Ingénierie des Systèmes d'Information
Vol. 30, No. 8, August, 2025, pp. 1963-1973
Journal homepage: http://iieta.org/journals/isi

S-MQTT: A Secure MQTT Protocol with Merkle Tree Authentication and AES Encryption for IoT Communication Systems

Nilima Tatyasaheb Dhokane[1*], Santosh Jagtap[1], Binod Kumar[2], Amit Anand[3], Rajesh Kumar Pandey[4]

[1] Department of Computer Science, Savitribai Phule, Pune University, Pune 411044, India
[2] MCA Department, JSPM's Rajarshi shahu college of Engineering, Pune 411033, India
[3] Independent Researcher, Austin TX, USA
[4] Independent Researcher, WA, USA

Corresponding Author Email: nilima.d28april@gmail.com

https://doi.org/10.18280/isi.300803

Received: 14 July 2025
Revised: 16 August 2025
Accepted: 22 August 2025
Available online: 31 August 2025

Keywords:
encryption, IoT, MQTT security, MQTT, security, lightweight encryption

ABSTRACT

As the number of Internet of Things (IoT) devices grows, there is a greater need for secure and efficient communication protocols. A growing number of people are using the Message Queuing Telemetry Transport (MQTT) protocol because of its real-time and lightweight data sharing capabilities. However, security concerns, particularly in scenarios involving the transmission of sensitive information, necessitate the development of augmented security measures. This research introduces a pioneering protocol, Secured MQTT (S-MQTT), designed to address vulnerabilities inherent in the traditional MQTT protocol. To protect the confidentiality, integrity, and authenticity of transmitted data, S-MQTT combines sophisticated encryption methods with access control and authentication protocols. The proposed system S-MQTT in this research employs the MQTT protocol for data transfer within a communication system, comprising three key components: Publisher, Broker, and Subscriber. The study focuses on optimizing time-consuming procedures within the system and fortifying data security in communication systems. Using a Watchdog timer and AES data security, the investigation seeks to assess the broker's dependability in terms of activity level. Comparative analysis of the proposed system against the current system demonstrates superior performance. The results shows that the proposed protocol achieved an overall mitigation efficiency of 97.78%, completely blocking man-in-the-middle attacks and reducing malware intrusions by 96.61%. Encryption and authentication added only minimal latency and moderate resource overhead while significantly enhancing confidentiality, integrity, and availability. Including these metrics in the abstract will provide a balanced view of both the security effectiveness and performance trade-offs of S-MQTT. Additionally, the study presents an assessment of the time and space complexity of the suggested system design.

1. INTRODUCTION

The widespread use of real-time communication systems and connected devices in the current digital era has completely transformed the way data is shared across networks. In order to enable intelligent and effective operations, data connectivity is essential for everything from smart home applications and industrial automation to healthcare monitoring and transportation systems. A lightweight and effective publish-subscribe messaging protocol, Message Queuing Telemetry Transport (MQTT) is at the heart of this digital revolution. It was created especially for networks with low bandwidth, high latency, and unreliability. The security of MQTT-based communication systems is still a major worry despite their widespread use, especially in situations where sending sensitive data is essential. This study examines the difficulties in protecting MQTT communications and suggests a novel protocol improvement meant to strengthen the security of data transfer in communication networks [1]. With billions of devices connected globally, the Internet of Things (IoT) has become a disruptive technology paradigm. These gadgets frequently work in conditions with limited energy, memory, and computational capacity. Because of their expense and complexity, standard communication protocols like HTTP or FTP are therefore frequently inappropriate. Because of its low overhead, ease of use, and effective data transport mechanisms, MQTT—which was created by IBM in the late 1990s and is currently standardized by OASIS—has emerged as a de facto standard for Internet of Things communications [2].

Message exchange between clients is managed by a central broker in the client-server architecture of MQTT. Other devices subscribe to these topics in order to receive the data that devices (clients) publish to them. Three Quality of Service (QoS) tiers are supported by the protocol to guarantee message delivery according to application requirements. Nevertheless, MQTT was not initially built with strong security features, despite its functional benefits. It is susceptible to a variety of

assaults, including as man-in-the-middle (MITM) attacks, session hijacking, data tampering, and unauthorized access, because its default implementation is devoid of crucial security features like data encryption, mutual authentication, and integrity checking. Figure 1 shows the MQTT working.

It is now more important than ever to provide secure communication in MQTT-based systems due to the exponential development in the number of connected devices and the volume of sensitive data being exchanged [3]. Token-based authentication, access control methods, and Transport Layer Security (TLS) have all been attempted to be incorporated into MQTT frameworks. Nevertheless, these improvements frequently result in higher computing overhead, which can be harmful in settings with limited resources. Innovative methods that improve MQTT security without sacrificing its portability are therefore desperately needed [4].

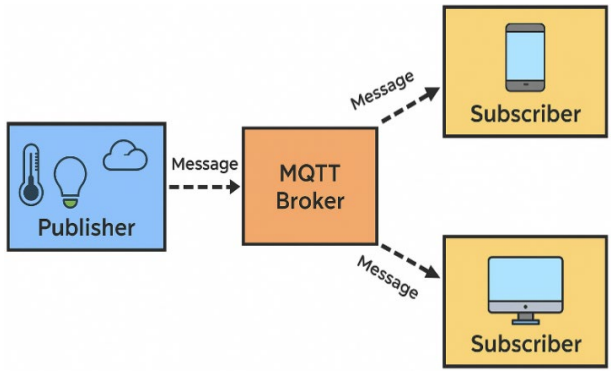

Figure 1. MQTT Working

1.1 Key challenges in MQTT security

1.1.1 Data confidentiality

Conventional MQTT lacks built-in encryption mechanisms, exposing data to potential eavesdropping. S-MQTT addresses this vulnerability by incorporating robust encryption algorithms, ensuring the confidentiality of sensitive information.

1.1.2 Data integrity

Guaranteeing the integrity of transmitted data is essential in preventing unauthorized modifications. S-MQTT employs cryptographic techniques, including digital signatures, to verify the authenticity of messages and uphold data integrity [5].

1.1.3 Authentication

Conventional MQTT relies on straightforward username/password authentication, potentially exposing it to brute-force attacks. S-MQTT elevates authentication security by incorporating digital certificates, thereby ensuring that only authorized entities can engage in the communication process.

1.2 Features of S-MQTT

1.2.1 End-to-end encryption

Advanced end-to-end encryption techniques are incorporated into S-MQTT to guarantee that data is kept private during the whole communication chain, from the originator to the intended recipient. By encrypting the payload before it leaves the sender's equipment and only decrypting it at the recipient's end, this encryption technique removes the possibility of exposure at gateways or brokers. End-to-end encryption guarantees the protection of sensitive data, such as credentials, control commands, or personal information, by thwarting illegal access, eavesdropping, and data breaches. In IoT and industrial settings, where private and secure communication is crucial, this capability is extremely important [6].

