ABSTRACT Recently, the number of Internet of Things (IoT) networks has been grown exponentially, which results in more data sharing between devices without appropriate security mechanisms. Since huge data management is involved, maintaining the time constraints between the devices in IoT networks is another significant issue. To address these issues, an intelligent intrusion detection system has been adapted to recognize or predict a cyber-attack using Elite Machine Learning algorithms (EML), and a lightweight protocol is used to manage the time-constrained issue. The experimental analysis of work is done on a testbed setup with the hardware and sensors connected using a lightweight Message Queue Telemetry Transport (MQTT) protocol. This comprises three parts: (i) collection of data with the help of a sensor for three different scenarios called SEN-MQTTSET; (ii) multi-context feature generation using an ensemble statistical multi-view cascade feature generation algorithm from the SEN-MQTTSET dataset; and (iii) evaluating the dataset using ML algorithms. The SEN-MQTTSET dataset has been created from the three scenarios, such as normal, attack on a subscriber, and attack on a broker. The multi-context feature is generated from the raw dataset using an ensemble statistical multi-view cascade feature generation algorithm. The EML is proposed to select the best model for intrusion detection among ML algorithms such as Logistic Regression, K-Nearest Neighbour, Random Forest, Naive Bias, Support Vector Machine, Gradient Boosting, and Decision Tree by the performance metrics such as accuracy, prediction time, F1-score, and others. The proposed dataset is validated and the accuracy is found to be above 99% for the considered system model. Different quality parameters have been carried out for legitimate and attack traffic features to calculate the delay between the IoT-MQTT network.

INDEX TERMS IoT-MQTT Networks, DoS, Multi-Cascade Features, Machine Learning

I. INTRODUCTION

THE Internet of Things (IoT) is referred to as where the sensor and actuators can transfer the data over the internet with the help of a processor [1]. As per Statista reports, there will be 75 billion IoT devices will be connected together in 2025 [2]. Some of the IoT applications are interconnected vehicles, intelligent buildings, smart cities, retail environments, etc., which are vulnerable to attacks. Hence, it needs security improvements such as secure booting, access control, and device authentication [3], [4]. Many IoT communication are based on hypertext transfer protocol (HTTP) protocol, but it uses large network resources and latency. Alternatively, Message Queue Telemetry Transport (MQTT) protocol uses less payload overhead packets with different quality of service (QoS) and uses minimal network resources, overcoming problems with the HTTP protocol [5]. IBM created and released an open-source protocol called MQTT for publishing/subscribing messaging over client-server transport. The features of MQTT are used in restricted communications environments for the machine to machine (M2M) and IoT applications where a low code footprint is needed or low network bandwidth is available. Due to its small overhead packet in MQTT is considered as a lightweight and binary protocol for sending data over the wire compared to other traditional protocols like HTTP. The publish/subscribe pattern (also known as the "pub/sub") offers an alternative to the
TABLE 1. List of Acronyms

| Sl.No | Acronym | Definition |
|-------|---------|------------|
| 1     | IoT     | Internet of Things |
| 2     | HTTP    | Hypertext Transfer Protocol |
| 3     | MQTT    | Message Queue Telemetry Transport |
| 4     | M2M     | Machine to Machine |
| 5     | QoS     | Quality of Service |
| 6     | WSN     | Wireless Sensor Network |
| 7     | MQTT-SN | Message Queue Telemetry Transport Sensor Network |
| 8     | TLS     | Transport Layer Security |
| 9     | DoS     | Denial of Service |
| 10    | BCF     | Basic Connect Flooding |
| 11    | TCP     | Transmission Control Packet |
| 12    | LR      | Logistic Regression |
| 13    | KNN     | K Nearest Neighbour |
| 14    | SVM     | Support Vector Machine |
| 11    | NB      | Naive Bias |
| 12    | RF      | Random Forest |
| 13    | DT      | Decision Tree |
| 14    | GB      | Gradient Boosting |
| 15    | CSV     | Comma Separated Values |
| 16    | DDOS    | Distributed Denial of Service |
| 17    | ROC     | Receiver Operating Characteristic |
| 18    | ACC     | Accuracy |
| 19    | ADR     | Attack Detection Rate |
| 20    | APV     | Attack Predictive Value |
| 21    | NPV     | Normal Predictive Value |
| 22    | MCC     | Mathews Correlation Coefficient |
| 23    | FAR     | False Alarm Rate |

conventional client-server architecture. In the client-server model, a client communicates directly with an endpoint.

Figure 1 shows the general architecture of MQTT (pub/sub model). The pub/sub paradigm separates the user who sends a message (the publisher) and the client receiving the message (the subscriber). The pub/sub paradigm does not have any direct contact and no knowledge about the co-existence of each other. The third component, named the MQTT broker, relates the pub/sub to filtering and distributing all incoming messages accurately to subscribers. The MQTT control packet frame structure consists of a fixed header (mostly available in all MQTT control packets), a variable header (available in some MQTT control packets), and a payload (available in some MQTT control packets). The data transmission using the MQTT protocol can be done with the topic name and QoS levels. There are three QoS levels: (i) QoS 0 is responsible for only one delivery, (ii) QoS 1 is responsible for at least one delivery, and (iii) QoS 2 is responsible for exactly-once delivery [6]. The rest of this paper is organized as follows: Some of the recent related works are presented in Section II. Section III describes the architecture setup and the proposed algorithms by machine learning (ML) techniques. Section IV shows the experimental testbed with results and discussions. Section V concludes this paper.

II. RELATED WORKS

A detailed [7] taxonomy of features in various MQTT protocols has been compared with other M2M protocols which are useful for end-users in choosing an appropriate broker or client suitable for implementation. In [8] describes the publish and subscribe protocol using MQTT-S, which is the open-source secured MQTT protocol extension adopted for wireless sensor networks (WSN). The MQTT-S architecture consists of two components, namely MQTT-S client and an MQTT-S gateway. The MQTT-S client enables the pub/sub of messages from the sensors to actuators. The MQTT-S gateway has two separate gateways, namely transparent and aggregating. The transparent gateway has many MQTT connections between the gateway and the brokers with MQTT-S clients linked to the gateway. A single MQTT connection to the broker with MQTT-S clients is referred to as an aggregating gateway. An MQTT-S client is only used to send messages in QoS 0, and it does not allow the broker to send messages to clients in QoS 1 or QoS 2.

