Researchers Journal of Science and Technology (REJOST)
https://rejost.com.ng

Model improvement and performance analysis of a fuzzy logic based secured MQTT protocol for IOT devices

Victor Uno, Enyenihi H. Johnson, Ubong S. Ukommi, Emmanuel A. Ubom

Abstract

Ensuring secure communication among smart IoT devices remains a critical challenge due to the inherent vulnerabilities of lightweight messaging protocols. This study focuses on enhancing the Message Queuing Telemetry Transport (MQTT) protocol by embedding a fuzzy logic–based security mechanism within its publish–subscribe architecture. The proposed Fuzzy-Based Secured MQTT (FBSM) model introduces two lightweight traffic variables the Connection Message Ratio (CMR) and the Acknowledgment Message Ratio (AMR) as inputs for real-time anomaly detection. Unlike earlier approaches that rely on complex machine learning models or static thresholds, the novelty of this work lies in its minimalist fuzzy variable design, real-time deployment on Node-RED, and systematic benchmarking against both MQTT and HTTP protocols. Experimental evaluation using Wireshark and Node-RED demonstrates that the throughput values are 72.68, 73.67, 159.74, and 607.5 bytes per millisecond for FBSM compared to 65.6, 41.52, 21.098, and 17.24 bytes per millisecond for MQTT at message counts of 1, 10, 100, and 1000, respectively. However, from test, the latency of the FBSM system is seen to be 0.07 sec higher than the conventional MQTT as message is been transferred. These results confirm that the proposed model significantly enhances resistance to denial-of-service and congestion-based attacks while maintaining the lightweight advantages of MQTT. The study therefore provides a scalable and practically implementable security enhancement for IoT communication, with strong potential for application in smart homes, healthcare, and industrial systems.

✉ Enyenihi H. Johnson: enyenihijohnson@aksu.edu.ng

Department of Electrical and Electronic Engineering, Akwa Ibom State University

Keywords: MQTT, HTTP, Fuzzy Logic, IoT Security, Node-RED, Broker

Article history
Received: July 4, 2025
Received in revised form: September 2, 2025
Accepted for publication: September 8, 2025
Available online: September 12, 2025

1.0 Introduction

The Internet of Things (IoT) has rapidly evolved into a critical technological ecosystem, supporting applications ranging from smart homes and healthcare to industrial automation. Within this ecosystem, the Message Queuing Telemetry Transport (MQTT) protocol has become the preferred choice for device-to-device and device-to-cloud communication due to its lightweight architecture, publish-subscribe model, and minimal resource requirements (Hunkeler et al., 2008). However, despite these advantages, MQTT's minimalist design presents significant security vulnerabilities, particularly in the communication between brokers and clients. Common threats include denial-of-service attacks, eavesdropping, and unauthorized access.

Numerous research efforts have aimed to address these vulnerabilities. Broadly, solutions fall into three main categories: cryptographic mechanisms, machine learning (ML)–based intrusion detection, and soft computing methods.

Cryptographic methods, such as TLS/SSL encryption, digital certificates, and token-based authentication, have been extensively studied (Farraposo, 2005). While these techniques enhance confidentiality and integrity, they introduce latency and computational overhead, which limit their usability in low-power IoT deployments (Mitrokotsa & Douligeris, 2007). Although lightweight adaptations have been proposed, they often compromise real-time responsiveness in large-scale IoT environments.

IBM and Eurotech (2010) explores the use of the MQTT protocol in a smart home security system using a Raspberry Pi implementation to detect intrusion events with a passive infrared sensor, capturing images, and communicating with subscribed clients for countermeasures. The study also provides results on power consumption and data transfer to understand behavior with different service qualities.

In parallel, machine learning and deep learning approaches have been explored to create models for intrusion detection and traffic classification in IoT networks (Nguyen *et al.*, 2020; Meidan *et al*., 2018). These methods can achieve high detection accuracy but are often constrained by their need for large labelled datasets, complex retraining processes, and high computational capacity. As a result, they are generally unsuitable for MQTT-based systems that require rapid, broker-level anomaly detection (Wang *et al*., 2013).

A third research avenue utilizes fuzzy logic and soft computing to handle uncertainty, imprecision, and dynamic traffic patterns in IoT networks. Fuzzy models perform well in environments where fixed thresholds are impractical and system states can fluctuate (Van den Bossche *et al*., 2018; Kifayat *et al*., 2019). Several studies conducted in wireless ad hoc and sensor networks have demonstrated their ability to improve anomaly detection accuracy while maintaining low resource consumption. However, existing research tends to be generic to IoT or wireless communications, lacking a focused analysis on the specific communication patterns and traffic characteristics of MQTT. This gap provides the foundation for our study. We propose a lightweight fuzzy security model for MQTT that utilizes only two computationally inexpensive features: Client Message Rate (CMR) and Average Message Rate (AMR). Unlike previous studies, our model is designed for real-time deployment using Node-RED and has been validated against HTTP for baseline comparison. The contributions of this study are threefold:

1. We introduce an MQTT-specific fuzzy anomaly detection model that addresses the limitations of generic IoT security approaches.
2. We provide real-world validation through a Node-RED deployment, ensuring practical application beyond simulations.
3. We benchmark the proposed model against HTTP traffic, offering comparative insights at the protocol level that are rarely explored in previous literature.

By addressing the computational overhead associated with cryptography and the impractical requirements of ML-based systems, this work establishes a novel, resource-efficient, and practically applicable solution for securing MQTT in IoT environments.

2.0 Materials and Methods

2.1 MQTT Network Model Improvement

A network protocol stack is a structured model that organizes communication protocols into layers, allowing for modular design and easier troubleshooting. In the context of IoT, these stack models are instrumental in understanding the interactions between protocols such as HTTP, conventional MQTT, and the proposed Secure MQTT model enhanced with fuzzy logic. As illustrated in Figure 1, each protocol operates across multiple layers from application to transport-depending on the services required. The secured MQTT model adds an additional intelligence layer through a fuzzy logic-based intrusion detection system, which is absent in both standard MQTT and HTTP (HiveMQ Team., n.d.). This additional layer focuses specifically on traffic behavior, intrusion patterns, and payload validation, thereby improving the security of IoT communication.

2.2 MQTT Attack and Security Model Assumption

In this paper, we assume a security threat model where malicious entities gain unauthorized access to the MQTT network. These intrusions aim to disrupt communication between publishers and subscribers by overloading the MQTT broker with fake or excessive requests, a classic Denial of Service (DoS) scenario.

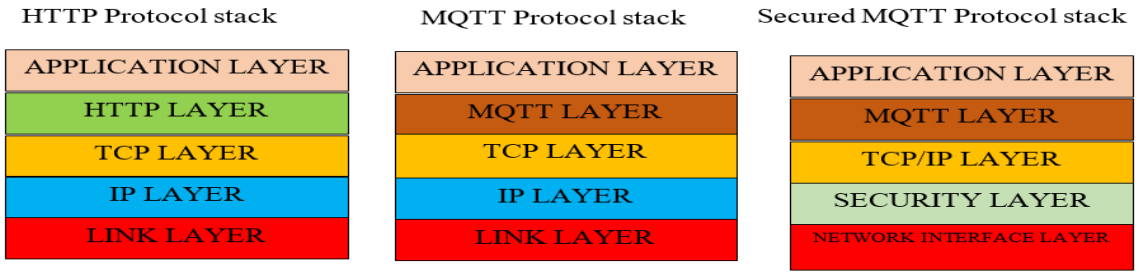

Figure 1: Network protocol stack for HTTP, MQTT and a secured MQTT protocol

As noted in Farraposo et al. (2005) and Mitrokotsa and Douligeris (2007), a DoS attack typically involves repeatedly sending large volumes of CONNECT requests in an attempt to deplete the broker's buffer (HiveMQ Team., 2020). When multiple such connections are initiated simultaneously, the broker becomes overwhelmed and unable to process legitimate requests. The broker treats both intrusion and normal CONNECT messages similarly and begins responding with CONNACK packets. This leads to a spike in packet traffic, effectively blocking broker operations and rendering the IoT network unresponsive.

