International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

A Comprehensive Survey of Authentication Mechanisms in MQTT Broker Implementations

A. Renuka Devi[1], M. Chandra Mohan[2]

Submitted:12/03/2024 **Revised**: 25/04/2024 **Accepted**: 01/05/2024

Abstract: The MQTT (Message Queuing Telemetry Transport) protocol has emerged as a prominent communication means in the world and the messaging applications. As MQTT deployments continue to grow in scale and complexity, ensuring robust authentication mechanisms becomes paramount to safeguarding the confidentiality, integrity and availability of MQTT communication. This survey paper provides a comprehensive analysis of authentication mechanisms in MQTT protocol, encompassing a wide range of approaches, from basic username/password authentication to advanced techniques such as client certificates, OAuth 2.0 integration, and token-based authentication. The paper begins by examining the foundational concepts of MQTT protocol and the importance of authentication in securing MQTT deployments. It then proceeds to systematically explore various authentication mechanisms available in MQTT, detailing their strengths, weaknesses. Furthermore, the survey investigates recent advancements in MQTT authentication, and the authentication schemes in different MQTT broker implementations. Additionally, the paper discusses challenges and open research areas in MQTT authentication, offering insights into potential future directions for research and development. By synthesizing existing literature and providing critical insights, this survey paper becomes an invaluable resource for researchers, practitioners, and IoT stakeholders seeking to understand, evaluate, comprehend and implement authentication mechanisms in MQTT protocol effectively.

***Keywords**: Internet of Things, MQTT, Authentication, One time password*

1. INTRODUCTION

The Internet of Things (IoT) refers to a network of connected devices with built-in sensors, software, and other technologies that allow them to collect and communicate data over the internet. These devices can range from everyday objects such as home appliances, wearable devices, and industrial machinery to complex systems like smart cities and autonomous vehicles. MQTT is a lightweight messaging protocol that is frequently used in Internet of Things (IoT) applications. It follows a publish/subscribe messaging pattern, facilitating efficient communication between devices and applications.

MQTT is well-suited for resource-constrained devices, such as sensors and actuators, and operates efficiently over low-bandwidth or unstable networks. While MQTT itself does not mandate specific security mechanisms, it can be used over secure transport protocols like TLS/SSL for encryption and authentication. Authentication and authorization mechanisms can be implemented at the application level to guarantee secure communication between clients and brokers.

MQTT has gained widespread implementation in various IoT applications, including home automation, industrial automation, smart cities, smart homes, healthcare monitoring, and smart agriculture. The Organization for the Advancement of Structured Information Standards (OASIS) defined it as an open protocol to encourage compatibility and interoperability across various implementations.

MQTT implements a publish/subscribe messaging pattern, where clients (devices or applications) send messages to topics called as publish and the client acting as subscriber receives messages from specific topics as shown in the fig 1.

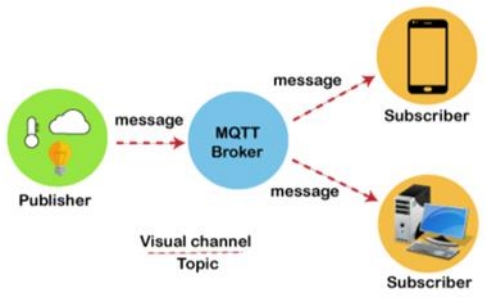

Fig. 1. Publish/Subscribe Mechanism

MQTT has the following features:

Quality of Service (QoS) Levels: MQTT supports different levels of message delivery assurance, known as QoS levels, allowing users to choose the appropriate level of reliability for their applications. QoS 0 (At most once): No confirmation or acknowledgment is provided; messages are delivered at most once. QoS 1 (At least once): Messages are sent out at least once, which

[1] *Research Scholar, JNTU, Hyderabad, renuka.adibhatla@gmail.com*
[2] *Professor of CSE, JNTU, Hyderabad, c_miryala@jntuh.ac.in*

guarantees message delivery but may cause duplication. QoS 2 (Exactly once): Despite higher overhead, messages are only transmitted exactly once, ensuring message delivery and eliminating duplicates.

Connection-Oriented Protocol: MQTT operates on top of TCP/IP or other transport protocols, establishing a connection between clients (publishers and subscribers) and a central broker. Clients connect to the broker and can maintain persistent sessions, enabling them to reconnect and resume communication without losing message state.

Hierarchical Topic Structure: Messages in MQTT are published to topics, which form a hierarchical structure delimited by slashes ("/"). Topics allow for flexible message routing and filtering, enabling subscribers to receive messages based on specific topics or using wildcards to match multiple topics.