The packet collection and analysis method [9] is used to measure the network environment and analyze the message loss and end-to-end delay for different QoS levels (QoS 0, QoS 1, and QoS 2) and payload sizes. The network experiments were done with wired and wireless connectivity between the pub/sub and the broker for comparison. The correlation coefficient measures the message loss and end-to-end delay in different QoS levels of the MQTT protocol. Smart home automation is implemented [10] using the ESP8266 controller and the MQTT protocol. The sensors and actuators such as light-dependent resistor (LDR), light-emitting diode (LED), and buzzers are connected to the ESP8266 with the Mosquito [11] MQTT broker. The intensity of light is measured by using an LDR sensor, and the data is sent to an MQTT broker. The actuator’s LED or buzzer can then subscribe to the data from the broker and act accordingly. All the data gathered from sensors and actuators is also sent to a cloud-based platform to aggregate, analyze, and visualize the data. A customized graphical user interface (GUI) was designed to monitor sensor data and control the devices remotely.

Table 2 elaborates the recent works related to intrusion detection system (IDS) on IoT networks. The MQTT protocol is used to obtain the status of crude oil production with a
two-step approach is discussed in [19]. As a first step, the service architecture of the crude oil refinery is constructed with the combination of MQTT and Azure cloud to enable reliable data and command delivery. The second step is a smart disruption management system that monitors the data, alerts modules, disruption management modules, and procedures for rescheduling the operations. During multiple data transmission in MQTT causes packet losses. A reliable messenger system between server and client in the MQTT protocol has been developed [20] by combining the message topic with the sequence number and flag number in order to prevent data losses during transmission.

A reliable MQTT broker implementation [21] has broadcast the information to the clients. Based on numerical computations, a solution has been developed to minimise the number of shared messages among brokers. Additionally, clusters of brokers are formed among the network edge nodes to minimise the delay between publishers and subscribers. To address the sub-linearity issue for multiple sessions in the MQTT protocol, greedy load balancing is used to serve publisher and subscriber topics to improve cluster scalability. 

### TABLE 2. Recent works related to IDS on IoT Networks

| Reference | Year | Detection Method | Inference |
|-----------|------|------------------|-----------|
| [12]      | 2013 | Anomaly-based    | The IPv6 enabled IDS framework for securing IoT devices using 6LoWPAN has been discussed with various vulnerable attacks inherited on the framework based on WSN and IoT protocols |
| [13]      | 2016 | Pattern-based    | The Raspberry Pi with an open-source IDS called Snort has been used to detect the vulnerability in the IoT networks |
| [14]      | 2018 | Anomaly-based    | A plug and play-based network IDS called Kit.NET has been designed based on autoencoders |
| [15]      | 2018 | Pattern-based    | An enhanced pattern recognition algorithm based on IDS has been used to secure the IoT networks with a four-layered architecture |
| [16]      | 2019 | Anomaly-based    | To achieve a high detection ratio of attacks in IoT networks, a deep learning-based IDS with a combination of spider monkey optimization algorithm and stacked deep polynomial network is used to select optimal features |
| [17]      | 2020 | Anomaly-based    | To detect DDoS, DoS, Scan, and Theft attacks on the IoT dataset, a two-stage hierarchical IDS employing Multimodel Deep Autoencoders with soft classifiers is used |
| [18]      | 2020 | Anomaly and Pattern-based | Passban-based IDS is used to protect IoT devices and gateways with ML algorithms |

The MQTT-SN security bootstrapping architecture [26] was designed to provide end-to-end security for the data. To achieve end-to-end security, the concept of topic certificates is employed. Topic certificates are used to build an encrypted direct route between a publisher and a group of subscribers without depending on transport layer security (TLS). MQTT threat model (MQTT-Threats) can secure the communication between pub and sub by adding the TTS header and security-specific header to the existing MQTT frame header [27].

A security scheme in the MQTT protocol uses cryptographic smart cards [28] for encrypting the client-to-broker data communication without changing protocol message specifications. This authentication scheme meets the requirements of both data secrecy and data integrity. The implementation of the Keyed-Hash Message Authentication Code (HMAC) generation algorithm [29] has been employed instead of a complex encryption algorithm to improve the efficiency with less time for the client to retrieve the messages. The different cloud MQTT brokers were analyzed from a security standpoint and conducted a denial-of-service (DoS) attack and information collection techniques to discover the least vulnerable broker was discussed in [30].

On analyzing traffic features using the internet protocol packet size entropy (IPSE) [34] method, the DoS attack can be detected. The design of IPSE can identify both long-term attacks and short-term attacks on any architecture by observing time series of packet size entropy and distinct traffic packet size distribution shifts during DoS attacks. A small change in step rise time in the time series indicates that a possible DoS attack is happening. In the case of a normal feature, no changes can occur in step rise time. An MQTT threat model [35] has been established between IoT devices and MQTT brokers and endpoints such as servers, personal computers, and smartphones. The testbed simulates and analyses attack scenarios such as DoS, identity spoofing, information exposure, and privilege elevation. The MQTT Mosquito broker was installed on a virtual computer running Ubuntu Linux 14.0 for analyzing the SYN flood attack. The large packet payload, up to 2000 PUBLISH messages with a
4 MB header payload, is sent to the broker during that time the subscriber cannot get the message or there is a time delay in receiving the message from the publisher.

To investigate further exploitations of the MQTT protocol with normal and encrypted communications, the target devices are subjected to a novel low-rate distributed denial of service (DDoS) attack known as SlowI Te [36]. The framework [37] for the DoS attack is created on the application layer and implemented to evaluate the MQTT protocol against the DoS attack with two classes: legitimate and DoS attack. The DoS attack dataset is created with different attack scenarios such as connect flooding, delayed connect flooding, payload connect flooding, and invalid subscription flooding. A ML model was developed using a dataset to mitigate the DoS attack on MQTT message brokers. This framework shows the high effectiveness of detecting DoS attacks. Furthermore, the multiclass IDS [38] has been designed with ML techniques with two methods of classification, such as ensemble and deep learning. First, the various attack captured datasets have been created in order to detect the anomalies in an IoT environment.

The security vulnerability of MQTT has been categorized into two: data security and network security. Where data security is referred to as the data transfer from sensors to actuators with an encrypted format. In the network, security is responsible for preventing unauthorized and flooding attacks. The main motivation for this paper is to prevent unauthorized and flooding attacks on IoT-MQTT networks using ML algorithms. The main technical challenges in other existing datasets are that they have more features without device-level implementation, which leads to more complex detection of attacks in real-time. The proposed contributions give a better solution when compared to other literature. Table 3 describes the various parameters considered for generating the IoT-MQTT context dataset. These datasets are differentiated by label, IoT telemetry data, Heterogeneity of IoT data sensor, real-time devices and device-level implementation.

### A. CONTRIBUTIONS
The main research contributions of this paper are described as follows.

1) A novel IoT experimental setup is created with heterogeneous sensors and real-time devices with legitimate and attack dataset SENMQTT-SET.

2) An ensemble statistical multi-view cascade feature generation algorithm is developed to evoke the important features from the SENMQTT-SET in order to detect the attack in less time.