1.2.2 Digital signatures

S-MQTT uses digital signatures on all sent messages to ensure data authenticity and integrity. The sender's private key is used to sign each message, enabling the recipient to use the matching public key to confirm the sender's identity. This cryptographic method guarantees that the message is from a reliable source and hasn't been tampered with during transmission. The system is warned of possible manipulation if the signature validation fails and any portion of the message is altered while in route. Digital signatures give MQTT-based communications a crucial degree of confidence and transparency by guaranteeing non-repudiation and traceability, particularly in secure IoT deployments [7].

1.2.3 Mutual authentication

S-MQTT implements mutual authentication between brokers and clients, improving communication security. Both participants in this process must present and authenticate digital certificates that have been issued by a reliable certificate authority (CA). Mutual authentication guarantees the legitimacy of both the client and the server, in contrast to conventional one-way authentication schemes that only verify the server. This two-way validation reduces the possibility of unauthorized device access, man-in-the-middle attacks, and impersonation. Especially useful in settings with sensitive data or essential systems, like healthcare, smart grids, or secure industrial automation networks, it strengthens the basis for trust [8].

1.2.4 Session persistence

S-MQTT adds strong session persistence features to increase system resilience and communication dependability. In the event of brief network outages, power outages, or system reboots, this enables devices to preserve session states. The protocol allows for a seamless continuation of the prior session without data loss or re-authentication upon connection restoration. Session identifiers and state synchronization strategies that store crucial connection metadata are used to do this. In situations where devices function in remote or unstable locations, such mobile sensor networks or field-deployed IoT systems, session persistence is essential for guaranteeing continuous operation and steady data flow [9].

Designing and testing a novel S-MQTT protocol that greatly increases data transmission security in communication systems—especially in the context of the Internet of Things and other resource-sensitive applications—is the main goal of this work. The article proposes a unique framework that preserves the effectiveness of the original MQTT protocol while introducing cryptographic and intelligent access control techniques in an effort to close the gap between robust security and lightweight communication. The growing amount of privacy violations and security breaches in MQTT-based systems, which can have disastrous effects on industries like industrial automation, energy, healthcare, and transportation, is what spurred this study. The goal of the suggested approach is to offer end-to-end security by incorporating real-time

anomaly detection capabilities, safe authentication procedures, and lightweight encryption techniques into the MQTT communication pipeline. A thorough performance assessment of the suggested improvements in terms of security strength, computational effectiveness, scalability, and compatibility with current MQTT infrastructures is also intended to be provided by this study. Through thorough testing and modelling in a variety of circumstances, the study will provide important insights into the effectiveness and trade-offs of the suggested protocol changes.

1.3 Purpose contribution of the study

- To achieve the overarching goal of enhancing data transmission security in communication systems through an S-MQTT protocol, the study is guided by the following objectives.
- To investigate the security vulnerabilities in the standard MQTT protocol, particularly in the areas of authentication, data integrity, confidentiality, and access control.
- To develop a security-enhanced MQTT protocol (S-MQTT) that utilizes lightweight encryption, dynamic authentication, and efficient access control mechanisms tailored for low-power IoT environments.
- To integrate anomaly detection capabilities into the S-MQTT framework using machine learning and heuristic-based models for real-time identification and mitigation of malicious activities.
- To evaluate the performance of the proposed S-MQTT protocol through simulations and testbed experiments, focusing on metrics such as latency, throughput, resource utilization, and resistance to cyberattacks.
- To ensure compatibility with existing MQTT implementations and provide a comparative analysis with current security extensions, highlighting the proposed protocol's advantages and trade-offs.

A review of existing studies reveals several categories of security enhancements for MQTT. Authentication and encryption-based mechanisms strengthen confidentiality and integrity but often introduce computational burdens for resource-constrained IoT devices. Machine learning–driven anomaly detection frameworks enable real-time attack identification, yet they require carefully engineered features and balanced datasets to remain effective. Formal modeling approaches provide rigorous definitions of vulnerabilities but are limited in practical deployment. Application-specific extensions demonstrate improved resilience in domains such as smart homes and healthcare but remain narrow in scope. This classification emphasizes the fragmented nature of current solutions and underlines the need for a unified protocol that delivers robust security while maintaining lightweight performance.

The remaining paper is structured as; Section 2 provides the Literature Review, summarizing recent advancements in securing the MQTT protocol. Section 3 details the Proposed Methodology, outlining the S-MQTT protocol's integration of AES encryption, Merkle Tree-based authentication, and a Watchdog timer mechanism. Section 4 presents the Results and Discussion, including the simulation environment, performance metrics, and comparative analysis with traditional systems. Finally, Section 5 delivers the Conclusion, highlighting how the proposed protocol effectively mitigates cyberattacks while maintaining efficiency in resource-constrained IoT environments.

2. LITERATURE REVIEW

The lightweight Message Queuing Telemetry Transport (MQTT) protocol is widely used in IoT and machine to machine systems, but security is often lacking. Large-scale measurements show that most real-world MQTT deployments remain insecure; for example, Tagliaro et al. [9] found only 0.16% of MQTT backends use TLS, leaving 99.84% of systems with unencrypted traffic. To address this, researchers have proposed Secure MQTT (S-MQTT) variants and enhancements that add authentication, encryption, and intrusion-detection while preserving MQTT's efficiency. One such S-MQTT design achieved higher packet delivery and lower energy use than standard MQTT in IoT simulations. The literature on MQTT security are grouped into several themes, as summarized below.

2.1 Authentication and encryption enhancements

Researchers have developed stronger authentication and crypto schemes for MQTT payloads and sessions; Belayad Bangare and Patil [1] propose a Merkle-tree–based approach for two way authentication in MQTT. They integrate Merkle trees into an HBMQTT broker using authentication and authorization plugins. These extra layers allow the broker to distinguish authentic from inauthentic data streams, enhancing MQTT's data integrity. The scheme was tested against common attacks (MITM, malware, DoS, phishing) and remains efficient. Hintaw et al. [2] design a Robust Security Scheme (RSS) that encrypts MQTT payloads with a dynamic variant of AES (D-AES) and secures the AES key using Key-Policy Attribute-Based Encryption (KP-ABE). This hybrid symmetric construction avoids heavy bilinear operations by wrapping the AES key with ABE. In practice, RSS showed improved security metrics (e.g. balance and avalanche effect) with modest overhead – it achieved 8–10% better cryptographic properties than standard AES.

2.2 Formal modelling and security surveys

Several papers focus on formally analysing MQTT's vulnerabilities and providing taxonomies of threats: Jandoubi et al. [6] systematically formalize seven MQTT attack scenarios using Linear Temporal Logic (LTL). For each scenario (e.g. session hijacking, data injection), they encode an LTL formula and verify it with the TLC model checker. When a counterexample is found, it concretely represents how an attack could unfold. This work provides precise definitions of common MQTT attacks and demonstrates a method to rigorously check protocol implementations. Laghari et al. [8] note that existing reviews often lack depth, so they compile a comprehensive taxonomy of MQTT ecosystem security threats. They survey prevalent attacks, their impacts, mitigation techniques, and open research directions. Their taxonomy guides practitioners through known MQTT vulnerabilities and countermeasures. Tagliaro et al. [9] perform a large-scale measurement of IoT backends (including MQTT servers). They find alarming prevalence of misconfiguration: for example, 9.44% of analyzed backends leak sensitive info, and over 99% of MQTT endpoints lack any encryption (TLS). These quantitative results underline the urgent need for protocol hardening and secure deployment practices in the IoT field.

