al. (2024)	Supervised Ensemble	CICIDS2017	99.9	99.9	99.9	99.9	N/A	0.1	N/A
	UE	CICIDS2017	84.3	77.3	64.5	70.3	N/A	8.3	N/A
Tariq et al. (2022)	Un-supervised iForest with EXP4	MAGPIE	74.9	78.7	78.0	N/A	N/A	28.1	N/A
This study	UE	CIC IoT 2023 – 98-02	98.6	99.71	98.86	99.28	86.12	13.87	1.15
	UE	CIC IoT 2023 – 50-50	91.76	89.21	98.77	93.15	84.74	15.27	1.23

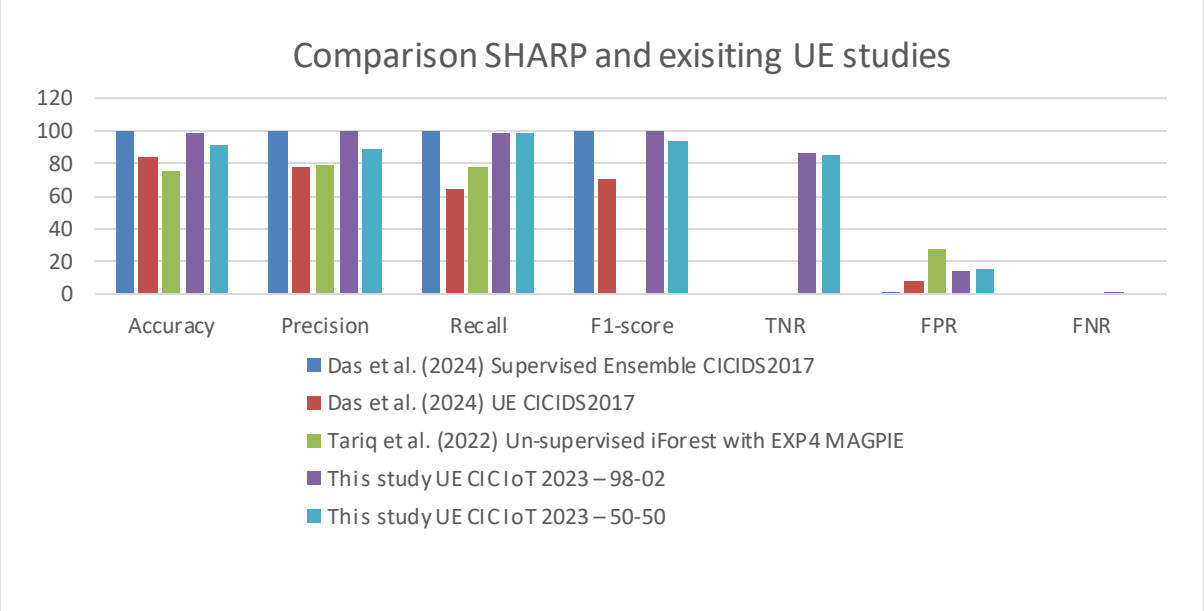

Figure 18. Comparison SHARP's ensemble anomaly scores with existing UE studies.

First and foremost, the comparison illustrates that supervised ensembles outperform UE. However, differences in performance between UE and single unsupervised classifiers can be observed among the studies.

Starting with the comparison between this study's UE and the UE findings of Das et al. (2024), the comparison reveals that SHARP's evaluation surpasses Das et al. (2024) with a difference in accuracy by 7.46%, difference in precision by 11.91%, difference in recall by 34.27% and the F1-score yielding a higher value by 22.85%, even though both studies employed similar train and test dataset ratios. However, SHARP's UE showed a worse FPR than the UE findings of Das et al. (2024), with SHARP yielding a higher FPR value of 6.97%. This finding suggests a correlation between SHARP's high FPR value and the performance of iForest, as the results show that OCSVM and LOF produce much lower FPR values. Given the results of only two iterations, SHARP's UE FPR may lower over time as the SHARP punishes unsupervised classifiers that perform poorly as the expert's weight is diminished, making their evaluation less important in the final ensemble anomaly score.