3) Seven ML algorithms are evaluated to show the effectiveness and reliability of the proposed dataset. And for building the best model using Elite Machine Learning (EML) for detecting the attack. The accuracy, f1-score, and other performance metrics are compared with the existing dataset.

4) Finally, performance metrics such as cumulative distribution function, jitter, packet length, and latency are evaluated and validated in three scenarios: normal, subscriber attack, and broker attack.

### III. METHODOLOGY
This section describes three parts of a proposed method such as (i) Framework for attack detection and proposed SENMQTTSET dataset generation, (ii) DoS attack model, and (iii) Proposed multi-context features generation and EML.

#### A. FRAMEWORK FOR ATTACK DETECTION AND PROPOSED SEN-MQTTSET DATASET GENERATION
A typical IoT-MQTT network has a group of sensors connected to IoT devices. The proposed framework testbed environment is created by using the four Raspberry Pi [39] (RPi) (namely A, B, C, and D), two Node Mcu [40] (NMcu) (namely E and F) and one Raspberry Pi (G) are connected to router as depicted in Figure 2. Here RPi (A, B) and NMcu (E) act as a publisher, RPi (C, D) and NMcu (F) act as a subscriber and RPi (G) act as a Mosquito MQTT broker the IP address of each device is tabulated in Table 4. To monitor or control the device a router is interconnected.

#### TABLE 4. IP address connection configuration

| Device ID | Device Name | IP Address     | Device Type |
|-----------|-------------|----------------|-------------|
| A         | Raspberry Pi| 192.168.0.101  | Publisher   |
| B         | Raspberry Pi| 192.168.0.102  | Publisher   |
| C         | Raspberry Pi| 192.168.0.103  | Subscriber  |
| D         | Raspberry Pi| 192.168.0.104  | Subscriber  |
| E         | Node Mcu    | 192.168.0.105  | Publisher   |
| F         | Node Mcu    | 192.168.0.106  | Subscriber  |
| G         | Raspberry Pi| 192.168.0.107  | Broker      |
The proposed traffic generation consists of three devices such as (1) Five RPi, Two NMcu (2) an attack machine, and (3) a router. For attacks, the personal computer (PC) is equipped with an Intel 8-core i5 CPU and 8 GB RAM. In these three different scenarios, the raw traffic features pcap is captured by using tcpdump [41] tools later the specific MQTT traffic filter is applied with tshark [42] to extract the particular features and processed to csv file. After generating the dataset from the devices, two attributes of label like normal or attack type is added. These labels are appended to the original csv files generated after feature extraction.

Algorithm 1 describes the script for four simulated sensor data such as air quality sensor, passive infrared motion sensor, temperature sensor, and CO2 gas sensor implemented on RPi and NMcu. Python [43] and microPython [44] are used to create the sensors script for all the physical devices RPi and NMcu respectively. The main objective of the script is to measure the four different sensors such as air quality sensors, passive infrared sensors, temperature sensors, and CO2 sensors according to the parameter given in Table 5. These sensor data with the specific topic is published to the MQTT broker of RPi G.
### TABLE 5. Sensor data parameters

| Sl.No | Sensor Name | Min/Max range | Description |
|-------|-------------|---------------|-------------|
| 1     | Air quality sensor (MQ135) [46] | 10-300 ppm for NH3, 1000 ppm for Benzene, and 10-300 ppm for Alcohol | MQ135 sensor is used for detecting gases like NH3, NOx, alcohol, benzene, smoke and CO2. These can be employed in industry and home. |
| 2     | Passive infrared sensor [47] | 0-no detection 1-any human detected | PIR sensor is used to detect the motion of humans. This also can be employed in industry and home. |
| 3     | Temperature sensor (DS18B20) [48] | -55 to 125°C | Temperature sensor is used to detect the temperature of the particular place. This can be employed in industry and home. |
| 4     | CO2 sensor (MH-Z19) [49] | 0 to 5% VOL optional | CO2 sensor is used to detect the carbon content in the air by using non-dispersive infrared (NDIR). This can be employed in the industry. |

An attacker can send the MQTT CONNECT packet without valid credentials, hence the clients cannot publish or subscribe without successful broker connections. Using a brute force attack, an attacker can also change the PUBLISH or SUBSCRIBE packet parameters without correct credentials and authorization. The control packet data and the level of access being provided to an attacker is referred to as basic connect flooding (BCF).

#### B. ATTACK MODEL - DOS

The main aim of DoS attacks is to send many flooding messages to the server to prevent legitimate user access. The average number of users to access the system is called Little’s Law [45] and the formula is given as follows:

\[
L = \lambda \times V, \tag{1}
\]

where \( \lambda \) is the system arrival rate for sensor data, where \( V \) is the average system time a sensor data spends, and \( T \) is the total time taken in the system. DoS attacks try to occupy the system for a long time, so legitimate features cannot get the service. It increases the processing time per packet and increases the arrival rate of packets through complicated computer activities on the victim’s device. Mostly the authentication and authorization can prevent unauthorized access in any industrial MQTT-IoT systems, and authenticated clients are permitted to send or receive messages on selected subjects only. The access levels of MQTT-IoT data are two types: valid MQTT broker connection credentials and valid pub/sub service authorization.

#### Algorithm 2: Basic Connect Flood generation steps [37]

**Input:** \( N_c, bip \) and \( port \) where \( N_c \) is the number of connect packets, \( bip \) is MQTT broker IP address and \( port \) is the MQTT port number.

1. \( bip \leftarrow 192.168.0.107 \)
2. \( port \leftarrow 1883 \)
3. \( cid \leftarrow mqtt - random(1,1000) \)
4. \( r = client.connect(bip, port, cid) \)
5. \( c = [] \)
6. for \( i = 1 \) to 10,000 do
7. \( c[i] \leftarrow cid \)
8. \( N_c = c[i] \)
9. \( N_c[i - 1].connect(bip, port, c[i]) \)

Once the connection is established on the MQTT protocol between the publisher and the subscriber, the publisher will send a CONNECT frame packet to the MQTT broker to establish the connection. The broker will reply CONACK to the publisher. The basic connect flooding attacks will send \( N \) connects to the MQTT broker in order to keep it busy. In this way, the subscriber cannot receive the packets from the publisher on time.