Last Will and Testament (LWT): MQTT's "Last Will and Testament" function enables users to designate a message that the broker will publish in the case of an ungraceful disconnection. This function makes sure that clients can update others on their whereabouts, even in the event that they unexpectedly disconnect from the network.

MQTT packet structure: The structure of an MQTT packet consists of three main components: the Fixed Header, the Variable Header, and the Payload as shown in Fig.2. The Fixed Header is always there and includes the Packet Type, flags for DUP (duplicate delivery), QoS (Quality of Service), and RETAIN (whether the message should be retained by the broker), along with the Remaining Length, which indicates the number of bytes left in the packet. The Variable Header is optional and varies depending on the packet type, containing fields like the Protocol Name, Packet Identifier, and other relevant information depending on the packet type, such as CONNECT, PUBLISH, or SUBSCRIBE. The Payload, also optional, contains the actual message content, such as a client identifier, topic filters, or application messages, depending on the type of packet. This compact structure allows MQTT to be highly efficient, making it ideal for environments with limited resources.

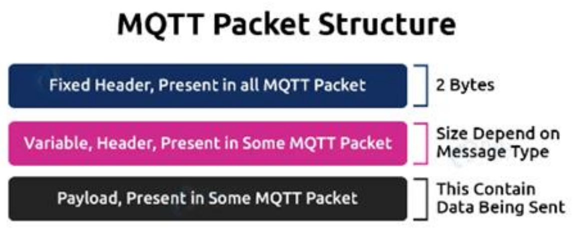
MQTT Packet Structure
Fixed Header, Present in all MQTT Packet
2 Bytes
Variable, Header, Present in Some MQTT Packet
Size Depend on Message Type
Payload, Present in Some MQTT Packet
This Contain Data Being Sent

Fig 2. MQTT Packet Structure

The remainder of this paper is structured as follows. In Section 2, the previous works are reviewed. The importance of authentication and types of authentication are introduced in Section 3. Then, the types of authentication in MQTT are analyzed in Section 4. The authentication methods in MQTT and different MQTT broker implementations are discussed in 5. Implementation challenges are summarized in 6. The paper is summarized and conclusion is given in Section 7 .

2. RELATED WORKS

This section provides review of the current work related to MQTT authentication.

Authentication can be based on credentials such as username and password or using an identification token created by the server. [1], [2], [3] have presented different surveys on the authentication mechanisms of MQTT brokers and different authentication schemes. The security mechanisms implemented by different MQTT brokers is studied by Jaidip et. al in [4]. In addition to the traditional authentication using username and password, token based authentication and JWT based authentication have been used for additional security of the system.

By employing token-based authentication and utilizing the secret key for both encryption and decryption, the study in [5] presented a lightweight authentication technique. Niruntasukrat [6] introduced an authorization mechanism for MQTT-based IoT systems. The authorization mechanism presented in this work is based on OAuth 1.0a, an open authorization standard for web applications. [7] have proposed the design and implementation of token based authentication of MQTT protocol in constrained devices. . In order to secure the communication protocol between the device and cloud using an SSL certificate, [8] have introduced a novel security mechanism for MQTT environments called MQTT-Auth, which is based on AugPAKE, on an authorization token and an authentication token. Authors in [9] have proposed a prototype access control system to manage the device node(publisher) and the authentication server(gateway). SSL certificate is used for establishing the secure connection and communication between the device and the authentication server. [10] proposed JWT based client authorization for MQTT systems. [11] have used the JSON Web Token (JWT) to build a token-based authentication mechanism on MQTT as a second authentication factor other than username and password.

3. TYPES OF AUTHENTICATION IN IOT

Authentications in IoT environments are crucial to ensure security, integrity, and trustworthiness of the interconnected devices and systems. Some of the authentication types used in IoT are:

3.1 Device Authentication

IoT has variety of heterogeneous, networked devices that vary in terms of size, shape, storage capacity, processing speed, and battery life. Tracing back the device's who, what, when, where, and other actions is a laborious task when there are so many devices connected to the network. Numerous studies have also demonstrated how difficult it is to authenticate devices because of the variety of connected gadgets. Pre-shared keys and public-key certificates have formed the foundation for several inter-device authentication methods that have been suggested thus far [12, 13]. Nguyen et al. [14] put forth a symmetric key protocol for peer-to-peer device authentication that makes use of identity-based encryption and is appropriate for a variety of application areas. An approach to IoT device authentication based on behavioral usage patterns was developed by Ashibani and Mahmoud [15]. Their authentication approach based on behavioral usage pattern. Their authentication model employs ML to detect any deviation from the normal pattern.