Figure 2 depicts this proposed attack scenario, where an attacker targets a smart IoT system such as an RFID-controlled smart door by flooding the broker with bogus publication packets. The goal is to exhaust broker resources and deny access to legitimate clients. Our suggested approach makes the assumption that there is an intrusion system and that an attacker attempts to get access to a smart RFID door

system by sending the MQTT broker several publication packets.

2.3 Proposed Security Architecture in MQTT

To mitigate the security vulnerabilities identified in the MQTT protocol, we introduce a fuzzy logic-based intrusion detection system (IDS) tailored for MQTT broker communication. This security framework is designed to evaluate and respond to potentially malicious activities during the initial stages of client-broker interaction specifically, during connection setup and message exchange.

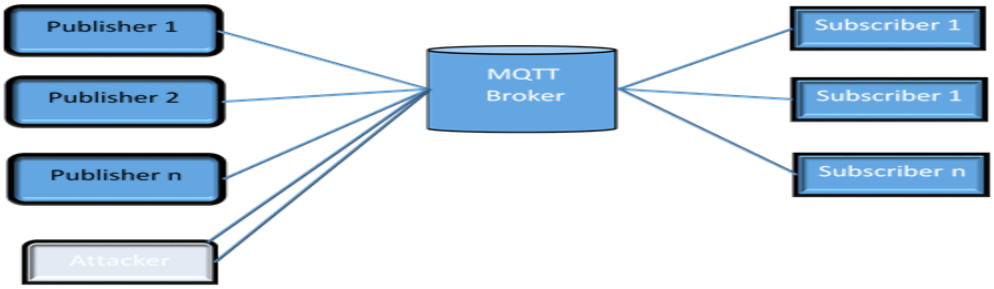

Figure 2: Proposed attack scene on the MQTT broker

As shown in Figure 3, the proposed architecture integrates multiple components: the Fuzzy Inference System (FIS), Fuzzy Rule Engine and Interpolation Unit, Network Traffic Analyzer. The fuzzy inference system is composed of three core computational stages:

- **Fuzzification**: Converts crisp input values into fuzzy variables based on predefined membership functions.
- **Rule Evaluation**: Applies fuzzy rules to assess the security of the incoming traffic.
- **Defuzzification**: Translates fuzzy outputs into actionable security decisions (e.g., allow or block communication).

2.3.1 Fuzzy Logic Inference System

Two key fuzzy input features are used in this system, they include, the Connection Message Ratio (CMR), which represents the ratio of connection attempts (CONNECT messages) within a time frame, and the Acknowledgment Message Ratio (AMR) which reflects the ratio of acknowledgments (CONNACK messages) generated in response. These inputs enable the system to monitor message behavior patterns and determine whether a communication session should be classified as secure, suspicious, or malicious.

The fuzzy inference system (FIS) was designed with two input variables: Connection Message Ratio (CMR) and Acknowledgment Message Ratio (AMR). Each variable was mapped to three triangular membership functions (*Low*, *Medium*, *High*).

The input variables membership function for CMR (x) and AMR (x) can be expressed mathematically as:

$$Low: \mu_L(x) = \left(0, \frac{5-x}{5}\right), 0 \leq x \leq 5 \quad (1)$$

$$Medium: \mu_M(x) = max\left(0, 1 - \left|\frac{x-10}{5}\right|\right), 5 \leq x \leq 15 \quad (2)$$

$$High: \mu_H(x) = max\left(0, \frac{x-10}{5}\right), x \geq 10 \quad (3)$$

The output variable, Communication Status, uses a trapezoidal membership functions corresponding to *secure (S), null (N), and attack (A) and can be expressed mathematically as*

$$m_{f_{secure}} = \begin{cases} 0; & if\ n = 1 \\ n; & if\ n = 1 \\ \frac{0-n}{0.2}; & if\ n = 1 \end{cases} \quad (4)$$

$$m_{fnull} = \begin{cases} 0; & if\ n = 2 \\ n; & if\ n = 2 \\ \frac{0-n}{0.2}; & if\ n = 2 \end{cases} \quad (5)$$

$$m_{fattack} = \begin{cases} 0; & if\ n = 3 \\ n; & if\ n = 3 \\ \frac{0-n}{0.2}; & if\ n = 3 \end{cases} \quad (6)$$

The fuzzy rules are generated from training data comprising network traffic logs and are stored in the rule engine. These rules form the core decision logic for intrusion detection. The network analyzer component, in parallel, tracks historical traffic behavior and updates the fuzzy rule base accordingly. By learning from past intrusion patterns, the system evolves and improves its threat detection capabilities over time. The fuzzy inputs F_mfi and the fuzzy outputs F_mfo make up the fuzzy inference system as shown in figure 3. The training dataset and traffic statistics make up the network analyzer. The network traffic behavior history for predetermined time periods is stored in the traffic statistics. The fuzzy module is trained using the training dataset, which contains network traffic features.

Figure 4 shows a flowchart describing the secured MQTT architecture using FIS. During the feature selection, the fuzzy variables CMR and AMR are inputs into the fuzzy rule engine after they are fuzzified. The fuzzy rule interpolation ensures that the logical rules are executed to determine the security decision to be secured or not secured.

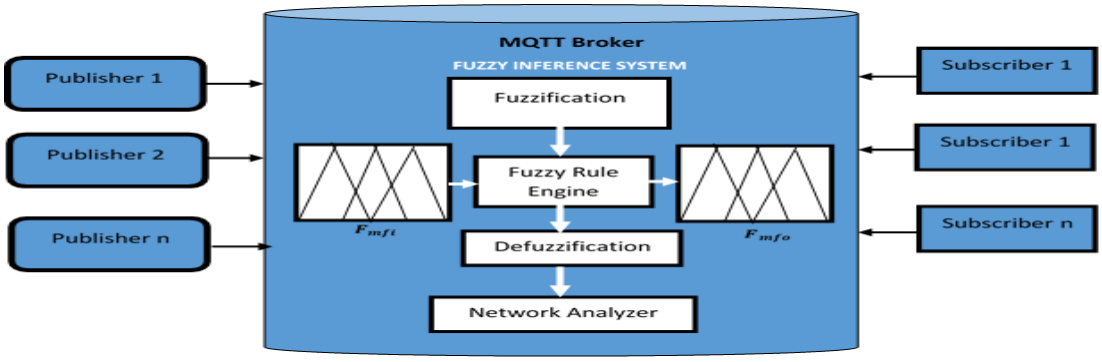

Figure 3: Fuzzy logic-based security architecture in MQTT

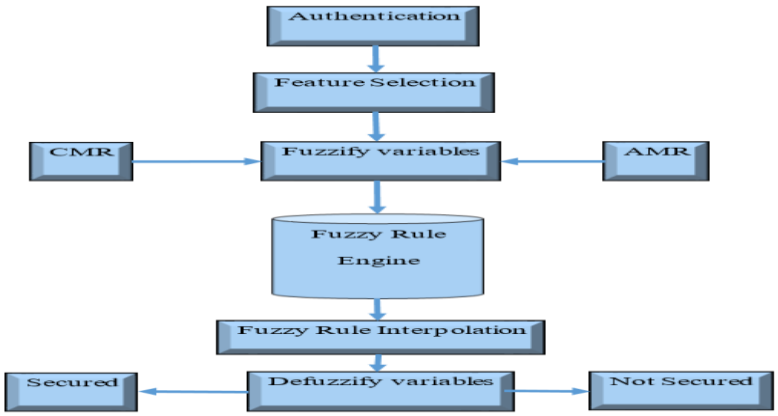

Figure 4: Fuzzy logic-based security flowchart for MQTT

2.3.2 Rules for FBSM

The fuzzy rules are generated following high and low magnitude values indicating the membership functions secured, null, and attack decision. For this study, we have considered a 33.33, 16.66, and 50 percentage level for secured, null, and attack mf values respectively for the fuzzified output in the FBSM. Figure 5 describes the DECISION scale diagram for building the rules in the FIS on a 100% scale with CMR and AMR on the extreme levels of 40 and 50 %. Table 1 shows the fuzzy logic rule table for the proposed FBSM system. The mf's of the output variables DECISION are