#### C. PROPOSED MULTI-CONTEXT FEATURES GENERATION AND ML TECHNIQUES

The proposed SEN-MQTTSET dataset has a combination of both TCP and MQTT traffic data. Each network traffic-identified flow has been retrieved for each pcap file: frame length, delta time, IP length, source IP, destination IP, IP port, MQTT message type, length, message, etc. The raw dataset has 120 features, including the labels of classification. As the features of the dataset are in large volume, the space required to store the dataset is greater and some of the ML algorithms may not work properly, so the feature reduction helps to solve this issue. The proposed multi-context feature generation is combined with two parts (i) The best of two feature selection from the raw dataset using Logistic Regression (LR) (ii) Ensemble statistical multi-view feature generation. The proposed flow diagram is given in Figure 3. The proposed multi-context feature generation steps are presented in Algorithm 3. The best-selected features are frame length and frame delta time, obtained from the raw dataset among 120 features including flow-based, TCP, IP, and MQTT using Logistic Regression. Based on the selected features, the statistical features with combined session features have been created and named the optimized SENMQTT-SET dataset. Now the proposed optimized dataset is preprocessed using normalization techniques and trained using ML techniques such
as LR, K-Nearest Neighbour (KNN), Random Forest (RF), Naive Bias (NB), Support Vector Machine (SVM), Gradient Boosting (GB), and Decision Tree (DT). To detect the attack by using an elite ML prediction model.

**1) Data pre-processing**

The Min-Max scaling normalizes the input data feature of optimized SENMQTT-SET into the static interval to tackle varied dynamic ranges for the 10 input features shown in Table 6. By applying Min-Max normalization, the optimized SENMQTT-SET dataset features are converted into a linear transformation so that the ML algorithms can easily learn the features and develop the prediction model.

**2) EML and ML models**

The EML is proposed to select the best ML algorithm attack detection model from the LR, K-Nearest Neighbour (KNN), Random Forest (RF), Naive Bias (NB), Support Vector Machine (SVM), Gradient Boosting (GB), and Decision Tree (DT).

a: **LR**

The LR [50] model is based on statistical calculations which follow the tree-based calculations to forecast the value of the binary predictive variable $y \in [0, 1]$, where 0 is a negative class, and 1 is a positive class.

b: **KNN**

KNN [50] is a type of supervised classification algorithm that identifies the output for newer input. It employs the training set for training and learning as it takes control of input sample similarity to predict the output.

c: **SVM**

SVM [51] is a classification technique that transforms complex quadratic programming problems into solutions with maximum margin strategies.

---

**Algorithm 3: Proposed multi-context feature generation**

**Input:** $M[v][f]$, where the $v$ is the set of flows in input data from $[v_1, v_2, ..., v_L]$ and $f$ is the number of feature in input data from $[f_1, f_2, ..., f_N]$, where $L, N$ is the length of data and number of features respectively.

**Output:** $O[v][f]$

| Sl.No | Symbol | Features | Description | Type |
|-------|--------|----------|-------------|------|
| 1     | $P_n$  | num_pks  | Number of Packets in the flow | N    |
| 2     | $M_Pn$ | min_pkc_len | Minimum packet length | N    |
| 3     | $M_Pn$ | max_pkc_len | Maximum packet length | N    |
| 4     | $M_Pn$ | mean_pkc_len | Average packet length | N    |
| 5     | $SP_n$ | sd_pkc_len | Standard deviation of packet length | N    |
| 6     | $T_d$  | time_epoch | Arrival inter time | N    |
| 7     | $MT_d$ | min_tat | Minimum inter arrival time | N    |
| 8     | $M_dT_d$ | max_tat | Maximum inter arrival time | N    |
| 9     | $M_dT_d$ | mean_tat | Average inter arrival time | N    |
| 10    | $ST_d$ | sd_intarr | Standard deviation of inter arrival time | N    |
| 11    | $A_k$  | is_attack | Normal or Attack | B    |
d: NB
The NB [50] is a type of supervised learning and classification algorithm. The class of a new feature vector is figured out via the conditional probability theorem, which obtains the conditional estimate of vectors for a given class by analyzing the training dataset. The new vector class is determined based on the conditional probability of each vector’s probability values.

e: RF
RF [52] is a classification method that is based on tree structures. The results are obtained by the formation of a data forest with many sub-trees. It generates a random selection of the training set for decision trees and forms the tree according to the outcome of the individual set. The RF algorithm can handle missing values and it also helps to avoid overfitting by using an appropriate parameter tweaking method.

f: DT
DT [50] is a supervised ML algorithm used for classification and regression. Every node in a decision tree is either a leaf node or a decision node. The decision-making processes are simply comprehensible because they are straightforward and the tree is interconnected with internal and external nodes. The decisions are made by internal nodes, which also visit their child nodes in the next decision-making steps, provided the leaf node is node-less and tagged.

g: GB
GB [50] is a ML technique used for classification and regression. A GB uses many decision trees to merge their individual forecasts into final predictions. The approach of boosting, which relies on the concept of combining a bunch of simple learners, gives better prediction performance.

IV. RESULTS AND DISCUSSIONS
The testbed consists of five RPi devices and two NMcu connected with simulated sensors. The RPi and NMcu are connected by a modem with a common SSID and password. The attacker/victim is a personal computer with an i3 processor to inject the basic connect flooding attack on the subscriber and broker. All the publisher and subscriber devices are interconnected with the Mosquito MQTT broker. Using the Paho MQTT client library in the program of RPi can publish and subscribe to the sensor data connected to the devices. Similarly, the esp8266 client library is used in NMcu to publish and subscribe to the sensor data. The dataset was collected for three scenarios, such as no attack, attack on subscriber, and attack on broker. Figure 4 shows the experimental testbed with four RPi, two NMcu and a router. In all scenarios, the RPi A, B, and E of NMcu publishes the sensor data with the help of Python script and embedded c via MQTT broker. And the RPi C, D, and G of NMcu for the subscriber of the sensor data with the help of Python and microPython script via MQTT broker. Using the tcpdump tool, the packets are captured for 1800 seconds.
from the subscriber to the publisher. In RPi C executed the tcpdump command for obtaining the legitimate and attack traffic features.

A. SCENARIO 1: NO ATTACK
In scenario 1 (no attack) the traffic features have been captured between the publisher and the subscriber via a broker. The four sensors data will keep on going from publisher to MQTT broker. The schematic diagram shows the connections of scenario 1 is illustrated in Figure 5.

Figure 6 shows the details of sniffed packets of tcpdump in wireshark for obtaining the individual packet details. In wireshark the seven important pieces of information about the packets are visible such as number, time, source, destination, protocol, length, and info. The number represents the number of the network packets, time represents the time at which the specific packet has been recorded, the source is the IP address of where the packets are coming from, the destination is the IP address of where the packet is going, protocol refers to the TCP, MQTT, etc., and info represents the additional information about the packets which has been recorded. In the info field, there is only one CONNECT command and one ACK packet present for scenario 1. Later these captured pcap files are converted to csv files using Python script. Table 7 displays the four sensor data from Scenario 1 along with the timestamp and message topic.