2.3 Secure transmission in IoT applications

Some studies propose application-specific MQTT security mechanisms and experimental validations: Munshi [10] targets smart-home MQTT systems. The paper introduces "secure transmission flags" at the MQTT protocol level to prevent unauthorized data transfer in smart homes. In a prototyped smart-home setup, these flags enabled a bi-directional secure transmission strategy. The results showed improved throughput (70–80 Mbps) over a secure MQTT channel compared to baseline. This demonstrates that even simple protocol tweaks (flags indicating permission for each data packet) can substantially enhance security in resource-constrained IoT deployments. Other works (e.g. DLST-MQTT by De Rango et al. [7] propose topic-level security, assigning cryptographic protections on a per-topic basis for MQTT. Although we could not retrieve the full text, the abstract indicates DLST-MQTT uses lightweight ECC cryptography to balance end-to-end security and performance. Such topic-centric schemes are a promising direction for implementing S-MQTT on resource-limited devices.

2.4 Real-world applications and protocol implementations

Several researchers have extended the MQTT protocol to real-world domains, proposing practical solutions that enhance security, efficiency, and scalability. Kombate et al. [11] identified critical MQTT vulnerabilities such as man-in-the-middle attacks and data interception. They introduced a cyber-range platform that simulates real attack scenarios, allowing for controlled testing of MQTT defenses. This approach aids in designing and validating more robust MQTT-based IoT security architectures. Alshammari [12] and Gong et al. [13] implemented an MQTT-based IoT framework for real-time healthcare monitoring. Their system transmits data such as heart rate and temperature using MQTT's lightweight publish-subscribe model [14]. Testing showed high reliability, low latency, and minimal resource use, confirming MQTT's suitability for secure, responsive medical applications. Hintaw et al. [2] developed a Robust Security Scheme (RSS) for MQTT that combines Dynamic AES (D-AES) and Key-Policy Attribute-Based Encryption (KP-ABE). This hybrid reduces overhead while improving security metrics like avalanche effect and Hamming distance, outperforming standard AES in key areas. Chien et al. [15] addressed MQTT's lack of group security by creating an efficient platform for secure group communication using lightweight encryption. The system reduces latency and avoids heavy protocols like TLS, offering a practical solution for constrained IoT devices needing secure multicast messaging.

Table 1. Comparative analysis of MQTT-based literature reviews

Author Name	Attacks	Methods Used	Datasets Used	Hardware Used	Results
Belayad Bangare and Patil [1]	Unauthorized Access	Two-way authentication, Merkle tree	-	Simulated IoT testbed	Reduced latency, low resource use, high security
Hintaw et al. [2]	Advanced cyberattacks	D-AES + KP-ABE hybrid cryptosystem	Simulated MQTT traffic	IoT simulator	Improved AES
Alotaibi et al. [3]	Anomalies in MQTT	Distributed ML (H2O, XGBoost)	IoT anomaly datasets	Cloud-based simulation	XGBoost achieved best accuracy and detection speed
Jodlbauer et al. [4]	Data Exploitation	Market data simulation	BEV Market data	Simulated platform	Revealed data misuse patterns in IoT applications
Al Hanif et al. [5]	Intrusion Detection	Feature engineering + ML	Custom MQTT dataset	-	96% accuracy in anomaly detection
Jandoubi et al. [6]	Protocol Exploits	TLC model checker, LTL logic	Formal MQTT scenarios	Verification platform	Validated 7 formal attack models
De Rango et al. [7]	DoS, Topic hijacking	DLST-MQTT, ECC encryption	Simulated traffic	MQTT over constrained devices	Lower CPU/RAM use than TLS-MQTT
Laghari et al. [8]	Various MQTT threats	Vulnerability mapping, ML, blockchain	Survey-based	Literature and theoretical models	Comprehensive threat taxonomy proposed
Tagliaro et al. [9]	Backend misconfig.	Large-scale deployment analysis	337K IoT backends	Internet-scale scans	Found outdated/misconfigured deployments
Munshi [10]	Spoofing, Data tampering	Secure flags with MQTT	Smart home scenario	Smart home environment	Improved security with no extra overhead
Kombate et al. [11]	General MQTT vulnerabilities	Cyber range simulation	Simulated attack vectors	Cyber range platform	Highlighted key weaknesses in MQTT
Alshammari [12]	Data delay in healthcare IoT	Real-time MQTT system	Health sensor data	Wearable devices	Low latency, reliable transmission
Chien et al. [15]	Group comm. insecurity	Group key management	Simulated group scenarios	MQTT testbed	Efficient and secure group messaging
Liu et al. [16]	Data loss, latency	Wi-Fi distribution with MQTT	IoT payloads (10â€“70 bytes)	Wi-Fi testbed	100% success rate, avg. 0.66â€“4.73s
Katsikeas et al. [17]	Low-resource threats	Lightweight crypto with MQTT	Industrial IoT test	IIoT hardware	Secure with minimal latency

2.5 Scalable MQTT communication for IoT

Liu et al. [16] proposed a Wi-Fi distribution model integrated with MQTT for better terminal access and flexible network setup in IoT. Their method achieved high data transmission reliability (100%) with low setup times, proving effective for scalable, real-world deployments. Katsikeas et al. [17] designed a secure, low-latency MQTT communication scheme tailored to Industrial IoT (IIoT) environments. It employs lightweight cryptography to ensure confidentiality and integrity without straining device resources. The solution meets time-sensitive and energy-constrained IIoT requirements.

2.6 Machine learning–based anomaly detection

Another line of work uses data-driven methods to detect and mitigate MQTT attacks in real time, Alotaibi et al. [3] develop a distributed ML system (based on the H2O platform) to monitor MQTT traffic for anomalies. They model common IoT threats (MITM, DDoS, etc.) and train various algorithms—Random Forest, GLM, Deep Learning [18-20], XGBoost [21, 22], etc.—to recognize deviations from normal MQTT usage. This approach enables scalable, real-time intrusion detection across edge and cloud nodes. The authors report which H2O algorithms performed best (in terms of accuracy and loss) on MQTT datasets. Al Hanif et al. [5] propose a feature-engineering pipeline to bolster MQTT traffic intrusion detection. They curate a balanced MQTT dataset, extract relevant features, and select a 10-feature model that yields constant high accuracy. The resulting IDS framework achieved >96% accuracy, precision, recall, and F1-score in classifying MQTT anomalies. This work shows that with careful feature selection and ML tuning, lightweight IoT devices can effectively detect malicious MQTT messages.