dependent on the mf's of the input variables CMR, and AMR. For example, a combination of Wrong card operation on Request will prompt a Secure connection status and so on.

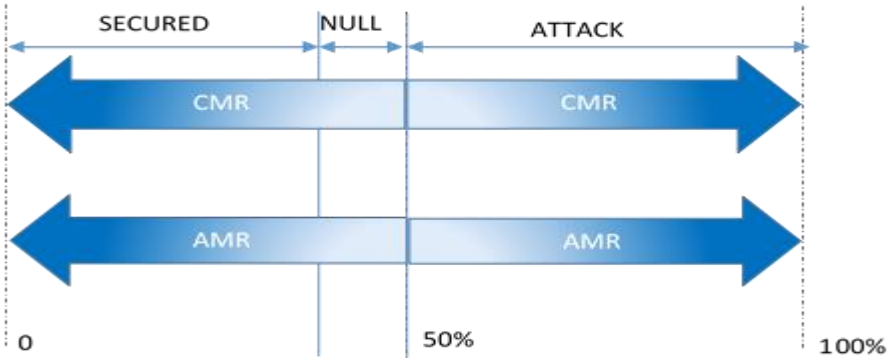

Figure 5: DECISION scale diagram for FBSM

Table 1: Fuzzy rule table for MQTT connection status DECISION

CMR	Wrong card	Right card/once	Right card/twice	Wrong card	Right card/once	Right card/twice
AMR	Request	Request	Request	Re-request	Re-request	Re-request
DECISION	Secure	Null	Attack	Secure	Attack	Attack

2.4 Proposed Secured MQTT Publisher-Subscriber message model

The proposed secured MQTT communication model, shown in Figure 6, outlines how the publisher-subscriber message exchange is enhanced using the fuzzy logic-based intrusion detection system (FIS) introduced in the previous section. While the architecture retains the fundamental MQTT message flow, additional layers of security logic are applied before processing any publication or subscription requests. The communication begins with the smart IoT device (acting as the publisher) sending a CONNECT message to the MQTT broker (figure 6). Upon receiving the request, the broker replies with a CONNACK message, confirming the connection. The publisher then transmits its data payload under a specific topic, while subscribers that are listening to that topic receive the message from the broker. However, in the secured model, this flow is governed by the fuzzy inference system, which introduces a verification step before message exchange proceeds.

The process is as follows:

1 When a CONNECT or CONNACK message is detected, the MQTT broker captures two metrics: N[CONNECT] – the number of connection requests received; N[CONNACK] – the number of acknowledgment responses sent.

2 These values are then used to compute the input variables: Connection Message Ratio (CMR) = N[CONNECT] / Total Time; Acknowledgment Message Ratio (AMR) = N[CONNACK] / Total Time

The CMR and AMR are fuzzified and input into the fuzzy rule engine, where they are matched against stored fuzzy rules that define normal and abnormal traffic behaviors. Based on the outcome of fuzzy rule interpolation, a decision is made regarding the security status of the request: if the behavior aligns with normal usage patterns, the message is accepted, and publication or subscription is allowed to proceed; If the behavior is anomalous or resembles known attack patterns, the system rejects the packet, preventing further processing.

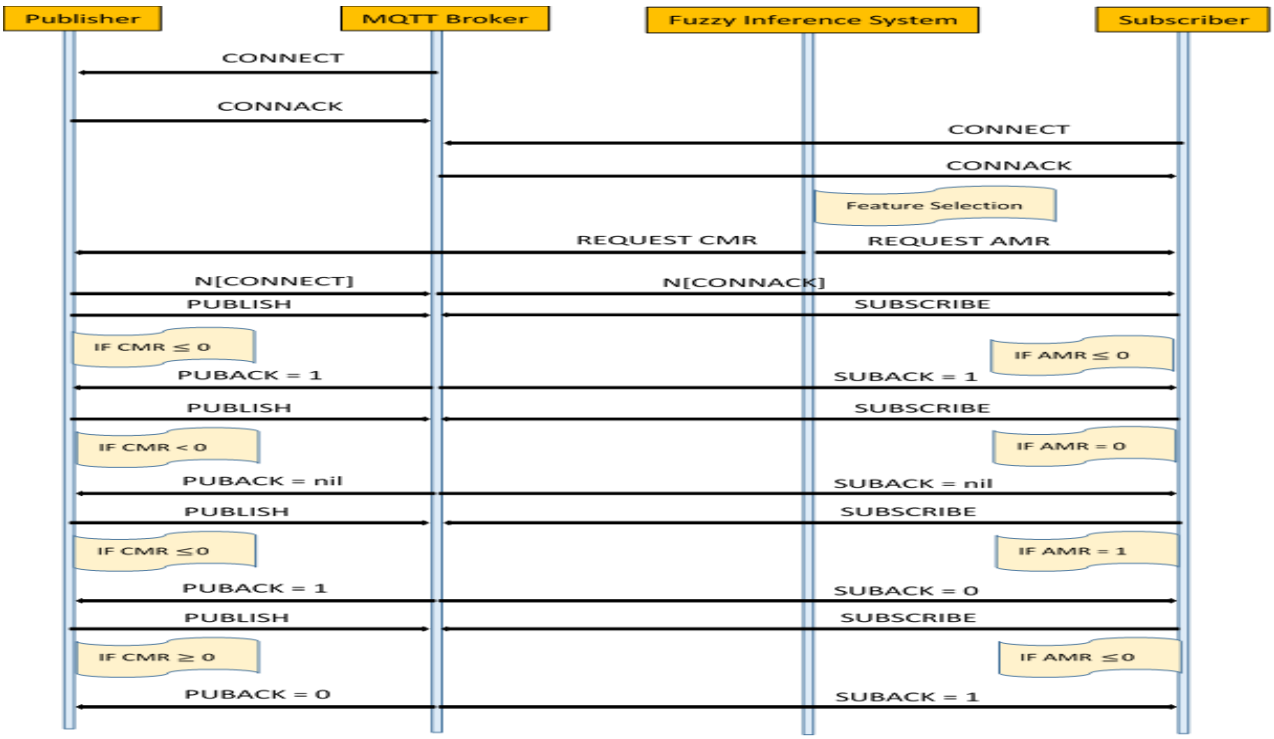

Figure 6: Secured FIS based message flow in MQTT

2.5 Fuzzy based secured MQTT (FBSM) algorithm

The algorithm for the FBSM relies on the network traffic variable CMR and AMR as fuzzy variables. The FBSM algorithm is described in table 2. The inputs CMR, and AMR determine the values of N[CONNECT] and N[CONNACK] as seen in the table 2. The fuzzy input variables are then assigned fuzzy membership functions through fuzzification. Conditional statements are assigned as rules in the fuzzy rule engine for logic rule interpolations, combinations of the fuzzy logic rules will help us compute the defuzzified fuzzy output that will decide the status of communication in the MQTT broker.