Table 7. Scenario 1: Sensor data

| Sl.No | Timestamp | Message | Message Topic |
|-------|-----------|---------|---------------|
| 1     | 11:54:02  | 33      | sens/temp     |
| 2     | 11:54:06  | 17      | sens/air      |
| 3     | 11:54:10  | 01      | sens/motion   |
| 4     | 11:54:14  | 45      | sens/gas      |
| 5     | 11:54:40  | 37      | sens/temp     |
| 6     | 11:55:25  | 21      | sens/air      |

B. SCENARIO 2: ATTACK ON SUBSCRIBER
The architecture setup for scenario 2 is depicted in Figure 7 where an attack is on the subscriber side. A basic connect flooding attack is injected at the subscriber node by the Python script on the PC. Tcpdump tools capture the packets transferred from the publisher to the subscriber and saved to a pcap file.

Figure 8 shows the details of sniffed packets using tcpdump for obtaining the individual packet details. In the info field, there are many CONNECT commands and ACK packets present for scenario 2. Later these captured pcap files are converted to csv files using Python script. Table 8 shows the four sensor data with timestamp and message topic of scenario 2.

C. SCENARIO 3: ATTACK ON BROKER
The architecture setup is illustrated in Figure 9 where an attack is on the MQTT Broker. A Basic connect flooding attack is injected at the MQTT broker node by the Python script in the PC. Tcpdump tools captured the packets transferred from the publisher to the subscriber and saved to the pcap file.
Figure 8. Wireshark Sniffed Packets Scenario 2.

Table 8. Scenario 2: Sensor data

| Sl.No | Timestamp   | Message | Message Topic |
|-------|-------------|---------|---------------|
| 1     | 11:54:02    | 37      | sens/temp     |
| 2     | 11:54:25    | 218     | sens/air      |
| 3     | 11:55:10    | 0       | sens/motion   |
| 4     | 11:56:08    | 87      | sens/gas      |
| 5     | 11:56:28    | 37      | sens/temp     |
| 6     | 11:57:05    | 208     | sens/air      |

D. QUALITY OF PARAMETERS

Quality of parameters such as jitter, packet length, and latency is computed for the three scenario experimental setups.

Figure 10 shows the details of sniffed packets of tcpdump in wireshark for obtaining the individual packet details. In info field there are many CONNECT commands and ACK packets present for scenario 3. Later these captured pcap files are converted to csv files using Python script. Table 9 show the four sensor data with timestamp and message topic of scenario 3.

Table 9. Scenario 3: Sensor data

| Sl.No | Timestamp   | Message | Message Topic |
|-------|-------------|---------|---------------|
| 1     | 11:51:42    | 39      | sens/temp     |
| 2     | 11:52:17    | 158     | sens/air      |
| 3     | 11:53:10    | 00      | sens/motion   |
| 4     | 11:53:29    | 97      | sens/gas      |
| 5     | 11:53:59    | 197     | sens/air      |
| 6     | 11:54:09    | 01      | sens/motion   |

Figure 11. CDF of Packet Length

a: Cumulative Distribution Function

The cumulative distribution function (CDF) calculates the cumulative probability for a frame packet length value. The probability of a random observation taken from the features is less than or equal to a certain value that can be determined by the CDF. The distribution of frame packet lengths can also be observed using the cumulative distribution function as shown in Figure 11. There is a shift in the curve towards the right when the node is under attack.

b: Jitter

The publisher has four sensor message topics, namely sens/temp, sens/air, sens/motion, and sens/gas. In jitter analysis, the packet delay between the same message topic and the
time difference between all sensor data packets received by the subscriber and the publisher has been computed. Table 10 shows the packet loss for three scenarios such as no attack, attack on subscriber, and attack on broker. The packet loss is calculated for 10 and 20 s, when the attack on broker the packet loss of 9.1% and 5.8% respectively, when there is no attack on subscriber 6.6% and 6.8% respectively and when no attack 0% and 0% respectively.

**TABLE 10.** Packet Loss Comparison between Normal and Attack Scenario.

| Time (s) | No Attack | | Subscriber Attack | | Broker Attack | |
|---|---|---|---|---|---|---|
| | Broker | Subscriber | PL(%) | Broker | Subscriber | PL(%) | Broker | Subscriber | PL(%) |
| 10 | 197 | 197 | 0 | 197 | 184 | 6.6 | 177 | 161 | 9.1 |
| 20 | 267 | 267 | 0 | 267 | 249 | 6.8 | 243 | 229 | 5.8 |

data published to the subscriber via the MQTT broker. The average time delay without attack and with attack is 80 ms and 1230 ms, respectively.

**FIGURE 12.** End to End time delay distribution of Subscriber without attack and with attack

**FIGURE 13.** End to End time delay distribution of broker without attack and with attack

Figure 12 shows the end-to-end time delay distribution of subscribers without and with an attack. The end-to-end time delay time is calculated at the subscriber node by the sensor data published to the subscriber via the MQTT broker. The average time delay without attack and with attack is 30 ms and 640 ms, respectively. Figure 14 shows the transmit time delay between the subscriber packet without attack and with attack. 10000 packets were sent from the publisher to the end-user, called a subscriber, via a broker. The average time delay between two consecutive packet data transmissions without attack and with attack is 200 ms and 750 ms, respectively.

**FIGURE 14.** Packet transmitting delay in subscriber without attack and with attack

**FIGURE 15.** Packet transmitting time difference in broker without attack and with attack

Figure 13 shows the end-to-end time delay distribution of brokers without and with an attack. The end-to-end time delay time is calculated at the MQTT broker, where the sensor data is published to the subscriber. On seeing this graph, the average time delay for without attack and with attack is 80 ms and 1230 ms, respectively. Figure 14 shows the transmit time delay between the subscriber packet without attack and with attack. 10000 packets were sent from the publisher to the end-user, called a subscriber, via a broker. The average time delay between two consecutive packet data transmissions without attack and with attack is 200 ms and 750 ms, respectively.

**FIGURE 15.** Packet transmitting time difference in broker without attack and with attack

Figure 12 shows the end-to-end time delay distribution of subscribers without and with an attack. The end-to-end time delay time is calculated at the subscriber node by the sensor data published to the subscriber via the MQTT broker. The average time delay without attack and with attack is 80 ms and 1230 ms, respectively.

**FIGURE 14.** Packet transmitting delay in subscriber without attack and with attack

**FIGURE 15.** Packet transmitting time difference in broker without attack and with attack

Figure 13 shows the end-to-end time delay distribution of brokers without and with an attack. The end-to-end time delay time is calculated at the MQTT broker, where the sensor data is published to the subscriber. On seeing this graph, the average time delay for without attack and with attack is 30 ms and 640 ms, respectively. Figure 14 shows the transmit time delay between the subscriber packet without attack and with attack. 10000 packets were sent from the publisher to the end-user, called a subscriber, via a broker. The average time delay between two consecutive packet data transmissions without attack and with attack is 200 ms and 750 ms, respectively.