Furthermore, when SHARP's worst ensemble anomaly score (50-50 test scenario) is compared against the results of a single unsupervised iForest from Tariq et al. (2022), it is evident that SHARP's ensemble anomaly score achieved a higher accuracy with a difference of 16.8%, a higher precision by 10.51%, a higher recall by 20.77% and a significant reduction of FPR values by 12.83%.

The comparison of SHARP's ensemble anomaly scores with other existing UE models suggests that SHARP can identify DDoS attacks correctly and does not overlook DDoS attacks, as indicated by its high ensemble accuracy, precision, recall, and low FNR scores. Despite SHARP's excellent performance in terms of its ensemble accuracy, precision, recall, and FNR scores, the high FPR remains an issue that requires further optimisation to increase the reliability of SHARP's ensemble anomaly scores.

6 Conclusion

This study aimed to contribute to the IoT and smart home security field by investigating the feasibility and efficiency of an ML-based ANIDS employing UL and RL to detect and respond to cyber-attacks in a smart home network. To facilitate the investigation of the main research question, three sub-questions were established, all of which were examined by performing a comprehensive literature review.

The first sub-question dove into the nature of anomalies and explored various anomaly detection techniques used to detect cyber-attacks in a smart home network. A comprehensive literature review was conducted regarding anomaly detection, smart home components, and prevalent smart home network cyber-attacks. Literature analysis revealed that anomalies are irregularities that deviate from normal behaviour and can be categorised into point, contextual, or collective anomalies. Furthermore, literature highlighted the effectiveness of ML-based ANIDS for identifying network anomalies, as this technique excels in identifying unknown cyber-attacks. Despite the lack of consensus regarding the definition of a smart home observed across literature, this study defined the smart home using the common main components observed across various literature articles: IoT/smart devices, smart home gateway, communication protocols, the Cloud and applications. The literature review highlighted the complexity and heterogeneity of smart home network topologies, structures and communication protocols. Due to the scale of smart home environments and insufficient security measures in their components, the literature revealed that smart home networks are susceptible to cyber-attacks that compromise the owner's privacy, financial stability, physical safety and emotional state. The literature identified three active cyber-attack categories that exploit smart home networks: DoS/DDoS, routing attacks, and others. This study focused on detecting DDoS anomalies.

The second sub-question examined how an ML-based ANIDS can utilise UL and RL to enhance the detection of anomalies in a smart home network. This study focused on establishing an ML-based ANIDS that identifies DDoS anomalies in a smart home network. The literature review indicated that while supervised EL excel in performance in detecting anomalies in smart home environments, this method lacked adaptability. Consequently, ERL methods addressed this gap by facilitating adaptability and continuous optimisation. However, literature analysis revealed that ERL relies on labelled datasets, which poses significant challenges for the practical deployment within a smart home environment due to the scarcity of labelled data.

This study identified the gap in existing literature regarding the utilisation and practical deployment of UE in combination with RL to perform ML-based ANIDS activities. Based on the findings of the literature, the study implemented three unsupervised classifiers- OCSVM, LOF, and iForest - to establish the UE, as they demonstrated high performance in DDoS anomaly detection. Furthermore, incorporating RL techniques, specifically the EXP4 algorithm, has shown potential to enhance the performance of the single unsupervised classifiers within the UE by facilitating a dynamic weight-based decision-making system. The insights derived from the literature review led to the introduction of SHARP, an ML-based ANIDS that incorporates UE and EXP4 to facilitate the integration of UL and RL. SHARP implements a dynamic weighing decision framework that leverages the strength of the UE by iteratively evaluating the significance of each unsupervised classifier against the environmental state using the EXP4 algorithm. The SHARP framework produces a collective ensemble anomaly score that identifies DDoS anomalies within a smart home network, while accommodating unlabelled data and facilitating adaptability to the dynamic smart home network traffic.

The third sub-question examined the effectiveness of SHARP, which utilises UL and RL. The practical implementation of SHARP provided promising results, with the ensemble anomaly scores achieving higher performance than existing UE methods. SHARP's novelty-based classifiers excelled at identifying DDoS data correctly while mitigating the chance of overlooking them. This is evident from the ensemble performance metrics, which showed a 7.46% increase in accuracy, 11.91% increase in precision, 34.27% increase in recall, 22.85% increase in F1-score, and a low FNR of less than 2% compared to prior studies. In addition, the dynamic weighing system allows individual unsupervised classifiers to cover each other's weaknesses when establishing the ensemble anomaly score, therefore continuously optimising the UE over time.