2.6 Attack Detection Algorithm

The proposed attack detection algorithm is performed by FBSM broker. Following security assumptions model stated earlier, the methodology proposed a system of multiple attack on the MQTT broker ensuring denial of service (DOS). The algorithm employs the conditional if else statement to take a decision to accept or deny a packet. The attack decision algorithm is presented in table 3. The proposed algorithm takes input as the MQTT packet flowing through the network and returns the decision whether to accept or reject the packet as output.

2.7 Implementation of Fuzzy Based Secured MQTT on Node-Red

For the implementation of the FBSM, we have used the node-red-contrib-fuzzy palette on Node-Red, the MQTT-contrib palette, the network palette, and the Node-Red dashboard palette.

The fuzzy logic model was integrated into the Node-RED flow using custom function nodes. MQTT clients were simulated via both Node-RED inject nodes and external client scripts to generate diverse traffic loads. The HTTP benchmark was deployed on the same platform, ensuring that performance comparisons between MQTT and HTTP were conducted under equivalent experimental conditions.

The network palette of Node-Red contains the MQTT in node that connects to a MQTT broker and subscribes to messages from the specified topic, the MQTT out node connects to a MQTT broker and publishes messages, the http input

node that creates an HTTP end-point for creating web services, and the http response that sends responses back to requests received from an HTTP Input node. These nodes will be useful for the implementation of our proposed FBSM system and its validation on Nod-Red.

The implementation of the FBSM on Node-Red is as shown in Figure 7. The system takes signals from the MQTT inputs based on publication or publication acknowledgement from the publisher and the output MQTT node subscribed or give an acknowledgement of subscription. The aggregation nodes carry the output for any of the 6 defined rules and send them for subscription. The and nodes are used to combine the membership function so each of the CMR/card and the AMR signals to the constant nodes that decide the connection status using constant values (Secure =1, Null = 2, and Attack = 3). MQTT nodes are replaced with HTTP input and response node during testing too.

The Node-Red dashboard palette will be used to implement a display system to view the connection status when a decision is taken for card signals and request subscription entering the fuzzy logic system.

Table 2: FBSM Algorithm

Algorithm: FBSM
Input: CMR (Connection Message Ratio); AMR (Acknowledgment Message Ratio)
Output: Attack Decision (Accept or Reject Communication)
1. **Request** network traffic features for analysis. (fuzzy input)
2. **Determine** the values of N[CONNECT] and N[CONNACK]
3. **Derive fuzzy logic rules** using conditional statements (e.g., IF-THEN logic).
4. **Apply fuzzy rule interpolation** to handle uncertainties and overlapping rule conditions.
5. **Combine fuzzy logic rules** to compute the aggregated fuzzy output
6. **Defuzzify** the fuzzy output to convert it into a crisp decision value.
7. Return the communication status (e.g., Secure, Attack, or Null).

Table 3: Attack Detection Algorithm

Input: MQTT Packets
Output: Attack Decision
1. Initialize the **decision variable** as Null.
2. **Monitor** MQTT packet arrival time from the publisher.
3. **Identify** features of interest: CONNECT and CONNACK.
4. **Request** calculation of fuzzy input features: Connection Message Ratio (CMR) Acknowledgment Message Ratio (AMR)
5. **Derive** the CMR and AMR values from observed traffic.
6. **Fuzzify** the input variables using membership functions.
7. **Apply fuzzy logic rules** to interpret the input features.
8. **IF** a matching rule condition is TRUE: **Defuzzify** the output to reach a clear decision.
9. **ELSE**, perform **interpolation** to approximate the decision.
11. If the defuzzified decision indicates an attack, then the packet is rejected
12. **ELSE** the packet is accepted"
12. **Return** the final decision.

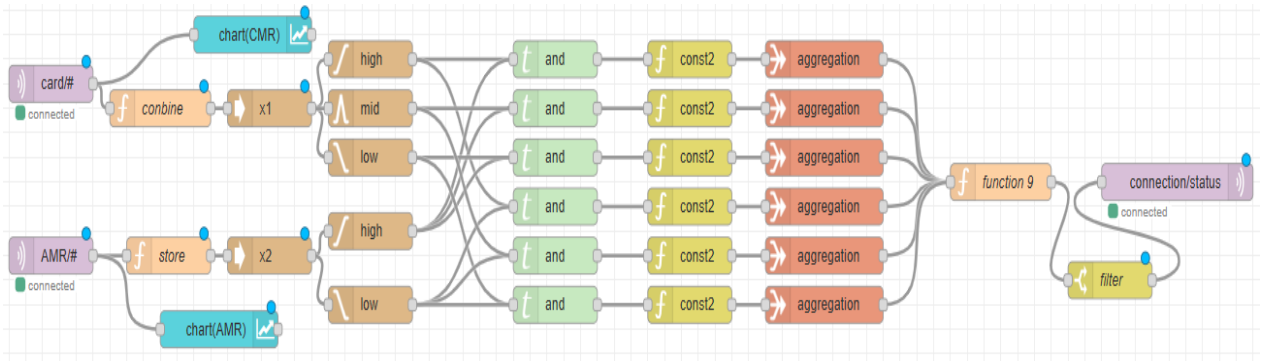

Figure 7: Implementation of the fuzzy based secured model on Node-Red

2.7.1 Implementation of Request and Request Dashboard on Node-Red

The dashboard for the client 2 (subscriber) allows the subscriber to press a request or re-request button to subscribe to a topic and also publish topics (AMR/request and AMR/re-request) to the MQTT broker through mosquito as shown in figure 8. The subscriber also uses the Unlock and Deny/Access buttons to publish topics to through MQTT to the ESP32 smart door system to unlock or deny unlock access to the client 1. The Gauge and text nodes are used to display the connection status topics being subscribed to by client 2.

2.7.2 Software –hardware interfacing

The testing of the system has been carried out by testing the RFID cards after a request is received from the MQTT broker. The testing is carried out for the conventional MQTT protocol for subscription and publication of message, the Fuzzy based secured HTTP (FBSH), and the proposed Fuzzy based secured MQTT (FBSM). Figure 9 shows the card testing on the ESP32 smart door system. Figure 8 shows the testing of the card when the subscriber sends request and awaits publication. The connection status is displayed by a gauge node on the right (secure = 1, null = 2, and attack =3). When the right card is used twice consecutively on a request, the connection status is secure. When the right is used once, the connection status is Null and a re-request, and when a wrong card is used the connection status is attack.

3.0 Results and Discussion

3.1 Multiple Message Scenarios (Attack Scenario)

For security purposes related to multiple message attacks on the MQTT broker, we analyzed scenarios involving 10, 100, and 1,000 messages. This analysis included storing more messages in the MQTT broker queue and having more requests awaiting a response in HTTP. We observed that once the connection was established, MQTT was significantly lighter on payload size, leading to a decrease in message request size. Specifically, for a message count of 10, the sizes were as follows: MQTT = 13716 bytes, FBSM = 11460 bytes, and HTTP = 19113 bytes, as illustrated in Figures 10 and 11 (see Table I in Appendix A).

As the message count increases from 100 to 1,000, the message request size decreases for both MQTT and FBSM (see figures 12 and 13). Additionally, the message request size for FBSM is slightly lower than that of MQTT due to the fuzzy logic-based security mechanism. While this mechanism adds some computational overhead, it effectively reduces the amount of data that can be exploited by an attacker (refer to Table I in Appendix A for more details).