**FIGURE 15.** Packet transmitting time difference in broker without attack and with attack

Figure 12 shows the end-to-end time delay distribution of subscribers without and with an attack. The end-to-end time delay time is calculated at the subscriber node by the sensor data published to the subscriber via the MQTT broker. The average time delay without attack and with attack is 80 ms and 1230 ms, respectively.
two consecutive packet data transmissions without attack and with attack is 250 ms and 600 ms, respectively.

c: Packet length

![Packet Length Subscriber With and Without Attack](image)

**FIGURE 16.** Subscriber node packet length of sensor data in without attack and with attack

Figure 16 shows the packet length of sensor data without attack and with attack received at subscriber node. The packet length varies from 32 to 58 for without attack scenario and in with attack varies from 40 to 50. This graph shows that some of the packet lengths get increased and truncated in with attack when compared to without attack.

![Packet Length Broker With and Without Attack](image)

**FIGURE 17.** Broker node packet length of sensor data in without attack and with attack

Figure 17 shows the packet length of sensor data without attack and with attack received at broker node. The packet length varies from 42 to 50 for without attack scenario and in with attack varies from 0 to 56. This graph shows that some of the packet length gets decreased and truncated in with attack when compared to without attack.

d: Latency

![Latency Analysis](image)

**FIGURE 18.** Latency on subscriber node without and with attack

![Latency Analysis](image)

**FIGURE 19.** Latency on broker node without and with attack

In latency analysis, the timestamp is appended to all the sensor data of the message topic, such as sens/temp, sens/air, sens/motion, and sens/gas, shown in Table 11. The timestamp of the received data has been recorded at wireshark, allowing the time delay between subscriber and publisher to be calculated for both the subscriber without and with attack, as well as the broker without and with attack.

Figure 18 shows the time delay between the publisher and the subscriber without and with an attack. The average time delay between the publisher and a subscriber is 100 ms without an attack and 300 ms with an attack on the subscriber node.

Figure 19 shows the time delay between the publisher and the subscriber without and with an attack. The average time delay between the publisher and a subscriber is 100 ms without an attack and 400 ms with an attack on the broker node.

E. ASSESSMENT OF ML ALGORITHMS

The assessment of various ML algorithms has been carried out to find the performance metrics such as accuracy, attack detection rate, attack predictive value, normal predictive value, mathews correlation coefficient, F1 score, and false alarm rate, to validate the ML models by using a confusion
### TABLE 11. Latency Analysis

| Sl.No | Timestamp | Message | Message Topic | Time difference (s) |
|-------|-----------|---------|---------------|---------------------|
| 1     | 11:39:31.6864 | 11:39:31.6739 | sens/temp    | 0.0124             |
| 2     | 11:39:31.8079 | 11:39:31.7960 | sens/air     | 0.0118             |
| 3     | 11:39:31.8168 | 11:39:31.8055 | sens/motion  | 0.0113             |
| 4     | 11:39:31.9410 | 11:39:31.9294 | sens/gas     | 0.0115             |
|       | 11:54:32.4503 | 11:54:30.9320 | sens/temp    | 1.5182             |
| 2     | 11:54:32.4600 | 11:54:30.9414 | sens/air     | 1.5186             |
| 3     | 11:54:32.4957 | 11:54:30.9602 | sens/motion  | 1.5355             |
| 4     | 11:54:32.5294 | 11:54:31.0074 | sens/gas     | 1.5220             |

Broker without attack

| Sl.No | Timestamp | Message | Message Topic | Time difference (s) |
|-------|-----------|---------|---------------|---------------------|
| 1     | 11:39:29.5550 | 11:39:29.5487 | sens/temp    | 0.0063             |
| 2     | 11:39:29.5690 | 11:39:29.5577 | sens/air     | 0.0113             |
| 3     | 11:39:29.7494 | 11:39:29.7731 | sens/motion  | 0.1723             |
| 4     | 11:39:29.7834 | 11:39:29.7731 | sens/gas     | 0.0103             |

Broker with attack

| Sl.No | Timestamp | Message | Message Topic | Time difference (s) |
|-------|-----------|---------|---------------|---------------------|
| 1     | 11:54:02.9533 | 11:54:02.9113 | sens/temp    | 0.0420             |
| 2     | 11:54:02.9817 | 11:54:02.9301 | sens/air     | 0.0516             |
| 3     | 11:54:03.2952 | 11:54:03.1138 | sens/motion  | 0.1813             |
| 4     | 11:54:03.4626 | 11:54:03.2821 | sens/gas     | 0.1805             |

#### FIGURE 20. Confusion Matrix.

1) Confusion matrix

The confusion matrix [53] is an error matrix used to define a classification models performance on a test data set with certain true values. By using the confusion matrix, it is possible to visualize the performance of various ML algorithms, as shown in Figure 20.

- **a: Accuracy (ACC)**
  
  The percentage of all features correctly defined over all the data is as follows.

  \[
  ACC = \frac{TP + TN}{TP + FP + TN + FN},
  \]

  where TP—true positive, TN—true negative, FP—false positive, and FN—false negative.

- **b: Attack Detection rate (ADR)**
  
  The percentage of the ratio between the total features of an attack instance and the correctly identified attack features called ADR.

  \[
  ADR = \frac{TP}{TP + FN}
  \]

- **c: Attack predictive value (APV)**
  
  The percentage of the ratio between attack feature instances correctly detected among the expected cases as an attack is called APV.

  \[
  APV = \frac{TP}{TP + FP}
  \]

- **d: Normal predictive value (NPV)**
  
  This refers to the percentage of normal features instances correctly detected among the real normal features instances.

  \[
  NPV = \frac{TN}{TN + FP}
  \]

- **e: Mathews correlation coefficient (MCC)**
  
  MCC is a statistical metrics that is used to evaluate a classification models performance.

  \[
  MCC = \frac{TP*TN - FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}
  \]

- **f: F1 Score**
  
  The F1 Score rate is the ratio of normal requests that are detected as attacks overall normal requests as follows:

  \[
  F1score = \frac{2*TP}{2*TP + FP + FN}
  \]

- **g: False alarm rate (FAR)**
  
  False alarm rate (FAR) is an error metric used to evaluate the classification model performance based on the probability of false-positive with the null hypothesis.

  \[
  FAR = \frac{FP}{FP + TN}
  \]

- **h: Receiver operating characteristic (ROC):**
  
  Receiver operating characteristic (ROC) curve is helpful to classify the thresholds performance model with the plot. This plot consists of two parameters namely true positive rate (TPR) and false positive rate (FPR).

  \[
  TPR = \frac{TP}{TP + FN}
  \]

  \[
  FPR = \frac{FP}{FP + TN}
  \]

Figure 21 shows the ROC curves for the seven ML models. It can be seen that DT, RF, and GB classifiers provide better performance when compared with the other classifiers (LR, KNN, SVM and NB).