To conclude, this study sheds light on how an ML-based ANIDS can be employed using UL and RL to detect and respond to cyber-attacks in a smart home network. The introduction and implementation of SHARP offered the foundations of a dynamic weighing system that incorporates UE and EXP4 and prioritises the contributions of unsupervised classifiers based on their performance to effectively enhance the detection of DDoS anomalies in a smart home network.

6.1 Limitations and recommendations

While the study presents promising findings in anomaly detection in smart home networks utilising UE and RL, theoretical and practical limitations must be acknowledged. The current ensemble anomaly score, where weighted anomaly scores are summed using Equation 7, makes SHARP incapable of properly incorporating extreme FPR or FNR scores of individual unsupervised classifiers. This was observed in SHARP's ensemble FPR score, where solely iForest generated a disproportionate FPR value compared to the FPR values of OCSVM and LOF. Therefore, it is recommended that SHARP compute the mean of FPR and FNR and compute the average to a predetermined baseline to prevent disproportionate ensemble scores.

Several practical limitations were identified during the establishment of SHARP. Starting with the absence of a live smart home network environment. Due to the scope of this study, the practical implementation relied on static variables to demonstrate various test scenarios. SHARP's behaviour and ensemble anomaly scores were not examined in a live smart home network environment. When implementing SHARP in a live smart home network, the actions should be linked to executable activities, and the network's status impact should be observed in real time to ensure SHARP's effectiveness. In addition, SHARP currently lacks the ability to trigger a response mechanism due to the absence of a live smart home network, which should be integrated into a real smart home network. Whereas SHARP's ensemble anomaly scores showed promising performance, it was evaluated using controlled smart home network datasets. Thus, SHARP's performance was not verified in a live smart home network, suggesting its resilience against real-world DDoS attacks may deviate from this study's results.

Next, the train and test dataset volume was reduced due to computation resource constraints, potentially affecting the UE performance. Furthermore, SHARP was evaluated using four iterative cycles. The first two contained DDoS test scenarios, the third used predetermined static values to portray the scenario of limiting the network traffic, and the fourth iteration observed SHARP's behaviour when solely exposed to benign traffic. The four iterations only encapsulated a handful of action-status impact scenarios that demonstrated SHARP's behaviour for short-term iterations. However, the number of test scenarios that use different action-status impact scenarios is recommended to be investigated further to assess SHARP's ensemble anomaly scores.

The final limitation is regarding the implementation of the unsupervised classifiers within the UE used for the SHARP framework. SHARP utilised two novelty-based anomaly detection methods and one outlier-based anomaly detection method. While this composition of the UE presented compelling results, it is recommended to experiment with other novelty and outlier-based classifiers and evaluate the ensemble anomaly scores with different ratios of novelty- and outlier-based classifiers.

6.2 Future work

Building upon the proposed SHARP framework, several prospective research topics can be of interest to enhance SHARP. Firstly, evaluating SHARP in a dynamic environment would provide valuable insight into its performance in a live smart home network. Additionally, by integrating SHARP into a live smart home network, more iterations and action-status impact combinations can be observed, providing more insight into SHARP's effectiveness.

Next, experimenting with various unsupervised classifiers, training and testing on larger datasets and using different dataset ratios may refine SHARP's UE and enhance the reliability of the ensemble anomaly scores. Finally, SHARP currently focuses on the detection of DDoS anomalies. However, expanding SHARP's resilience against other prevalent active cyber-attacks, such as routing or other attacks, in smart home networks and evaluating its performance, is another topic of interest that can contribute to its advancement.