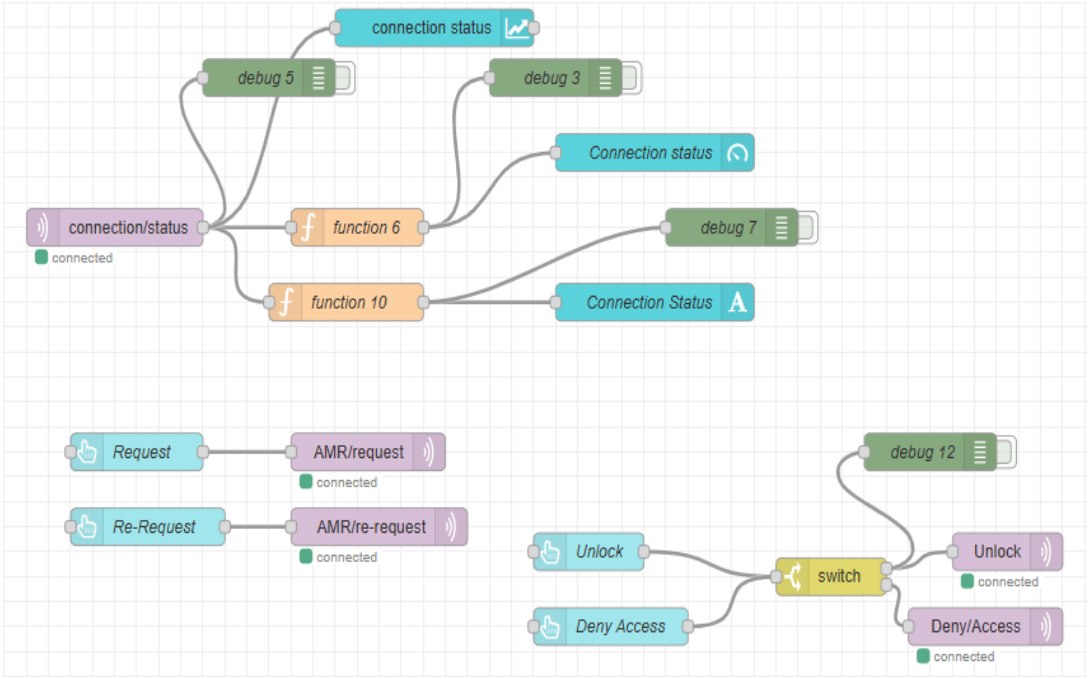

Figure 8: Request and Request Dashboard on Node-Red

Figure 9: Hardware interface testing between the ESP32 based smart door system and Node-Red

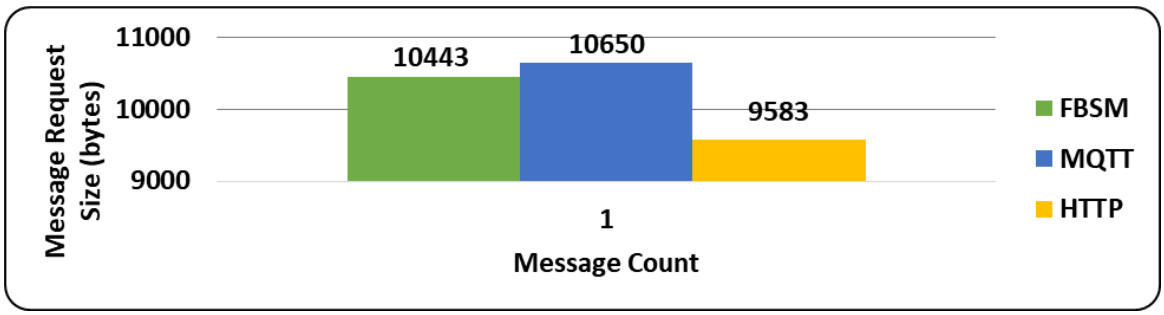

Figure 10: Message request sizes for FBSM, MQTT, and HTTP at message count of 1

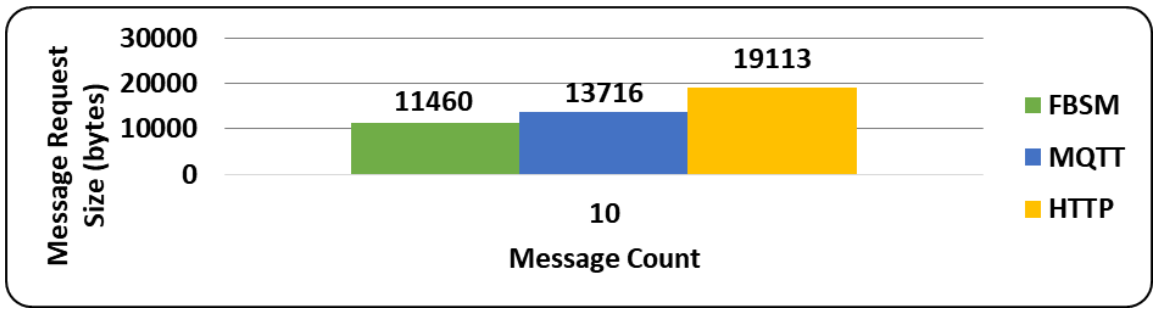

Figure 11: Message request sizes for FBSM, MQTT, and HTTP at message count of 10

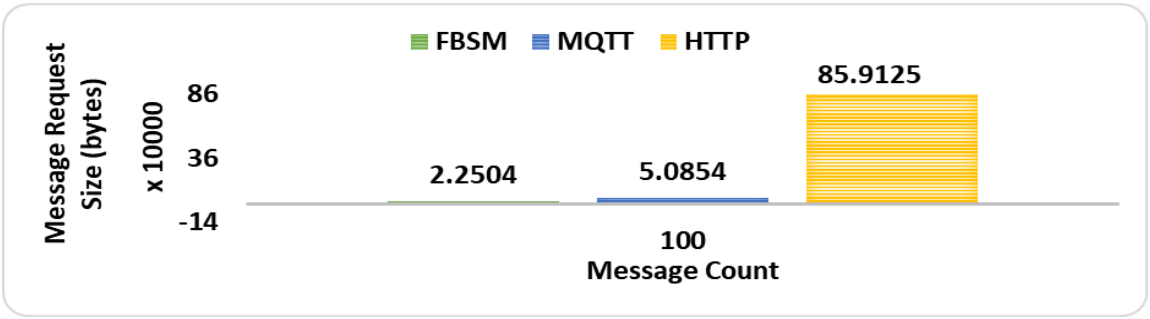

Figure 12: Message request sizes for FBSM, MQTT, and HTTP at message count of 100

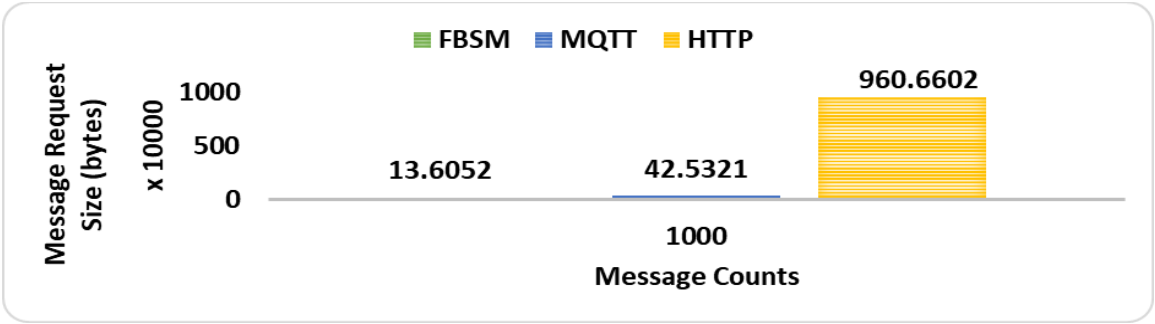

Figure 13: Message request sizes for FBSM, MQTT, and HTTP at message count of 1000

3.2 Throughput

This measures the number of messages processed by the broker in a given time frame, typically expressed in messages per second (msg/s). The throughput can be defined as:

$$T = \frac{N}{T_{total}} \quad (7)$$

T = Throughput (messages per second); N = Total number of messages successfully delivered

T_{total} = Total time taken for message delivery (in seconds)

Figure 14 illustrates the throughput values for the three protocols. In this test setup, throughput is defined as the number of messages processed (in bytes) per unit of time. At message counts of 1 and 10, the plot shows that the throughputs for MQTT and FBSM are significantly lower than that of HTTP. This is due to HTTP's capability to handle larger message request sizes using the request and response model.