Table 12 describes the evaluation metrics such as train accuracy, test accuracy, ADR, APV, NPV, F1-Score, MCC, etc.
TABLE 12. Evaluation of ML models using proposed optimized SENMQTT-SET dataset.

| Models | Train Accuracy | Test Accuracy | ADR | APV | NPV | F1-Score | MCC | FAR | ROC | Training time(s) | Prediction time(s) |
|--------|----------------|---------------|-----|-----|-----|---------|-----|-----|-----|-----------------|--------------------|
| LR     | 90.04          | 90.36         | 0.926 | 0.9152 | 0.8691 | 0.920 | 0.797 | 0.13 | 0.953          | 0.115              |
| KNN    | 100            | 99.89         | 0.998 | 1.0 | 1.0 | 0.99 | 0.998 | 0 | 0.99 | 0.013            | 0.187              |
| SVM    | 94.13          | 94.61         | 1.0 | 0.918 | 0.863 | 0.957 | 0.890 | 0.136 | 0.984 | 0.303            | 0.254              |
| NB     | 83.16          | 84.1          | 0.823 | 0.905 | 0.869 | 0.863 | 0.680 | 0.13 | 0.912 | 0.004            | 0.0038             |
| DT     | 100            | 100           | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0 | 1.0 | 0.008            | 0.004              |
| RF     | 100            | 100           | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0 | 1.0 | 2.80             | 0.462              |
| GB     | 100            | 100           | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0 | 1.0 | 0.68             | 0.012              |

TABLE 13. Performance comparison of proposed SENMQTT-SET, [31], [32] and [33] dataset.

| Reference | Algorithm | ACC % | F1-Score |
|-----------|-----------|-------|----------|
| SENMQTT-SET | LR       | 90.36 | 0.92     |
|           | KNN       | 100   | 0.99     |
|           | SVM       | 99.73 | 0.957    |
|           | NB        | 97.8  | 0.863    |
|           | DT        | 100   | 1.0      |
|           | RF        | 100   | 1.0      |
|           | GB        | 100   | 1.0      |
| [31]      | NB        | 98.7  | 0.9897   |
|           | DT        | 97.79 | 0.985    |
|           | RF        | 99.42 | 0.9943   |
|           | GB        | 99.11 | 0.9916   |
| [32]      | LR        | 61    | 0.46     |
|           | KNN       | 84    | 0.84     |
|           | SVM       | 61    | 0.46     |
|           | NB        | 62    | 0.51     |
|           | RF        | 85    | 0.85     |
| [33]      | LR        | 98.23 | 0.98     |
|           | KNN       | 99.68 | 0.99     |
|           | SVM       | 82.6  | 0.8242   |
|           | NB        | 78    | 0.752    |
|           | DT        | 99.96 | 0.99     |
|           | RF        | 99.98 | 0.998    |

Far, ROC, training time and testing time of ML models using proposed optimized SENMQTT-SET. DT classifiers outperforms well when compared to the other ML model. Table 13 describes the comparison of the various algorithms for accuracy and F1 score of the proposed multi-context algorithm with [31], [32] and [33]. The elite ML model is generated and tested for attack detection using the proposed feature generation from the SEN-MQTTSET dataset. The proposed ensemble multi-view cascade feature consists of 11 attributes such as time_epoch, num_pkts, mean_iat, sd_iat, min_iat, max_iat, mean_pkt_len, std_pkt_len, min_pkt_len, max_pkt_len and is_attack. The dataset consists of normal traffic instances and attacks class instances with labels 0 for normal and 1 for the attack. Among the 11 features, 10 are numerical and 1 is binary. The proposed optimized SENMQTT-SET dataset is divided randomly into 20% of the testing and 80% is training to evaluate using ML algorithms. Figure 22 shows the comparison of performance metrics such as accuracy, F1-score, attack detection rate, and attack prediction for all ML algorithms. The maximum testing accuracy obtained from DT, RF, and GB is 100%, KNN is 99.89, SVM is 94.13%, NB is 83.16% and LR is 90.04. Among the seven ML models, the best model is called the Elite ML model, based on all the best metrics of the DT classifier algorithm. When compared to the optimised SENMQTT-SET with [31], [32] and [33], the proposed multi-view cascade feature dataset accuracy and f1 score are higher and differ by approximately 1%.
DT has outperforms with training time and detection time of 8 ms and 4 ms respectively, even though the NB achieved less training time and detection time of 4 ms and 3.8 ms respectively but the accuracy and other performance metrics are low when compared to DT. Finally, the elite ML chooses the best ML model to detect attacks in real-time.

V. CONCLUSION

In this work, the SENMQTT-SET dataset has been generated and analyzed for MQTT attack detection in IoT contexts. The intrusion detection testbed includes three scenarios: no attack, attack on a subscriber, and attack on a broker and it has been designed to record regular traffic and attack characteristics. The SEN-MQTTSET dataset was developed from the traffic features while the data was received from different networked IoT sensors (temperature, motion sensors, etc.). An ensemble multi-view cascade feature generation algorithm was used to generate optimized SENMQTT-SET from the raw dataset. Later, these features are evaluated by using ML algorithms and selected the best model using elite ML to detect an attack or normal in the MQTT-IoT network.

The generated proposed features show the effective intrusion detection system in MQTT cyber-attacks and achieve an accuracy of more than 99%. The detailed quality of parameters was measured for three scenarios, and the packet time delay for without attack was less than that for with attack. In the future, deep learning-based explainable artificial intelligence (XAI) methods [54] can be used to improve the behavior of multi-modal traffic classifiers.

ACKNOWLEDGMENT

The authors would like to thank Dr. Sathish Babu S, Chandhar Research Labs, Chennai, Tamil Nadu, India, for his valuable suggestions to improve the quality of the manuscript and Aditi Bagat, Nayana Jyoti, Ashuwin S, Abhishek K, Department of Electrical and Electronics Engineering, NIT Tiruchirappalli, Tamil Nadu, India and Kunal Jhadav, Mahindra University, Hyderabad, Telangana for their valuable discussions on machine learning framework.