3.2 Identity-Based Authentication

Users, administrators, or applications interacting with IoT devices and systems should undergo authentication to access data, control devices, or perform administrative tasks. These authentication mechanisms help prevent unauthorized access and protect sensitive information. The author of [16] proposed the identity based authentication for IoT. An anonymous, secure, efficient, and unlinkable architecture for authentication was given by the authors in [18]. Their approach requires a lot of time due to the frequent session key modifications. For privileged devices, authors in [19] suggested a fast and error-free key agreement and authentication scheme that used channel state information as the common secret key. The author proposes in [20] an Identity-based Encryption (IBE) system as part of a Function-based Access Control (FAC) strategy for the Internet of Things. This proposed method prevents unauthorized functions from operating by offering incredibly fine-grained access control. The experimental results demonstrate that the proposed method is successful against a range of attack types, and the security analysis certifies the security of the suggested system.

3.3 Mutual Authentication

Mutual authentication ensures that both the IoT device and the server or service it interacts with authenticate each other. This two-way authentication mechanism provides security against impersonation attempts and guarantees that only reliable parties can communicate with one another. Nivethitha et al[21] proposed a mutual authentication scheme for managing end devices. Lu, Yanrong, et al. proposed a safe and efficient mutual authentication scheme for session initiation for IoT networks. Yanbin Zhang et al. suggested a mutual authentication approach to facilitate secure device-to-device communication [23]. Qingru Ma, Haowen Tan, and Tianqi Zhou suggested a mutual authentication mechanism for smart devices in Internet of Things (IoT)-enabled smart home systems[24].

3.4 Role-Based Access Control (RBAC)

RBAC mechanisms allow IoT devices, users, and applications to be assigned specific roles and permissions based on their identities and privileges. Niruntasukrut[6] used role-based access control ensures that only authorized entities can access certain resources, perform specific actions, or access sensitive data within the IoT ecosystem. A model-driven framework was developed by Bisma, Mariam, et al.[25] to guarantee role-based access management in Internet of Things devices.

3.5 Levels of Authentication

3.5.1 One-Factor Authentication (1FA)

It relies on a single type of credential to verify a user's identity. This is the most basic form of authentication. The authentication method can be a password which the user provides or a PIN, which is a personal identification number that the user must remember and input.

3.5.2 Two-Factor Authentication (2FA)

It requires two different types of credentials from two different categories to verify correct identity of the user. An extra layer of security is added to one-factor authentication. The methods can be a password or PIN along with a physical device like a Smartphone, security token, or smart card or biometrics like fingerprints, facial recognition, or iris scans.

- **Password + SMS Code:** The user enters their password and then an SMS containing code is sent to their mobile phone.
- **Password + Authenticator App:** The user enters their password and then a time-based code from an authenticator app (like Google Authenticator).

3.5.3 Three-Factor Authentication (3FA):

It involves three different types of credentials from three different categories, providing an even more secure method of verifying a user's identity. A password or PIN, a physical device like a smartphone, security token, or smart card along with Biometric details such as fingerprints, facial recognition, or iris scans.

- **Password + Security Token + Fingerprint:** The user enters their password, inserts a security token, and provides a fingerprint scan.
- **PIN + Smart Card + Iris Scan:** The user inputs a PIN, uses a smart card, and undergoes an iris scan.

Table. 3.1. Comparison of Different authentication factors

Type	*Factors Used*	*Examples*	*Merits*	*Challenges*
Single-Factor Authentication (SFA)	Single factor	Password, PIN	Simple, low cost	Low security, password management issues
Two-Factor Authentication (2FA)	Two factors	Password + SMS code, Password + Authenticator app	Enhanced security, deters unauthorized access	User experience, reliance on additional factors
Multi-Factor Authentication (MFA)	Two or more factors	Password + Security Token + Fingerprint	Highest security level, mitigates various threats	Complexity, user convenience, cost and infrastructure

Each level of authentication adds an additional layer of security but also introduces more complexity and potential inconvenience for users. The choice of which authentication method to use depends on the security requirements and the potential threats faced by the system or organization.

4. AUTHENTICATION MECHANISMS IN MQTT

Authentication mechanisms are typically implemented by MQTT brokers or servers as part of their extended functionality rather than being inherent to the MQTT protocol itself. However, MQTT does support a flexible authentication process, enabling clients and servers to authenticate each other through various methods.