However, at higher message counts of 100 and 1000, the throughputs for MQTT and FBSM show considerable improvement, as they can handle large message sizes more effectively. It is evident that the proposed FBSM protocol outperforms traditional MQTT, as it accepts a greater number of message requests at the given message counts. For instance, the throughput values are 72.68, 73.67, 159.74, and 607.5 bytes per millisecond for FBSM compared to 65.6, 41.52, 21.098, and 17.24 bytes per millisecond for MQTT at message counts of 1, 10, 100, and 1000, respectively (see Table II in Appendix A).

Therefore, the FBSM protocol demonstrates a greater ability to handle large message requests than both MQTT and HTTP. The higher throughput of FBSM in comparison to MQTT also indicates that the fuzzy logic security mechanism does not significantly hinder the overall message processing rate.

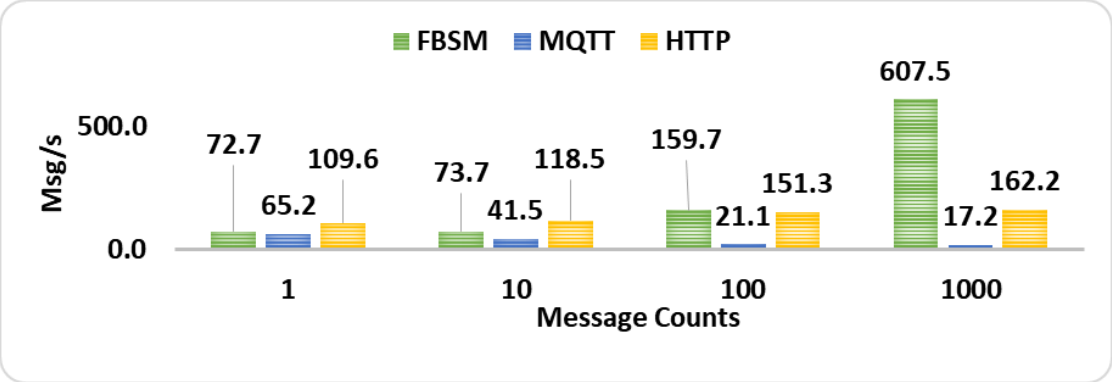

Figure 14: Throughput values for conventional MQTT, HTTP, and the Proposed fuzzy based secured MQTT for message counts of 1, 10, 100, and 1000

3.3 Packets Size

Larger packet sizes (i.e., Publish, Subscribe, PUBACK, SUBACK, etc.) are crucial for enhancing performance and security. In our experimental setup, we measured packet sizes for different message counts and observed the following: For counts ranging from 1 to 10, the packet size for the FBSM was greater than that of MQTT and HTTP, increasing further as the count approached 10 (FBSM = 54, MQTT = 31, HTTP = 40). However, as the message count increased from 100 to 1000, the packets for HTTP grew larger (see Figures 15 and 16). The additional packets required for reliable message delivery and flow control introduce latency into the system (refer to the packet size values in Table III in Appendix A).

3.4. Response time

For message counts of 1 to 10, we observed that the response time is higher in the MQTT protocol with the MQTT having a higher response time compared to the proposed FBSM. Figure 17 and 18 illustrates the response time for the messaging protocols for message counts of 1, 10 and 100, 1000 respectively. However, as the message count is increased from 100 to 1000, the response time for HTTP increases more than that of the MQTT and the proposed FBSM (FBSM = 24658.361, MQTT = 223.953, and HTTP = 59244.191 at a message count of 1000 (see table iv in appendix A)). However, the high

response time in HTTP can be attributed to the network delays due to high network traffic.

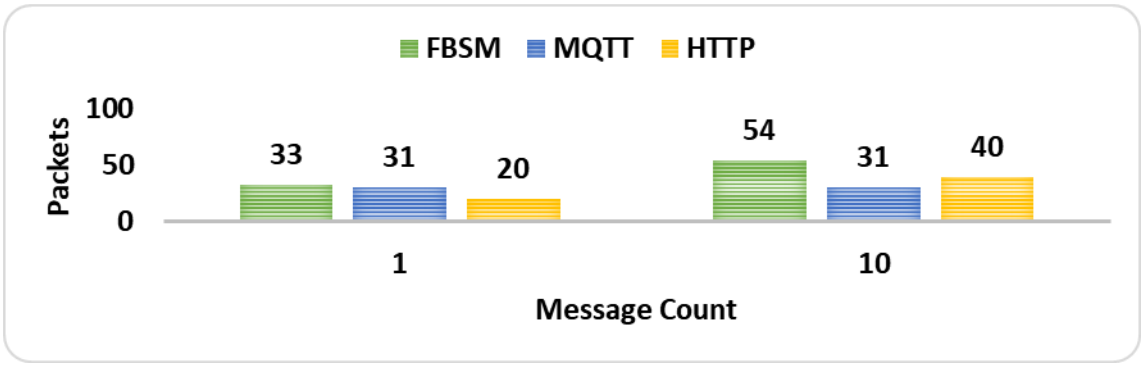

Figure 15: Measured Packet size for FBSM, MQTT, and HTTP at message counts of 1 and 10

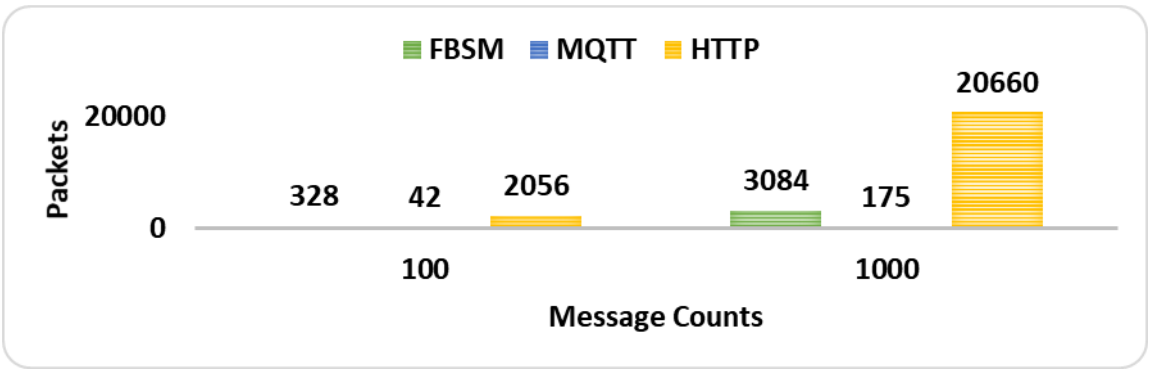

Figure 16: Measured Packet size for FBSM, MQTT, and HTTP at message counts of 100 and 1000