REFERENCES

[1] P. Sethi and S. R. Sarangi, “Internet of things: architectures, protocols, and applications,” Journal of Electrical and Computer Engineering, vol. 2017.
[2] Statista Research Department, “Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025," 2016. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/
[3] C. Tandark, “The security issues of the internet of things,” Computer Fraud & Security, vol. 2015, no. 9, pp. 11–14, 2015.
[4] M. Ahmad, T. Yousin, M. A. Habib, R. Ashraf, and S. H. Ahmed, “A review of current security issues in internet of things,” Recent trends and advances in wireless and IoT-enabled networks, pp. 11–23, 2019.
[5] T. Yokotani and Y. Sasaki, “Comparison with http and mqtt on required network resources for iot,” in 2016 international conference on control, electronics, renewable energy and communications (ICCCERE), IEEE, 2016, pp. 1–6.
[6] Edited by Andrew Banks and Rahul Gupta, “MQTT Version 3.1.1.” 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html
[7] B. Mishra and A. Kertesz, “The use of mqtt in m2m and iot systems: A M. T.,” IEEE Access, vol. 8, pp. 201 071–201 086, 2019.
[8] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, “Mqtt-s—a publish/subscribe protocol for wireless sensor networks,” in 2008 3rd International Conference on Communication Systems Software and Workshops (COMSWORK’08). IEEE, 2008, pp. 791–798.
[9] S. Lee, H. Kim, D.-k. Hong, and H. Ju, “Correlation analysis of mqtt loss and delay according to qos level,” in The International Conference on Information Networking 2013 (ICOIN). IEEE, 2013, pp. 714–717.
[10] R. K. Kodali and S. Soratkal, “Mqtt based home automation system using esp8266,” in 2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC). IEEE, 2016, pp. 1–5.
[11] Eclipse Mosquito, “Eclipse Mosquito.” [Online]. Available: https://mosquitto.org/
[12] P. Kasinathan, G. Costamagna, S. Vikram, and H. AlQahtani, “A lightweight signature-based ids for iot environment,” arXiv preprint arXiv:1802.09089, 2018.
[13] N. U. Sheikh, H. Rahman, S. Vikram, and H. AlQahtani, “An ids framework for internet of things empowered by flowpan,” in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, 2013, pp. 1337–1340.
[14] A. Sforzin, F. G. Mármo1, M. Conti, and J.-M. Bohli, “Rpidps: Raspberry pi ids — a fruitful intrusion detection system for iot,” in 2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalComm/IoP/SmartWorld), 2016, pp. 440–448.
[15] Y. Mirsky, T. Doitschman, Y. Eloveci, and A. Shabat, “Kitsuone: an ensemble of autoencoders for online network intrusion detection,” arXiv preprint arXiv:1802.09089, 2018.
[16] A. Sforzin, F. G. Mármo1, M. Conti, and J.-M. Bohli, “Rpidps: Raspberry pi ids — a fruitful intrusion detection system for iot,” in 2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalComm/IoP/SmartWorld), 2016, pp. 440–448.
[17] Y. Oroum, D. Liu, and A. Nayak, “DI-ids: a deep learning–based intrusion detection framework for securing iot,” Transactions on Emerging Telecommunications Technologies, p. e3803, 2019.
[18] G. Bovenzi, G. Aceto, D. Cionzio, V. Persico, and A. Pescapé, “A hierarchical hybrid intrusion detection approach in iot applications,” in GLOBECOM 2020–2020 IEEE Global Communications Conference. IEEE, 2020, pp. 1–7.
[19] A. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli, “Passban ids: An intelligent anomaly-based intrusion detection system for iot edge devices,” IEEE Internet of Things Journal, vol. 7, no. 8, pp. 6882–6890, 2020.
[20] Q. Duan, D. Sun, G. Li, G. Yang, and W.-W. Yan, “Iot-enabled service for crude-oil production systems against unpredictable disturbance,” IEEE Transactions on Services Computing, vol. 13, no. 4, pp. 579–586, 2020.
[21] H. C. Hwang, J. Park, and J. G. Shon, “Design and implementation of a reliable message transmission system based on mqtt protocol,” in 2016 Future Network and Mobile Summit (UIC/ATC/ScalComm/IoP/SmartWorld), 2016, pp. 1765–1777, 2016.
[22] R. Kawaguchi and M. Bandai, “Edge based mqtt broker architecture for geographical iot applications,” in 2020 International Conference on Information Networking (ICOIN). IEEE, 2020, pp. 232–235.
[23] A. Detti, L. Funari, and N. Blefari-Melazzi, “Sub-linear scalability of mqtt,” in 2017 IEEE 12th International Conference on Cloud Computing (CLOUD). IEEE, 2017, pp. 456–463.
[24] A. Detti, L. Funari, and N. Blefari-Melazzi, “Sub-linear scalability of mqtt,” in 2017 IEEE 12th International Conference on Cloud Computing (CLOUD). IEEE, 2017, pp. 456–463.
D. Dinculeană and X. Cheng, “Vulnerabilities and limitations of mqtt protocol used between iot devices,” Applied Sciences, vol. 9, no. 5, p. 848, 2019.

I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, “Mqttsn, a new dataset for machine learning techniques on mqtt,” Sensors, vol. 20, no. 22, p. 6578, 2020.

H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, “Machine learning based iot intrusion detection system: an mqtt case study (mqtt-iot-ids2020 dataset),” in International Networking Conference – Springer, 2020, pp. 73–84.

P. Du and S. Abe, “Detecting dos attacks using packet size distribution,” in 2007 2nd Bio-Inspired Models of Network, Information and Computing Systems. IEEE, 2007, pp. 93–96.

S. N. Firdous, Z. Baig, C. Valli, and A. Ibrahim, “Modelling and evaluation of malicious attacks against the iot mqtt protocol,” in 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, 2017, pp. 748–755.

I. Vaccari, M. Aiello, and E. Cambiaso, “Slowite, a novel denial of service attack affecting mqtt,” Sensors, vol. 20, no. 10, p. 2932, 2020.

N. F. Syed, Z. Baig, A. Ibrahim, and C. Valli, “Denial of service attack detection through machine learning for the iot,” Journal of Information and Telecommunication, vol. 4, no. 4, pp. 482–503, 2020.

H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A. L. Munoz-Castañeda, I. García, and C. Benavides, “Multiclass classification procedure for detecting attacks on mqtt-iot protocol,” Complexity, vol. 2019, 2019.

The Raspberry Pi Foundation, “The Raspberry Pi Foundation.” [Online]. Available: https://www.raspberrypi.org/

Tshark, “Tshark.” [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html

Python, “Python.” [Online]. Available: https://www.python.org

Micropython, “Micropython.” [Online]. Available: https://micropython.org

J. D. Little and S. C. Graves, “Little’s law,” in Building intuition. Springer, 2008, pp. 81–100.

A. Q. S. Datasheet, “Air Quality Sensor Datasheet.” [Online]. Available: https://www.rhydoglabz.com/documents/MQ135_datasheet_m.pdf

P. I. Sensor, “Passive Infrared Sensor.” [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf

T. Sensor, “Temperature Sensor.” [Online]. Available: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

C. Sensor, “CO2 Sensor.” [Online]. Available: https://robu.in/wp-content/uploads/2017/12/MH-Z19-CO2-Ver1.0.pdf

X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip et al., “Top 10 algorithms in data mining,” Knowledge and information systems, vol. 14, no. 1, pp. 1–37, 2008.