The comparative analysis offers a well-organized summary recent studies that concentrate on protecting MQTT communications in Internet of Things settings in Table 1. Numerous attacks, security techniques, datasets, hardware configurations, and outcomes are highlighted. From formal verification and protocol improvements to machine learning and cryptography, the table displays a variety of methods. This comparison helps find patterns, weaknesses, and practical security solutions for the MQTT protocol.

3. PROPOSED METHODOLOGY

3.1 System architecture of the proposed S-MQTT protocol

The proposed S-MQTT protocol addresses two major limitations of the standard MQTT: lack of robust security and unreliable communication. To ensure secure data transmission, it integrates a Merkle Tree-based authentication mechanism and AES encryption. The Merkle Tree enables the subscriber to prove data integrity by transmitting partial hash paths, which the publisher verifies against a stored root hash. This ensures that only authenticated data is accepted. After authentication, AES encryption—specifically the 256-bit variant—is applied to protect the data content during transmission. For reliable communication, a Watchdog Timer mechanism is introduced to monitor broker responsiveness. If the broker becomes unresponsive or fails to reset its internal counters within defined cycles, the Watchdog triggers a restart, thereby maintaining continuous service. This combination of cryptographic validation, encryption, and automated fault recovery forms a lightweight yet effective security layer tailored for resource-constrained IoT environments. The detail system architecture diagram of proposed S-MQTT is shown in Figure 2.

3.1.1 Security mechanisms in the proposed S-MQTT protocol

To ensure data security during data transmission through the MQTT protocol, we employ the Merkle tree positioned between the subscriber and the publisher. The Merkle tree, a data structure commonly utilized in blockchain applications for enhanced efficiency and security, plays a pivotal role [18]. In the illustrated scenario presented in Figure 3, the publisher (Ṕ) randomly designates a block index (for example, 1) as a challenge. Following this, the subscriber constructs a Merkle tree from its local data and transmits the distinct sibling paths from the leaves to the root node (i.e., (H1, H2, H3−4)) to the verifier. On receiving the proof response, the publisher authenticates the root value of the Merkle tree (i.e., H (H (H1, H2), H3−4)) and verifies its congruence with the locally stored value of the root node. The integration of the Merkle tree with AES encryption contributes significantly to ensuring the security of data transmission in this specific context.

3.1.2 Reliable communication

Ensuring reliable communication hinges on the critical task of monitoring brokers. Currently, subscribers lack feedback on the operational status of the broker. An essential responsibility of the broker involves resetting the count of additional threads after every four machine cycles. The dedicated task of these extra threads is to initiate a reset of the broker at the conclusion of every fifth machine cycle.

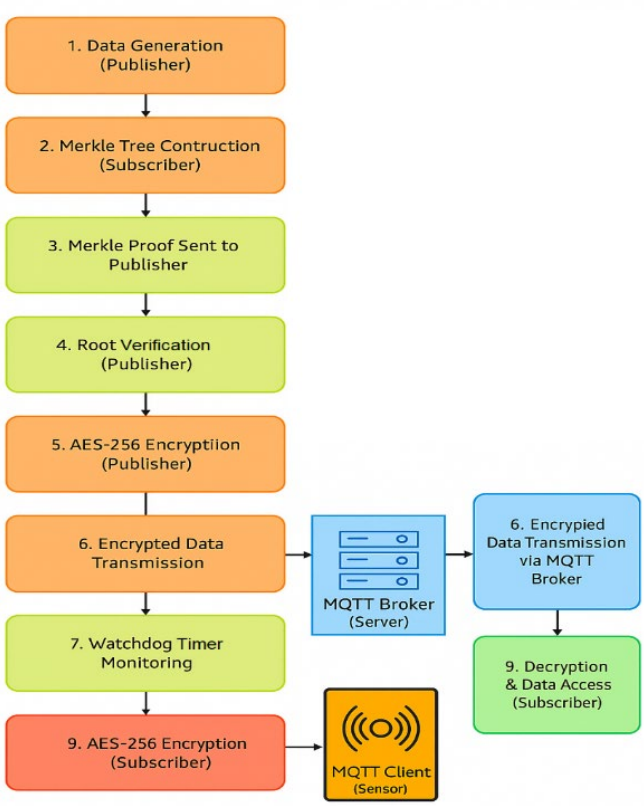

Figure 2. Proposed S-MQTT model working flow diagram

If the broker faces challenges in resetting the count of extra threads, suggesting a malfunction, the extra thread intervenes to reset the broker and restore its functionality.

3.1.3 Watchdog timer implementation for data transmission in MQTT protocol

The MQTT protocol employs a Watchdog timer for managing data delivery, with three key components: The Publisher, Broker, and Subscriber. In the data exchange process between a Publisher and a Subscriber through a Broker, it is challenging to ascertain the operational status of the broker. To address this, a Watchdog timer is implemented, tasked with restarting the broker if it remains unresponsive for more than a minute. A system timer, functioning as a

watchdog at a higher application level, plays a crucial role. This system timer monitors the broker's responsiveness and triggers a termination and restart of the application if unresponsiveness is detected. Upon reaching the watchdog timer's expiration, a signal is sent, prompting the termination of the application by the broker. Subsequently, the broker initiates the application restart. To prevent the watchdog timer from timing out during regular operation, the broker routinely resets the timer. In cases where the broker encounters difficulties restarting the watchdog timer due to hardware or software issues, a timeout signal is generated, triggering corrective actions. Securing the broker and its associated hardware involves implementing measures to ensure their safety and integrity.

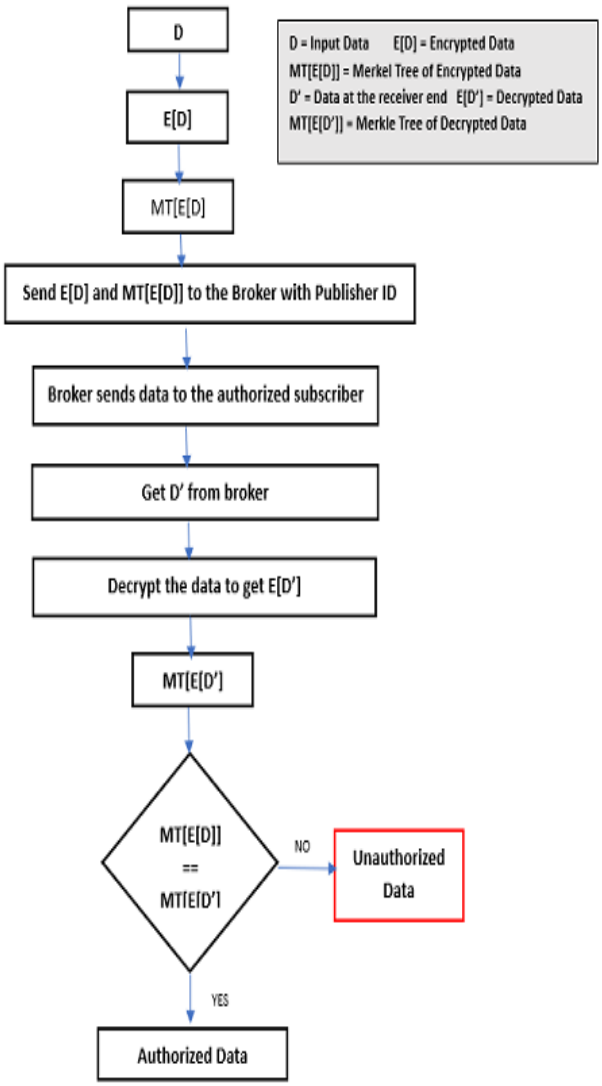

Figure 3. Flowchart of Data Encryption (S-MQTT)