References

Abdulla, N. N., & Hasoun, R. K. (2022). Review of Detection Denial of Service Attacks using Machine Learning through Ensemble Learning. *Iraqi Journal for Computers and Informatics*, *48*(1), 13-20. https://doi.org/10.25195/ijci.v48i1.349

Agyemang, E. F. (2024). Anomaly detection using unsupervised machine learning algorithms: A simulation study. *Scientific African, 16,* Article e02386. https://doi.org/10.1016/j.sciaf.2024.e02386

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, *60*, 19-31. https://doi.org/10.1016/j.jnca.2015.11.016

Alabadi, M., & Çelik, Y. (2020). Anomaly Detection for Cyber-Security Based on Convolution Neural Network : A survey. *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey*, 1-14. https://doi.org/10.1109/hora49412.2020.9152899

Ali, B., & Awad, A. I. (2018). Cyber and Physical Security Vulnerability Assessment for IoT-Based smart Homes. *Sensors*, *18*(3), 817. https://doi.org/10.3390/s18030817

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A. Q., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(53). 1-74. https://doi.org/10.1186/s40537-021-00444-8

Anthi, E., Williams, L., Slowinska, M., Theodorakopoulos, G., & Burnap, P. (2019). A supervised intrusion detection system for smart home IoT devices. *IEEE Internet of Things Journal*, *6*(5), 9042-9053. https://doi.org/10.1109/jiot.2019.2926365

Anwar, R. W., Zainal, A., Abdullah, T., & Iqbal, S. (2020). Security Threats and Challenges to IoT and its Applications: A Review. *2020 Fifth International Conference on Fog*

and Mobile Edge Computing (FMEC), Paris, France, 301-305. https://doi.org/10.1109/fmec49853.2020.9144832

Araya, J. I. I., & Rifà-Pous, H. (2023). Anomaly-based cyberattacks detection for smart homes: A systematic literature review. *Internet of Things*, *22*, Article 100792. https://doi.org/10.1016/j.iot.2023.100792

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. (2014). Network Anomaly Detection: Methods, systems and tools. *IEEE Communications Surveys and Tutorials*, *16*(1), 303-336. https://doi.org/10.1109/surv.2013.052213.00046

Çakir, S., Toklu, S., & Yalçın, N. (2020). RPL attack detection and prevention in the internet of things networks using a GRU based deep learning. *IEEE Access*, *8*, 183678-183689. https://doi.org/10.1109/access.2020.3029191

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection. *ACM Computing Surveys*, *41*(3), 1-58. https://doi.org/10.1145/1541880.1541882

Chatterjee, A., & Ahmed, B. S. (2022). IoT anomaly detection methods and applications: A survey. *Internet of Things, 19*, Article 100568. https://doi.org/10.1016/j.iot.2022.100568

Chen, D., Zhuang, Y., Huai, J., Sun, X., Yang, X., Javed, M. A., Brown, J., Sheng, Z., & Thompson, J. (2021). Coexistence and Interference Mitigation for WPANs and WLANs From Traditional Approaches to Deep Learning: A review. *IEEE Sensors Journal*, *21*(22), 25561-25589. https://doi.org/10.1109/jsen.2021.3117399

Das, S., Ashrafuzzaman, M., Sheldon, F. T., & Shiva, S. (2024). Ensembling supervised and unsupervised machine learning algorithms for detecting distributed denial of service attacks. *Algorithms*, *17*(3), 99. https://doi.org/10.3390/a17030099

Das, S., Venugopal, D., & Shiva, S. (2020). A holistic approach for detecting DDOS attacks by using ensemble unsupervised machine learning. In: Arai, K., Kapoor, S., Bhatia, R.

(Eds.), *Advances in Intelligent Systems and Computing* : *Vol 1130*. *Advances in Information and Communication* (pp.721-738). Springer, Cham. https://doi.org/10.1007/978-3-030-39442-4_53

Denning, T., Kohno, T., & Levy, H. M. (2013). Computer security and the modern home. *Communications of the ACM*, *56*(1), 94-103. https://doi.org/10.1145/2398356.2398377

Fahim, M., & Sillitti, A. (2019). Anomaly Detection, Analysis and Prediction Techniques in IoT Environment: A Systematic Literature review. *IEEE Access*, *7*, 81664-81681. https://doi.org/10.1109/access.2019.2921912

Gai, A., Azam, S., Shanmugam, B., Jonkman, M., & De Boer, F. (2018). Categorisation of security threats for smart home appliances. *2018 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India*. https://doi.org/10.1109/iccci.2018.8441213

García-Teodoro, P., Díaz-Verdejo, J. E., Maciá-Fernández, G., & Vazquez, E. A. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, *28*(1–2), 18-28. https://doi.org/10.1016/j.cose.2008.08.003