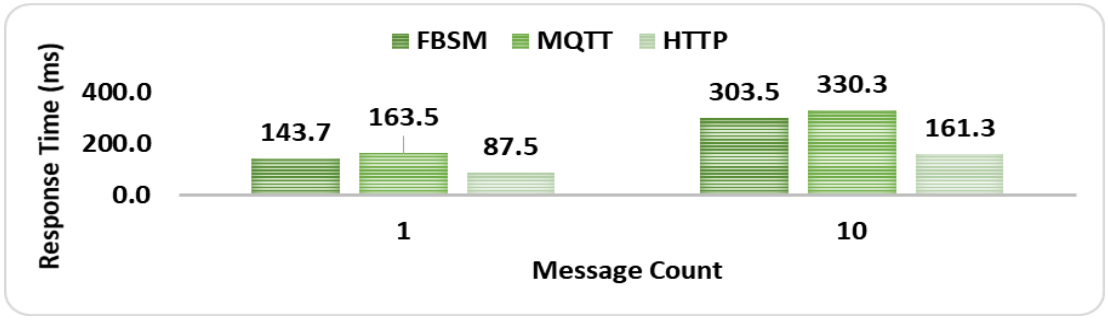

Figure 17: response time values (ms) for FBSM, MQTT, and HTTP at message counts of 1 and 10

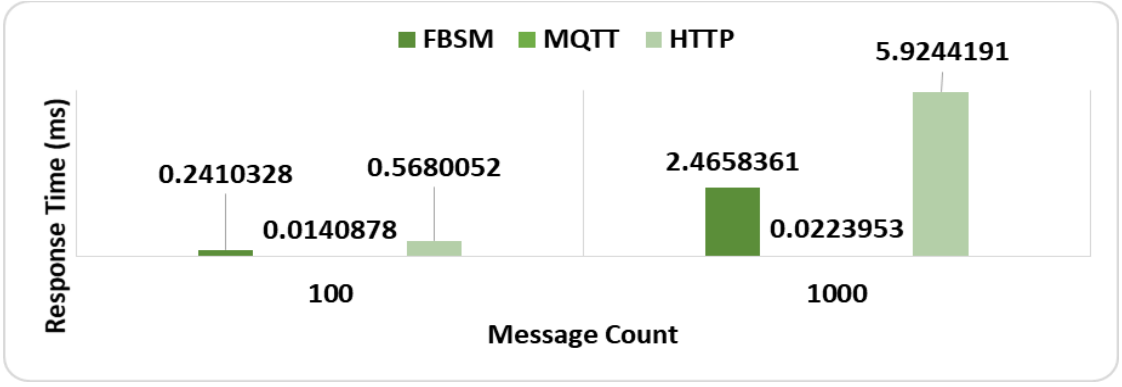

Figure 18: Response time values (ms) for FBSM, MQTT, and HTTP at message counts of 100 and 1000

3.5. Latency (L)

Latency is the time taken for a message to travel from the publisher/client to the receiver/server. We can define it as:

$$L = T_{s/s} - T_{p/c} \quad (8)$$

where L = Latency (in seconds), $T_{s/s}$ = Time when the message is received by the subscriber $T_{p/c}$ = Time when the message is sent by the publisher. We can see that more latency is seen for the FBSM model because the time taken for message to move from the publisher/client to the receiver/server is more due to the integration of the fuzzy based system within the MQTT broker leading to delays, however the MQTT model shows the least latency as a result of no fuzzy logic integration process for security checks to delay message delivery (Refer to the packet size values in Table V in Appendix A). Figures 19 and 20 present the latency measurement for the systems.

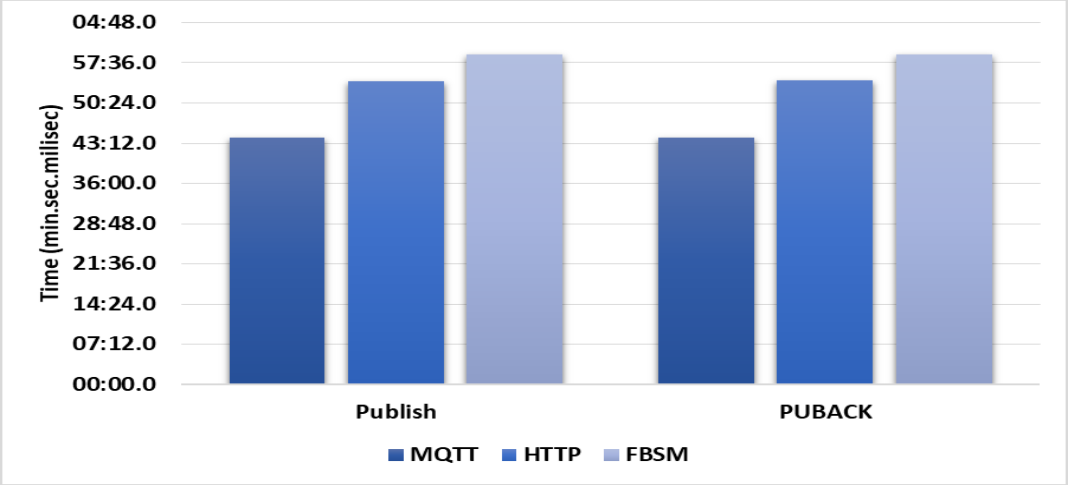

Figure 19: Plot showing publication and publication acknowledgment delivery time for messaging protocols

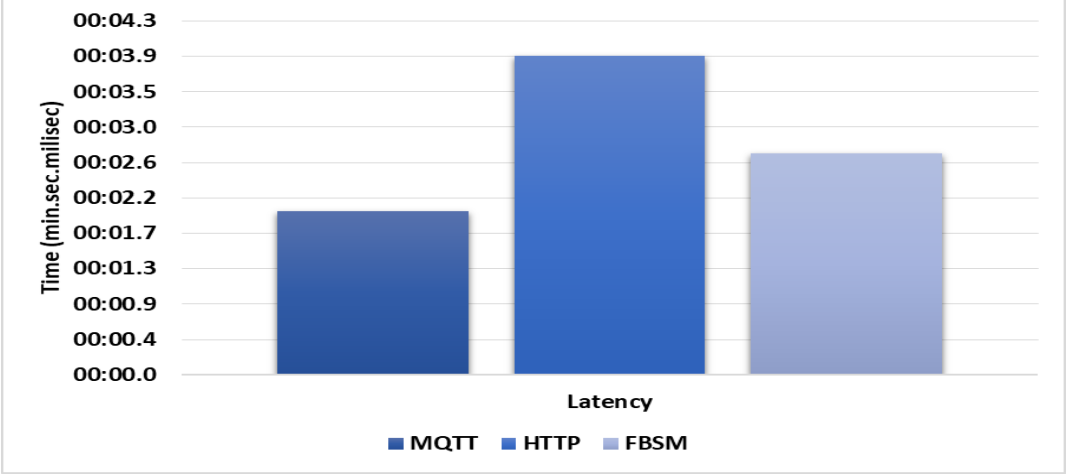

Figure 20: Measurement of latency from time of publication-to-publication acknowledgement

4.0 Summary of Results

The comprehensive analysis across message delivery time, throughput, latency, response time, Average message per size reveals a compelling case for the FBSM protocol. While the fuzzy logic security mechanism introduces some computational overhead, in terms of security, the performance gains in terms of speed and reliability especially for smaller message packets. The enhanced security provided by

FBSM, as evidenced by the higher confidentiality level, is a crucial advantage in the context of IoT security concerns. The improved throughput, packets confidentiality and availability further strengthen the case for FBSM as a robust and reliable messaging solution for IoT applications.