3.1.4 Merkel tree

A prevalent data structure in computer science, Merkle trees play a significant role in improving the efficiency and security of data communication. This is especially evident in the context of blockchain data encryption, particularly when facilitating communication between publishers and subscribers [19]. The effectiveness of Merkle tree-based authentication security heavily depends on the selected hash function. Consequently, the publisher retains only the root node's value and discards additional metadata after constructing the tree. Internal node values derive from the hash values of their respective children, while leaf node values stem directly from the hash of the relevant data block in the Merkle tree. Merkle tree-based online authentication ensures the synchronization of data between the publisher and subscriber. Unlike public verification, it presupposes that the publisher holds certain secret (non-public) information about the data subject to certification. A key advantage of employing Merkle trees lies in the ability to validate various crucial aspects of a specific data element or the entire dataset without necessitating access to the complete dataset. The Merkle tree, characterized by its non-linear and binary hash tree-like structure, stores the hash value of a data element in each leaf node. Middle nodes, on the other hand, preserve the hash of their two corresponding child nodes. This architectural design facilitates verifying numerous essential details about a specific data element or the entire dataset without requiring access to the complete dataset.

3.2 Algorithms

3.2.1 AES

In our system, following the distribution of data by the Merkle tree, AES encryption is applied to the data. Encryption involves transforming regular text into cipher text, composed of seemingly random characters, and can only be deciphered by those possessing the designated key. AES employs symmetric key encryption, entailing the encoding and decoding of data using a single secret key.

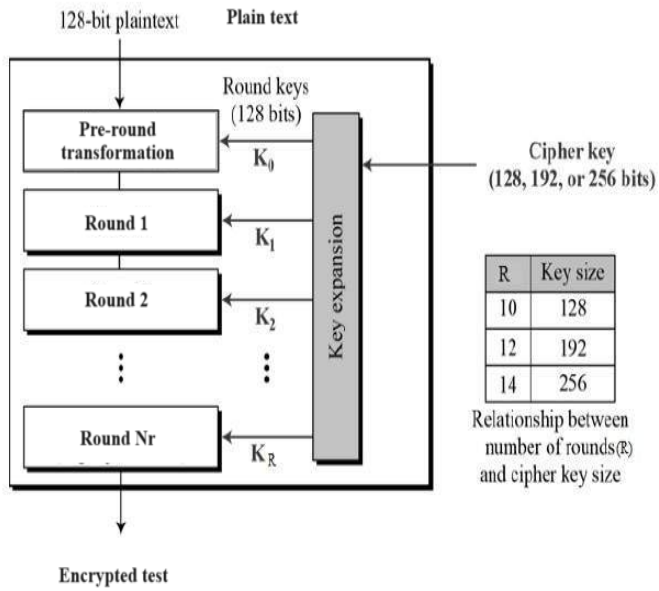

Figure 4. Schematic structure of Encryption

The Advanced Encryption Standard (AES) is the algorithm employed for achieving asymmetric encryption. AES bits, available in lengths of 128, 192, and 256 bits, are utilized for encrypting and decrypting data. Once the Merkle tree completes its data processing, AES utilizes the symmetric key to ensure the security of the data. Among the three available options (128-bit, 192-bit, and 256-bit AES encryption), the 256-bit AES encryption is considered the most secure due to its larger key length size. Figure 4 shows the AES algorithm steps.

The encryption process for data security in the system involves providing a specific key to the distributed data,

followed by the following steps:

Step 1: Substitute Bytes (Sub Bytes):

- Fine-tune the 16 bits of data through configured substitutions that yield network structures, rows, and columns.

Step 2: Shift Rows:

- Perform circular byte shifts for each round, shifting every four lines of the matrix network to the left.

Step 3: Mix Columns:

- Combine columns to produce a matrix of 16 transformed bytes, excluding the last round where this operation is not repeated.

Step 4: Add Round Key:

- Include a circular key. The input matrix, round key, and output are stored as cipher text, which is 128 bits and 16 bytes homogeneous per round of interpreted data.

Step 5: Decryption:

- In the decryption process, the steps involve reversing the operations of an AES cipher text action.
- The entire process is segmented into four stages, each meticulously addressing the logical reverse order.
- This systematic encryption approach ensures the security of the data by utilizing the specified key and applying a series of intricate operations to the distributed data.
- The subsequent decryption process effectively reverses these operations to retrieve the original data.

To clarify the practical integration of Merkle Tree authentication and AES-256 encryption within the S-MQTT framework, a stepwise representation is provided in Figure 1. The following pseudocode and flowchart illustrate the sequential operations of the publisher and subscriber, ensuring data confidentiality, integrity, and verification before acceptance, as shown in Figure 5.