Geneiatakis, D., Kounelis, I., Neisse, R., Nai-Fovino, I., Steri, G., & Baldini, G. (2017). Security and privacy issues for an IoT based smart home. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, *Opatija, Croatia*. https://doi.org/10.23919/mipro.2017.7973622

Gerodimos, A., Maglaras, L., Ferrag, M. A., Ayres, N., & Kantzavelou, I. (2023). IoT: Communication protocols and security threats. *Internet of Things and Cyber-Physical Systems*, *3*, 1-13. https://doi.org/10.1016/j.iotcps.2022.12.003

Grabowski, M., & Dziwoki, G. (2009). The IEEE wireless standards as an infrastructure of smart home network. In: Kwiecień, A., Gaj, P., Stera, P. (Eds.), *Communications in*

Computer and Information Science: Vol. 39. *Computer Networks* (pp. 302-309). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-02671-3_35

Gunawan, T. S., Yaldi, I. R. H., Kartiwi, M., Ismail, N., Za'bah, N. F., Mansor, H., & Nordin, A. N. (2017). Prototype Design of Smart Home System using Internet of Things. *Indonesian Journal of Electrical Engineering and Computer Science*, *7*(1), 107-115. https://doi.org/10.11591/ijeecs.v7.i1.pp107-115

Gupta, C., Johri, I., Srinivasan, K., Hu, Y., Qaisar, S. M., & Huang, K. (2022). A Systematic Review on machine learning and deep learning models for electronic information security in mobile networks. *Sensors*, *22*(5), 2017. https://doi.org/10.3390/s22052017

Gupta, H., & Sharma, S. (2021). Security Challenges in Adopting Internet of Things for Smart Network. *2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India,* 761-765. https://doi.org/10.1109/csnt51715.2021.9509698

HaddadPajouh, H., Dehghantanha, A., Parizi, R. M., & Aledhari, M. (2021). A survey on internet of things security: Requirements, challenges, and solutions. *Internet of Things*, *14*, Article 100129. https://doi.org/10.1016/j.iot.2019.100129

Hammi, B., Zeadally, S., Khatoun, R., & Nebhen, J. (2022). Survey on smart homes: Vulnerabilities, risks, and countermeasures. *Computers & Security*, *117*, Article 102677. https://doi.org/10.1016/j.cose.2022.102677

Hafeez, A., Kandil, N., Al-Omar, B., Landolsi, T., & Al-Ali, A. R. (2014). Smart Home Area Networks Protocols within the Smart Grid Context. *Journal of Communications*, *9*(9), 665-671. https://doi.org/10.12720/jcm.9.9.665-671

Holguin, I., & Errapotu, S. M. (2023). Smart Home IoT Communication Protocols and Advances in their Security and Interoperability. *2023 7th Cyber Security in*

Networking Conference (CSNet), Montreal, Canada, 208-211. https://doi.org/10.1109/csnet59123.2023.10339739

Jha, K., Jha, R. K., Jha, A. K., Hassan, M. a. M., Yadav, S. V., & Mahesh, T. R. (2021). A Brief Comparison On Machine Learning Algorithms Based On Various Applications: A Comprehensive Survey. *2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bangalore, India,* 1-5. https://doi.org/10.1109/csitss54238.2021.9683524

Jyothsna, V. (2011). A Review of Anomaly based Intrusion Detection Systems. *International Journal of Computer Applications*, *28*(7), 26-35. https://doi.org/10.5120/3399-4730

Kamel, S. O. M., & Hegazi, N. H. (2018). A Proposed Model of IoT Security Management System Based on A study of Internet of Things (IoT) Security. *International Journal of Scientific and Engineering Research*, *9*(9), 1227-1244.

Khampuengson, T., & Wang, W. (2022). Deep Reinforcement Learning Ensemble for detecting anomaly in telemetry water level data. *Water, 14*(16), 2492. https://doi.org/10.3390/w14162492

Khawla, M., & Mazri, T. (2018). A Survey on the Security of Smart Homes. *ICSDE'18: Proceedings of the 2nd International Conference on Smart Digital Environment, New York, USA*, 81-87. https://doi.org/10.1145/3289100.3289114

Kim, S. (2022). Enabling WLAN and WPAN coexistence via Cross-Technology communication. *Sensors*, *22*(3), 707. https://doi.org/10.3390/s22030707

Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. E. (2006). Data Preprocessing for Superviced Learning. *International Journal of Computer Science*, *1*(1), 111-117.