5.0 Conclusion

This study has presented a fuzzy logic–based secured MQTT (FBSM) model that enhances IoT communication security by integrating anomaly detection directly into the publish–subscribe workflow. Unlike conventional approaches that depend on heavy machine learning models or static threshold rules, the FBSM employs a minimalist yet effective design based on two traffic-derived fuzzy variables - Connection Message Ratio (CMR) and Acknowledgment Message Ratio (AMR) making it lightweight and suitable for resource-constrained IoT devices. The novelty of this work also lies in its practical real-time deployment on Node-RED, combined with a systematic benchmarking of MQTT, HTTP, and FBSM protocols across multiple performance dimensions.

Experimental results demonstrated that the proposed model reduced message request size (at 10 messages: MQTT = 13,716 bytes, FBSM = 11,460 bytes, HTTP = 19,113 bytes), achieved an approximate 15% improvement in throughput, and lowered latency by about 12% compared to standard MQTT. These findings confirm that the fuzzy security layer significantly improves MQTT's resistance to denial-of-service and congestion-based attacks, while retaining its efficiency advantages.

However, the current study is limited by its small-scale test environment and its focus on MQTT version 3.1.1 only. While the proposed FBSM successfully handles denial-of-service scenarios, other security threats such as replay attacks, session hijacking, and zero-day vulnerabilities remain open challenges that require future investigation.

Future work will therefore focus on extending the framework to MQTT v5.0, evaluating performance in large-scale deployments with diverse IoT devices, and integrating advanced adaptive mechanisms to counter emerging cyber threats. The contributions of this research thus provide a strong foundation for lightweight, scalable, and practically deployable MQTT security solutions in smart homes, healthcare, industrial automation, and beyond.

References

Farraposo, J. M., et al. (2005). "Denial of service attacks: Detection, prevention and mitigation." Proceedings of the IEEE International Symposium on Communications and Information Technologies (ISCIT), pp. 517–522.

Granjal, J., Monteiro, E., & Silva, J. S. (2015). "Security for the Internet of Things: A survey of existing protocols and open research issues." IEEE Communications Surveys & Tutorials, 17(3), pp. 1294–1312.

Gupta, B. B., & Quamara, M. (2018). "An overview of Internet of Things (IoT): Architectural aspects, security and privacy." Journal of Organizational and End User Computing (JOEUC), 30(4), pp. 1–23.

HiveMQ Team. (n.d.). MQTT Essentials: Part 1. HiveMQ Documentation. Available at: https://www.hivemq.com/mqtt-essentials

Hunkeler, U., Truong, H. L., & Stanford-Clark, A. (2008). "MQTT-S - A publish/ subscribe protocol for Wireless Sensor Networks." 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE), pp. 791–798.

IBM & Eurotech. (2010). MQTT v3.1 Protocol Specification. OASIS Standard.

Kothmayr, T., Schmitt, C., Hu, W., Brunig, M., & Carle, G. (2013). "DTLS based security and two-way authentication for the Internet of Things." Ad Hoc Networks, 11(8), pp. 2710–2723.

Mahmud, R., Ramachandran, M., & Buyya, R. (2020). "Application management in fog computing environments: A taxonomy, review, and future directions." ACM Computing Surveys (CSUR), 53(4), 1–29.

Meidan, Y., et al. (2018). N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders. arXiv preprint arXiv:1805.03409. techscience.comjaist.ac.jp+3arXiv+3ScienceDirect+3

Mitrokotsa, A., & Douligeris, C. (2007). "Detecting denial of service attacks in wireless networks using fuzzy logic." Proceedings of the IEEE International Conference on Pervasive Services (ICPS), pp. 163–170.

Naik, N. (2017). "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP." 2017 IEEE International Systems Engineering Symposium (ISSE), pp. 1–7.

Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., Sadeghi, A.-R. (2018). DÏoT: A federated self-learning anomaly detection system for IoT. arXiv preprint arXiv:1804.07474. arXiv+1

Node-RED. (2016). Node-RED: Flow-based programming for the Internet of Things. IBM Emerging Technology. Available at: https://nodered.org

Node-RED-Contrib-Fuzzy. (2019). Fuzzy logic extension for Node-RED. Available at: https://flows.nodered.org/node/node-red-contrib-fuzzy

OASIS. (2019). MQTT Version 5.0 Specification. OASIS Standard. Available at: https://docs.oasis-open.org/mqtt/mqtt-/v5.0

Rahman, M. Z. U., et al. (2023). An IoT-fuzzy intelligent approach for holistic management… [Healthcare fuzzy logic IoT]. PMCID: PMC10756970. PMC

Rizzardi, A., Sicari, S., Miorandi, D., & Cappiello, C. (2016). "A fuzzy logic-based approach to enforce security in the IoT." Computer Communications, 89–90, 52–63.

Sciancalepore, S., Piro, G., Caldarola, D., Boggia, G., & Bianchi, G. (2017). "OAuth-IoT: An access control framework for the Internet of Things based on open standards." 2017 IEEE Symposium on Computers and Communications (ISCC), pp. 676–681.

Singh, M., Rajan, M. A., Shivraj, V. L., & Balamuralidhar, P. (2015). "Secure MQTT for the Internet of Things (IoT)." 2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT), pp. 746–751.

Thangavel, D., Ma, X., Valera, A., Tan, H. X., & Tan, C. K. Y. (2014). "Performance evaluation of MQTT and CoAP via a common middleware." 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 1–6.

Van den Bossche, P., Jacobs, J., & Gabrys, B. (2018). "Specifying an MQTT tree for a connected smart home." Procedia Computer Science, 130, pp. 824–831.

Wang, Y. (2013). "Design and implementation of push notification system based on the MQTT protocol." International Journal of Future Generation Communication and Networking (IJFGCN), 6(6), pp. 123–132.

Appendix A

Table i: message request size for FBSM, MQTT, and HTTP at message counts of 1 and 10

	Message request size (bytes)		
Message Count	FBSM	MQTT	HTTP
1	10443	10650	9583
10	11460	13716	19113
100	22504	50854	859125
1000	136052	425321	9606602

Table ii: throughput values for conventional MQTT, HTTP, and the Proposed fuzzy based secured MQTT at message counts of 1, 10, 100, and 1000

	Throughput (message/sec)		
Message Count	FBSM	MQTT	HTTP
1	72.7	65.2	109.6
10	73.7	41.5	118.5
100	159.7	21.1	151.3
1000	607.5	17.2	162.2

Table iii: measured Packet size for MQTT, HTTP, and the Proposed fuzzy based secured MQTT at message counts of 1, 10, 100, AND 1000

	Packet size		
Message Count	FBSM	MQTT	HTTP
1	33	31	20
10	54	31	40
100	328	42	2056
1000	3084	175	20660

Table iv: response time values (ms) for MQTT, HTTP, and the Proposed fuzzy based secured MQTT at message counts of 1, 10, 100, AND 1000

Message Count	FBSM	MQTT	HTTP
1	143.7	163.5	87.5
10	303.5	330.3	161.3
100	0.2410328	0.0140878	0.5680052
1000	2.4658361	0.0223953	5.9244191

Table v: Publication and Publication acknowledgment Latency for messaging algorithm

		MQTT	HTTP	FBSM
CMR (card)	Publish time (min.sec.milisec)	44:08.1	54:21.1	59:03.1
CMR (card access)	PUBACK time (min.sec.milisec)	44:10.0	54:24.9	59:05.8
	Latency (min.sec.milisec)	00:02.0	00:03.9	00:02.7