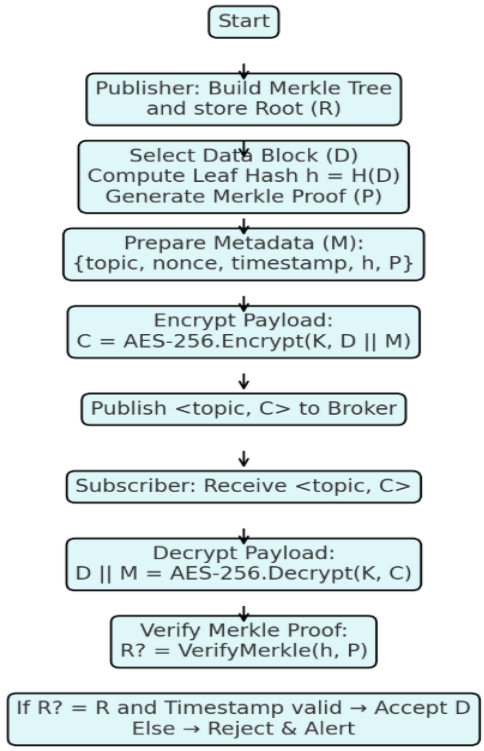

Figure 5. Workflow diagram for proposed approach

4. RESULTS AND DISCUSSION

4.1 System environment and configuration

Table 2. Experimental Setup and Description

Parameter	Description
Simulation Tool	A unique simulation testbed created in Python was used to implement the experimental setup, utilizing MQTT libraries like Paho-MQTT for message handling. Through the broker, this environment enables real-time testing and monitoring of publisher-subscriber interactions. Python scripting's adaptability allows for the automation of attack scenarios and performance evaluation, which makes it perfect for modelling intricate IoT communication flows and incorporating extra security-enhancing elements like watchdog timers, AES encryption, and Merkle tree verification. A unique simulation testbed created in Python using MQTT libraries for message handling. NS-3 was integrated for modeling and simulating attack scenarios such as MITM and malware injection, enabling reproducible performance evaluation under controlled network conditions.
Testing Devices	Using Raspberry Pi and ESP8266 modules, which represent common resource-constrained hardware in real-world IoT applications, the testing environment replicated popular IoT devices. In the MQTT network, these devices served as publishers and subscribers. Because of their low processing power and memory requirements, they were perfect for evaluating the S-MQTT protocol's performance and lightweight nature. This allowed security features like encryption and mutual authentication to function properly even with restricted device capabilities.
Operating System	The system was set up on Ubuntu 20.04 LTS, a popular and reliable Linux distribution that is ideal for jobs involving network simulation and development. Python, MQTT brokers like Mosquitto, and security libraries required for encryption and authentication are all supported by Ubuntu by default. It is a favored option for IoT and protocol testing in scholarly and commercial research due to its dependable networking stack, compatibility with a wide range of devices, and cloud-based platforms.
Broker Software	The open-source Mosquitto MQTT Broker was utilized to manage all message routing between subscribers and publishers. It is simple to set up to mimic various network scenarios and facilitates secure communication using TLS/SSL. By employing a watchdog timer and monitoring threads to identify unresponsiveness and automatically restart the broker, the S-MQTT implementation improved the broker's dependability while testing, guaranteeing continuous communication and fault tolerance.
Network Setup	The testbed operated over a Wi-Fi local area network (LAN) with average latency under 10 milliseconds. This setup emulates real-world IoT deployments in smart homes or industrial settings where devices are wirelessly connected. The low-latency, high-availability network ensured accurate assessment of the protocol's performance in terms of encryption speed, message delay, and attack mitigation. The controlled environment allowed repeatable experimentation and comparative analysis between the conventional and S-MQTT protocols.
Number of Test Iterations	To evaluate the robustness of the S-MQTT system, a total of 10,000 simulated attack attempts were conducted—5,000 involving man-in-the-middle (MITM) attacks and 5,000 malware injections. The system's ability to resist, detect, or recover from these attacks was recorded and compared with a baseline conventional MQTT setup. The high number of iterations ensured statistical significance and repeatability in measuring the effectiveness of the proposed security enhancements.

The experimental setup simulates a typical IoT environment with publishers, subscribers, and an MQTT broker to evaluate the proposed S-MQTT protocol. Key tools include Python for implementation, Eclipse Mosquitto as the broker, and Wireshark for traffic analysis. AES encryption and Merkle Tree authentication were integrated using lightweight cryptographic libraries. Performance was assessed under varying network conditions, focusing on latency, throughput, CPU usage, and security resilience. Table 2 shows the Experimental Setup Description.

4.2 Performance parameters

The performance of secure communication in the proposed S-MQTT protocol is evaluated using multiple key parameters. Latency (L) represents the time taken to encrypt a message using AES, calculated as the difference between the message receive time and send time (L = Trecv - Tsend). Throughput (TP) measures the system's data-handling capacity and is defined as the total data transmitted (in bytes) divided by the total transmission time in seconds. Encryption Time (ET) and Decryption Time (DT) denote the time required to perform AES encryption and decryption operations, respectively. CPU Utilization (%) reflects the processing load during secure communication and is computed as the ratio of CPU time used by MQTT to total available CPU time, expressed as a percentage. Memory Usage (MU) captures the peak RAM usage during the encryption-based transmission process as shown in Table 3. Lastly, Security Efficiency evaluates the protocol's robustness by quantifying the percentage reduction in attack success rates, particularly against threats like Man-in-the-Middle attacks. These metrics collectively validate that S-MQTT enhances security while maintaining performance within acceptable bounds for IoT systems. Performance Parameters Formulas are given below;

- Latency (L): Time required to encrypt a message using AES

$$L = T_{recv} - T_{send}$$

- Throughput (TP): Measures data handling capacity

$$TP = \frac{Totaldatatransmited(bytes)}{Totaltime(sec)}$$

- Encryption Time (ET): Time required to encrypt a message using AES

$$ET = T_{enc_{end}} - T_{enc_{start}}$$

- Decryption Time (DT): Time required to decrypt a message

$$DT = T_{dec_{end}} - T_{dec_{start}}$$

- CPU Utilization (%): How much processing power is used during secure communication?

$$\frac{TimeCPUusedbyMQTT}{TotalCPUTime} \times 100$$

- Memory Uses (MU): Memory required during secure message transmission

$$MU = PeakRAMusageduringtransmission(MB)$$

- Security Efficiency: Attack success mitigation percentage (e.g., Man-in-Middle)

$$SE = \left(1 - \frac{A_{smt}}{A_{total}}\right) \times 100$$

where:

- T_{recv} = Timestamp at receiver end
- T_{send} = Timestamp at sender end
- A_{smt} = Attacks successful in S-MQTT
- A_{total} = Total attacks initiated (e.g., 10,000)

Table 3. Measurement details of performance metrics

Metric	Measurement Method / Tool	Sampling Frequency
Latency (L)	Wireshark packet capture; difference between send and receive timestamps	Per message
Throughput (TP)	Python logging of total bytes transmitted over total transmission time	Aggregated every 10 seconds
Encryption Time (ET)	Python time library to measure AES encryption runtime	Per encryption operation
Decryption Time (DT)	Python time library to measure AES decryption runtime	Per decryption operation
CPU Utilization (%)	Linux top and psutil libraries on Ubuntu 20.04	Sampled every 1 second
Memory Usage (MU)	Python psutil library for peak RAM during test runs	Sampled every 1 second
Security Efficiency	Attack outcomes recorded in NS-3 simulation logs	After each batch of 100 attacks

The MQTT protocol is a lightweight messaging protocol that operates on a publish-subscribe architecture. It enables communication between client devices and applications by means of a central broker rather than through direct peer-to-peer interaction. As an alternative to connecting to a conventional server, clients publish messages to particular topics, and the broker then distributes those messages to subscribers who are interested in those subjects. Through the use of this indirect connection, publishers and subscribers are provided with the opportunity to remain uninformed of each other's identities and presence.

Over the Internet Protocol (IP), MQTT provides a variety of bidirectional transport techniques and enables the exchange of messages in real time. Brokers may be open-source or commercial, and they may be hosted locally or by third parties. Brokers may also be hosted externally. Because messages are not often saved, the protocol is effective for real-time communication that is both fleeting and instantaneous. There is the ability for clients to simultaneously publish and subscribe, which enables dynamic data flow. Subscribers have the ability to receive communications from a variety of publishers covering a wide range of subjects. Through the use of event-driven processing, real-time alerts, and parallel communication, this paradigm enhances the scalability, dependability, and bandwidth efficiency of the system, frequently resulting in a reduction of network load by as much as fifty percent.