Lee, C., Zappaterra, L., Choi, K., & Choi, H. (2014). Securing smart home: Technologies, security challenges, and security requirements. *2014 IEEE Conference on*

Communications and Network Security, San Francisco, USA, 67-72. https://doi.org/10.1109/cns.2014.6997467

Lesch, V., Züfle, M., Bauer, A., Iffländer, L., Krupitzer, C., & Kounev, S. (2023). A literature review of IoT and CPS—What they are, and what they are not. *Journal of Systems and Software*, *200*, Article 111631. https://doi.org/10.1016/j.jss.2023.111631

Liao, J., Teo, S. G., Kundu, P. P., & Truong-Huu, T. (2021). ENAD: An Ensemble Framework for Unsupervised Network Anomaly Detection. *2021 IEEE International Conference on Cyber Security and Resilience (CSR), Rhodes, Greece,* 81-88. https://doi.org/10.1109/csr51186.2021.9527982

Liu, B., Blancaflor, E., & Abisado, M. (2023). Application of Network Security Threat Detection in The Smart Home. *2023 3rd International Symposium on Computer Technology and Information Science (ISCTIS), Chengdu, China,* 19-24. https://doi.org/10.1109/isctis58954.2023.10213034

Lundin, E., & Jonsson, E. (2000). Anomaly-based intrusion detection: privacy concerns and other problems. *Computer Networks*, *34*(4), 623-640. https://doi.org/10.1016/s1389-1286(00)00134-1

Mahesh, B. (2020). Machine Learning Algorithms - A Review. *International Journal of Science and Research (IJSR)*, *9*(1), 381-386. https://doi.org/10.21275/ART20203995

Malhotra, M. (2023, October 23). *Creating a smaller dataset for CICIOT2023*. Kaggle. https://www.kaggle.com/code/madhavmalhotra/creating-a-smaller-dataset-for-ciciot2023/output. Last accessed 27 March 2025.

Mane, D., Sangve, S., Upadhye, G., Kandhare, S., Mohole, S., Sonar, S., & Tupare, S. (2022). Detection of Anomaly using Machine Learning: A Comprehensive Survey. *International Journal Emerging Technology and Advanced Engineering (IJETAE)*, *12*(11), 134-152. https://doi.org/10.46338/ijetae1122_15

Manhas, J., & Kotwal, S. (2021). Implementation of intrusion detection system for internet of things using machine learning techniques. In: Giri, K.J., Parah, S.A., Bashir, R., Muhammad, K. (Eds.), *Algorithms for Intelligent Systems. Multimedia Security (pp.217-237).* Springer, Singapore. https://doi.org/10.1007/978-981-15-8711-5_11

Manimurugan, S., Almutairi, S., Aborokbah, M., Chilamkurti, N., Ganesan, S., & Patan, R. (2020). Effective attack detection in internet of medical things smart environment using a deep belief neural network. *IEEE Access*, *8*, 77396–77404. https://doi.org/10.1109/access.2020.2986013

Mansour, M., Gamal, A., Ahmed, A. I., Said, L. A., Elbaz, A., Herencsár, N., & Soltan, A. (2023). Internet of Things: A comprehensive overview on protocols, architectures, technologies, simulation tools, and future directions. *Energies*, *16*(8), 3465. https://doi.org/10.3390/en16083465

Marksteiner, S., Jiménez, V. J. E., Valiant, H., & Zeiner, H. (2017). An overview of wireless IoT protocol security in the smart home domain. *2017 Internet of Things Business Models, Users, and Networks, Copenhagen, Denmark*, 108. https://doi.org/10.1109/ctte.2017.8260940

Mendes, T. D. P., Godina, R., Rodrigues, E. M. G., Matias, J. C. O., & Catalão, J. P. (2015). Smart Home Communication Technologies and Applications: Wireless Protocol Assessment for home Area Network Resources. *Energies*, *8*(7), 7279–7311. https://doi.org/10.3390/en8077279