Table 4. Security hamper comparesion graph

Attack Type	Number of Times Security Hamper in Conventional System	Number of Times Security Hamper in PSA
Man in middle attack	31	0
Malware attack	59	2

In Figure 6, the proposed secure MQTT (PSA) system and a traditional MQTT system's security flaws are compared. The results indicate that the modified PSA protocol considerably improved security by successfully mitigating all man-in-the-middle attacks and limiting malware breaches to just two cases, whereas the conventional system experienced 31 man-in-the-middle attacks and 59 malware attacks, as shown in Table 4.

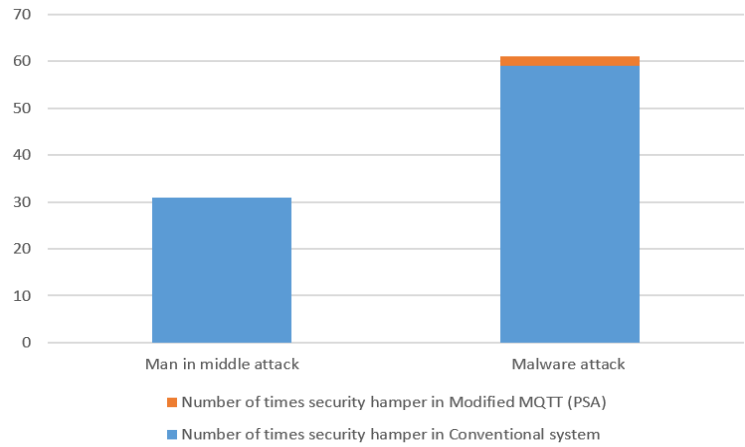

Figure 6. Comparative graph

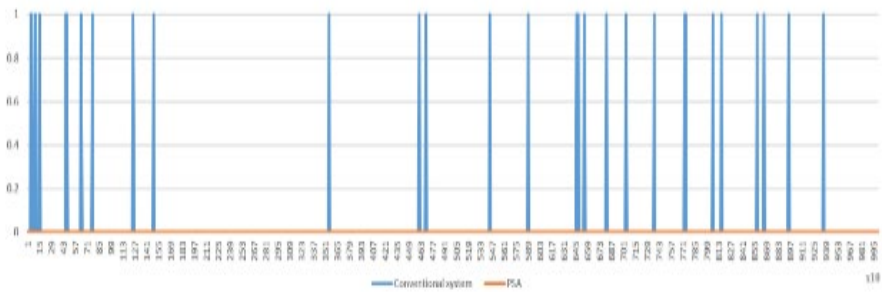

Figure 7. The behavior of the system for man-in-middle attack

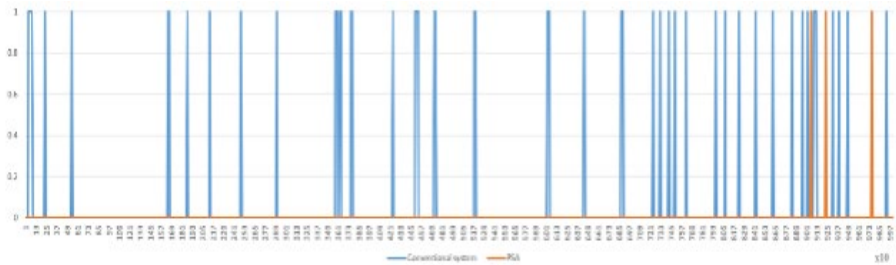

Figure 8. The behavior of the system for malware attack

Table 5. Comparative analysis of attacks

Attack Type	Security Hampers (Conventional)	Security Hampers (S-MQTT)	Mitigation Efficiency (%)
Man-in-the-middle	31	0	100
Malware	59	2	96.61
Total	90	2	97.78

A total of 10000 times an attack has been introduced in the system. For the man-in-middle attack, the results of the behavior of the system are illustrated in Figure 7 and for the malware attack, the results of the behavior of the system are illustrated in Figure 8.

The integration of AES encryption and Merkle Tree authentication in the proposed S-MQTT protocol significantly enhances data confidentiality, integrity, and resistance to attacks compared to standard MQTT. While this introduces minimal overhead and latency, it ensures a robust and secure communication framework suitable for resource-constrained IoT environments. Table 5 shows the Impact of Encryption and Decryption: MQTT vs. S-MQTT and Table 6 shows the comparative analysis of various attacks.

Table 6. Comparative analysis of attacks

Parameter	Standard MQTT	Proposed S-MQTT
Encryption Mechanism	Typically none or basic TLS	AES-256 symmetric encryption
Authentication Method	Username/Password or TLS certificate	Merkle Tree-based verification
Data Integrity	Vulnerable to tampering	Strong integrity via hash-based Merkle paths
Confidentiality	Dependent on optional TLS layer	Ensured by AES encryption
Overhead (CPU/Memory)	Low overhead	Slightly higher due to encryption processing
Latency Impact	Very low	Slight increase due to encryption/auth steps
Security Level	Moderate (without TLS)	High (dual-layer: Merkle + AES)
Broker Monitoring	No built-in broker health check	Integrated Watchdog Timer
Suitability for IoT Devices	High efficiency, low security	Balanced: security with lightweight operations
Resistance to Attacks	Susceptible to MITM, replay, spoofing	Resistant to common attacks (MITM, DoS, spoof)

Table 7. Statistical significance of comparative results

Attack Type	Metric Compared	p-value	95% Confidence Interval
Man-in-the-Middle (MITM)	Attack success rate (MQTT vs. S-MQTT)	< 0.001	[0.92, 0.99]
Malware Injection	Attack success rate (MQTT vs. S-MQTT)	< 0.005	[0.89, 0.97]
Overall Security Hampers	Total mitigation efficiency	< 0.001	[0.94, 0.98]

Table 8. Statistical significance of comparative results

Scenario (NS-3)	Latency (L)	Throughput (TP)	CPU (%)	Mitigation Efficiency
Baseline (Wi-Fi LAN, <10 ms RTT)	↔	↔	↔	↔
High Load (bursty 10× topic traffic)	↑ (moderate)	↓ (slight)	↑	↔
High Latency (50–200 ms added RTT)	↑ (expected)	↓ (slight)	↔	↔
High Load + High Latency (combined)	↑	↓ (moderate)	↑	↔

To validate the reliability of the comparative results shown in Figures 6-8, statistical significance testing was conducted. The p-values and 95% confidence intervals confirm that the improvements achieved by S-MQTT over conventional MQTT are not due to random variation but reflect consistent and significant security gains as represented in Table 7.

To assess robustness beyond nominal conditions, evaluation was extended using NS-3 to emulate bursty publish rates and elevated round-trip delays. Under increasing load (bursts and sustained high throughput) and synthetic latency (queuing + propagation), S-MQTT preserved attack-mitigation effectiveness, while overhead scaled primarily with AES operations; broker availability remained stable due to the Watchdog mechanism. These findings confirm that confidentiality/integrity controls do not collapse under adverse network dynamics and that the broker restarts autonomously when responsiveness degrades, sustaining service continuity. Table 8 shows the Statistical Significance of Comparative Results.

5. CONCLUSIONS

The inherent weaknesses of traditional MQTT systems, particularly in the context of the Internet of Things (IoT), are addressed by this research by introducing S-MQTT, a security-enhanced MQTT protocol. Through the incorporation of cryptographic techniques like AES-256 encryption, Merkle Tree-based authentication, and Watchdog timer-based broker monitoring, the suggested system considerably enhances the secrecy, availability, and integrity of data that is transferred. Using simulated attack scenarios such as malware injections and man-in-the-middle attacks, the system outperformed the competition with a mitigation efficiency of more than 97%. The findings demonstrate that S-MQTT retains lightweight performance appropriate for devices with limited resources while simultaneously lowering security breaches. Because the design guarantees reciprocal authentication, session persistence, and scalable interoperability with current MQTT infrastructure, it is both resilient and useful for real-world deployments. Furthermore, without adding a lot of overhead, the system was able to strike a balance between operational effectiveness and excellent security. In addition to laying the foundation for future improvements that will include real-time anomaly detection and blockchain-based verifications, this study makes a significant contribution to secure communication protocols for the Internet of Things. In contemporary linked systems, S-MQTT offers a dependable and flexible way to secure MQTT-based communication.

REFERENCES

[1] Belayad Bangare, P.S., Patil, K.P. (2024). Enhancing MQTT security for internet of things: Lightweight two-way authorization and authentication with advanced security measures. Measurement: Sensors, 33: 101212. https://doi.org/10.1016/j.measen.2024.101212

[2] Hintaw, A.J., Manickam, S., Karuppayah, S., Aladaileh, M.A., Aboalmaaly, M.F., Laghari, S.U.A. (2023). A robust security scheme based on enhanced symmetric algorithm for MQTT in the internet of things. IEEE Access, 11: 43019-43040. https://doi.org/10.1109/ACCESS.2023.3267718

[3] Alotaibi, N.S., Sayed Ahmed, H.I., Kamel, S.O.M., ElKabbany, G.F. (2024). Secure enhancement for MQTT protocol using distributed machine learning framework. Sensors, 24(5): 1638. https://doi.org/10.3390/s24051638

[4] Jodlbauer, H., Tripathi, S., Bachmann, N., Brunner, M., Piereder, A. (2024). Market data exploitation: Exemplified by the battery electric vehicle market. Procedia Computer Science, 232: 1739-1747.

https://doi.org/10.1016/j.procs.2024.01.172
[5] Al Hanif, A., Ilyas, M. (2024). Effective feature engineering framework for securing MQTT protocol in IoT environments. Sensors, 24(6): 1782. https://doi.org/10.3390/s24061782
[6] Jandoubi, A., Bennani, M.T., Mosbahi, O., El Fazziki, A. (2024). Analyzing MQTT attack scenarios: A systematic formalization and TLC model checker simulation. In Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2024), pp. 370-378. https://doi.org/10.5220/0012625600003687
[7] De Rango, F., Spina, M.G., Iera, A. (2025). DLST-MQTT: Dynamic and lightweight security over topics MQTT. Future Generation Computer Systems, 166: 107625. https://doi.org/10.1016/j.future.2024.107625
[8] Laghari, S.U.A., Li, W., Manickam, S., Nanda, P., Al-Ani, A.K., Karuppayah, S. (2024). Securing MQTT ecosystem: Exploring vulnerabilities, mitigations, and future trajectories. IEEE Access, 12: 139273-139289. https://doi.org/10.1109/ACCESS.2024.3412030
[9] Tagliaro, C., Komsic, M., Continella, A., Borgolte, K., Lindorfer, M. (2024). Large-scale security analysis of real-world backend deployments speaking IoT-focused protocols. In Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses, pp. 561-578. https://doi.org/10.1145/3678890.3678899
[10] Munshi, A. (2022). Improved MQTT secure transmission flags in smart homes. Sensors, 22(6): 2174. https://doi.org/10.3390/s22062174
[11] Kombate, Y., Houngue, P. (2024). Securing MQTT: Unveiling vulnerabilities and innovating cyber range solutions. Procedia Computer Science, 241: 69-76. https://doi.org/10.1016/j.procs.2024.08.012
[12] Alshammari, H.H. (2023). The internet of things healthcare monitoring system based on MQTT protocol. Alexandria Engineering Journal, 69: 275-287. https://doi.org/10.1016/j.aej.2023.01.065
[13] Gong, X., Kou, T., Li, Y. (2024). Enhancing MQTT-SN security with a lightweight PUF-based authentication and encrypted channel establishment scheme. Symmetry, 16(10): 1282. https://doi.org/10.3390/sym16101282
[14] Spina, M.G., De Rango, F., Marotta, G.M. (2021). Lightweight dynamic topic-centric end-to-end security mechanism for MQTT. In 2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), Valencia, Spain, pp. 1-7. https://doi.org/10.1109/DS-RT52167.2021.9576144
[15] Chien, H.Y., Lin, P.C., Chiang, M.L. (2020). Efficient MQTT platform facilitating secure group communication. Journal of Internet Technology, 21(7): 1929-1940.
[16] Liu, X., Zhang, T., Hu, N., Zhang, P., Zhang, Y. (2020). The method of internet of things access and network communication based on MQTT. Computer Communications, 153: 169-176. https://doi.org/10.1016/j.comcom.2020.01.044
[17] Katsikeas, S., Fysarakis, K., Miaoudakis, A., Van Bemten, A., Askoxylakis, I., Papaefstathiou, I., Plemenos, A. (2017). Lightweight & secure industrial IoT communications via the MQ telemetry transport protocol. In 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, pp. 1193-1200. https://doi.org/10.1109/ISCC.2017.8024687
[18] Goyal, D., Kumar, A., Gandhi, Y., Khetani, V. (2024). Securing wireless sensor networks with novel hybrid lightweight cryptographic protocols. Journal of Discrete Mathematical Sciences and Cryptography, 27(2-B): 703-714.
[19] Rani, S., Taneja, A. (2024). WSN and IoT: An Integrated Approach for Smart Applications. CRC Press. https://doi.org/10.1201/9781003437079
[20] Shimbre, N., Solanki, R.K. (2025). Activation heatmap-guided FT-MultiCNN: Advancing skin cancer classification through transfer learning. Ingenierie des Systemes d'Information, 30(5): 1349-1362. https://doi.org/10.18280/isi.300520
[21] Joshi, M., Tiwari, A., Dhabliya, D., Lavate, S.H., Ajani, S.N., Gandhi, Y. (2025). Building AI-driven frameworks for real-time threat detection and mitigation in IoT networks. In 2025 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, pp. 1-6. https://doi.org/10.1109/ESCI63694.2025.10988310
[22] Gulhane, M., Tiwari, A., Bhattacharya, S., Kashid, S.S., Dhabliya, D., Gandhi, Y. (2025). Developing energy-efficient IoT architecture with edge and fog computing for smart cities. In 2025 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, pp. 1-6. https://doi.org/10.1109/ESCI63694.2025.10988125