Mishra, A. K., Tripathy, A. K., Puthal, D., & Yang, L. T. (2019). Analytical model for Sybil attack phases in Internet of things. *IEEE Internet of Things Journal*, *6*(1), 379–387. https://doi.org/10.1109/jiot.2018.2843769

Mocrii, D., Chen, Y., & Musilek, P. (2018). IoT-based smart homes: A review of system architecture, software, communications, privacy and security. *Internet of Things*, *1–2*, 81–98. https://doi.org/10.1016/j.iot.2018.08.009

Mohanta, B. K., Jena, D., Satapathy, U., & Patnaik, S. (2020). Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology. *Internet of Things*, *11*, Article 100227. https://doi.org/10.1016/j.iot.2020.100227

Nassif, A. B., Talib, M. A., Nasir, Q., & Dakalbab, F. (2021). Machine Learning for Anomaly Detection: A Systematic Review. *IEEE Access*, *9*, 78658–78700. https://doi.org/10.1109/access.2021.3083060

Neto, E. C., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIOT2023: A real-time dataset and benchmark for large-scale attacks in IOT environment. *Sensors*, *23*(13), 5941. https://doi.org/10.3390/s23135941

Orfanos, V. A., Kaminaris, S. D., Papageorgas, P., Piromalis, D., & Kandris, D. (2023). A comprehensive review of IoT networking technologies for smart home automation applications. *Journal of Sensor and Actuator Networks*, *12*(2), 30. https://doi.org/10.3390/jsan12020030

Ozer, E., & Iskefiyeli, M. (2017). Detection of DDoS attack via deep packet analysis in real time systems. *2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey,* 1137-1140. https://doi.org/10.1109/ubmk.2017.8093526

Patcha, A., & Park, J. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, *51*(12), 3448–3470. https://doi.org/10.1016/j.comnet.2007.02.001

Rabbani, M., Wang, Y., Khoshkangini, R., Jelodar, H., Zhao, R., Ahmadi, S. B. B., & Ayobi, S. (2021). A review on Machine learning Approaches for network Malicious Behavior

Detection in Emerging Technologies. *Entropy*, *23*(5), 529. https://doi.org/10.3390/e23050529

Rokhade, A. A., Vithapnavar, A. S., Amruth, S., Shettigar, A. J. V., Supreetha, S., & Honnavalli, P. B. (2023). Anomaly Detection for IoT Security: Comprehensive Survey. *2023 International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS), Bangalore, India,* 84-92. https://doi.org/10.1109/icaecis58353.2023.10170192

Sakong, W., & Kim, W. (2023). Adaptive Ddos Response Policy by Reinforcement Learning with an Anomality Reward Function. *SSRN*. https://doi.org/10.2139/ssrn.4605933

Sarwar, N., Bajwa, I. S., Hussain, M. Z., Ibrahim, M. A., & Saleem, K. (2023). IoT network anomaly detection in smart homes using machine learning. *IEEE Access*, *11*, 119462–119480. https://doi.org/10.1109/access.2023.3325929

Saha, S., Priyoti, A. T., Sharma, A., & Haque, A. (2022). Towards an optimized ensemble feature selection for DDOS detection using both supervised and unsupervised method. *Sensors, 22*(23), 9144. https://doi.org/10.3390/s22239144

Saxena, U., Sodhi, J. S., & Singh, Y. (2017). Analysis of security attacks in a smart home networks. *2017 7th International Conference on Cloud Computing, Data Science & Engineering – Confluence, Noida, India,* 431-436. https://doi.org/10.1109/confluence.2017.7943189

Sharma, B., Sharma, L., Lal, C., & Roy, S. (2023). Anomaly based network intrusion detection for IoT attacks using deep learning technique. *Computers & Electrical Engineering*, *107*, Article 108626. https://doi.org/10.1016/j.compeleceng.2023.108626

Shouran, Z., Ashari, A., & Priyambodo, T. K. (2019). Internet of Things (IoT) of smart home: privacy and security. *International Journal of Computer Applications*, *182*(39), 3-8. https://doi.org/10.5120/ijca2019918450

Silverio-Fernández, M. A., Renukappa, S., & Suresh, S. (2018). What is a smart device? - a conceptualisation within the paradigm of the internet of things. *Visualization in Engineering*, *6*, Article 3. https://doi.org/10.1186/s40327-018-0063-8

Singh, M. M., & Kane, N. (2022). Outlier Detection using Ensemble Learning. *2022 6th International Conference on Information Technology (InCIT), Nonthaburi, Thailand,* 234-239. https://doi.org/10.1109/incit56086.2022.10067524

Sivapriyan, R., Sushmitha, S., Pooja, K., & Sakshi, N. (2021). Analysis of Security Challenges and Issues in IoT Enabled Smart Homes. *2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bangalore, India,* 1-6. https://doi.org/10.1109/csitss54238.2021.9683324

Slivkins, A. (2019). Introduction to Multi-Armed Bandits. *Now Foundations and Trends*, *12*(1-2), 1-286. https://doi.org/10.1561/2200000068

Song, Y., Suganthan, P. N., Pedrycz, W., Ou, J., He, Y., Chen, Y., & Wu, Y. (2023). Ensemble reinforcement learning: A survey. *Applied Soft Computing*, *149 (Part A)*, 110975. https://doi.org/10.1016/j.asoc.2023.110975

Stojkoska, B. R., & Trivodaliev, K. (2017). A review of Internet of Things for smart home: Challenges and solutions. *Journal of Cleaner Production*, *140*, 1454-1464. https://doi.org/10.1016/j.jclepro.2016.10.006

Tao, M., Zuo, J., Liu, Z., Castiglione, A., & Palmieri, F. (2018). Multi-layer cloud architectural model and ontology-based security service framework for IoT-based smart homes. *Future Generation Computer Systems*, *78*, 1040-1051. https://doi.org/10.1016/j.future.2016.11.011

Tariq, Z. U. A., Baccour, E., Erbad, A., Guizani, M., & Hamdi, M. (2022). Network Intrusion Detection for Smart Infrastructure using Multi-armed Bandit based Reinforcement

Learning in Adversarial Environment. *2022 International Conference on Cyber Warfare and Security (ICCWS), Islamabad, Pakistan,* 75-82. https://doi.org/10.1109/iccws56285.2022.9998440

Touqeer, H., Zaman, S., Amin, R., Hussain, M., Al-Turjman, F., & Bilal, M. (2021). Smart home security: challenges, issues and solutions at different IoT layers. *The Journal of Supercomputing*, *77*(12), 14053–14089. https://doi.org/10.1007/s11227-021-03825-1

Tournier, J., Lesueur, F., Mouël, F. L., Guyon, L., & Ben-Hassine, H. (2021). A survey of IoT protocols and their security issues through the lens of a generic IoT stack. *Internet of Things*, *16*, 100264. https://doi.org/10.1016/j.iot.2020.100264

Victor, P., Lashkari, A. H., Lu, R., Sasi, T., Xiong, P., & Iqbal, S. (2023). IoT malware: An attribute-based taxonomy, detection mechanisms and challenges. *Peer-to-Peer Networking and Applications*, *16*(3), 1380–1431. https://doi.org/10.1007/s12083-023-01478-w

Yahya, & Mohammad, M. B. (2023). The challenges of smart environments and future trends supports by the internet of things. *International Journal of Advanced Multidisciplinary Research and Studies*. *3*(3), 528-534. https://doi.org/10.62225/2583049X

Yang, C., Mistretta, E., Chaychian, S., Siau, J. (2017). Smart Home System Network Architecture. In: Hu, J., Leung, V., Yang, K., Zhang, Y., Gao, J., Yang, S. (Eds.), *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering: Vol 175*. *Smart Grid Inspired Future Technologies* (pp. 174-183). Springer, Cham. https://doi.org/10.1007/978-3-319-47729-9_18

Zhang, W., Yang, Q., & Geng, Y. (2009). A Survey of Anomaly Detection Methods in Networks. *2009 International Symposium on Computer Network and Multimedia Technology, Wuhan, China,* 1-3. https://doi.org/10.1109/cnmt.2009.5374676

Zhou, C., Wen-Hui, H., & Zhao, X. (2013). Study on architecture of smart home management system and key devices. *Proceedings of 2013 3rd International Conference on Computer Science and Network Technology, Dalian, China,* 1255-1258. https://doi.org/10.1109/iccsnt.2013.6967330