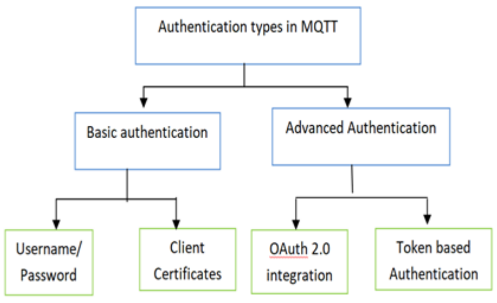

Fig 3: Authentication types in MQTT

4.1 Basic Authentication schemes

There are 2 basic authentication methods used in MQTT. They are username and password (credentials) and using client certificates.

4.1. 1 Username/password authentication in MQTT

Username/password authentication in MQTT provides a basic yet effective method for authenticating clients (devices or applications) when connecting to an MQTT broker. MQTT clients are configured with a username and password, which is typically provided by the MQTT broker administrator during client setup. The username/password combination, serves as the credentials that clients use to authenticate themselves to the MQTT broker when establishing a connection. Upon successful authentication, the connection has been accepted and the client can now proceed to publish messages, subscribe to topics, or perform other MQTT operations on the broker. The advantage of this method is that it provides basic security but it is very important to safeguard the passwords by storing them securely and then transmitting to prevent unauthorized access but the disadvantage is that the username/password authentication in MQTT is relatively simple and may not be suitable for high-security applications.

4.1.2 Client certificates

In MQTT (Message Queuing Telemetry Transport) protocol, client certificates play a vital role in ensuring secure communication between MQTT clients and brokers. Client certificates are part of the Secure Sockets Layer/Transport Layer Security (SSL/TLS) protocol, which is used for a secure connection between clients and brokers. When a client connects to an MQTT broker, it can present a client certificate to authenticate itself. A trusted Certificate Authority (CA) issues this certificate and contains all the information about the client, such as its identity and public key. The broker verifies the client's certificate against its list of trusted CAs to ensure the client's authenticity.

The advantage of this method is that **c**lient certificates offer stronger authentication compared to username/password, as they are based on cryptographic principles and MQTT messages exchanged between the client and broker are encrypted using SSL/TLS, providing confidentiality and integrity of data in transit but the limitation is that the MQTT clients need to be configured to use client certificates for authentication. This involves specifying the client's private key and presenting the client certificate during the SSL/TLS handshake with the MQTT broker

4.2 Advanced authentication schemes

The two advanced authentication schemes widely used are OAuth2.0 integration and token based authentication.

a) **OAuth 2.0 integration:** OAuth 2.0 is an authorization framework designed to enable secure access to resources on behalf of a user by third-party applications without requiring the user's credentials to be shared.

   OAuth 2.0 offers several advantages as an authorization framework for securing access to resources in web and API-based applications. One of its key strengths is its flexibility, which allows it to be adapted to various use cases and scenarios. It provides a standardized approach to authentication and authorization, promoting interoperability between different systems and services. It supports fine-grained access control through the use of scopes, allowing clients to request only the permissions they need to perform specific actions, thereby minimizing the risk of unauthorized access.

b) **Token-based authentication**: It provides a secure method for clients to authenticate themselves to the MQTT broker using access tokens. Clients obtain an access token from an authentication server or authorization server using an authentication mechanism such as OAuth 2.0 or JWT (JSON Web Tokens).

Token-based authentication offers several advantages in securing web and API-based applications. One of its primary strengths is its statelessness, which eliminates the need for servers to store session data, resulting in a more scalable and distributed architecture. Tokens can be easily distributed and verified across different services and servers, allowing for seamless integration and interoperability. Additionally, token-based authentication enhances security by reducing the risk of attacks such as session hijacking and cross-site request forgery (CSRF). Since tokens are typically generated using cryptographic algorithms and contain limited information, they are less susceptible to tampering and exploitation compared to traditional session-based authentication mechanisms. Furthermore, token-based authentication facilitates the implementation of single sign-on (SSO) and federated identity management solutions, enabling users to access several different applications and services with a pair of credentials.

Token-based authentication also has some limitations. One challenge is the need to securely transmit and store tokens to prevent unauthorized access. If tokens are compromised or leaked, they can be used by attackers to impersonate legitimate users and gain unauthorized access to resources. Additionally, token-based authentication introduces additional complexity to the authentication process, particularly in scenarios where tokens need to be refreshed or revoked to maintain security

4.3 Enhanced security measures in MQTT

Enhancing security in MQTT deployments involves implementing various measures to protect the data confidentiality, integrity of data, and availability of the data in MQTT communication. Some enhanced security measures specific to MQTT:

4.3.1 Transport Layer Security (TLS)

Transport Layer Security (TLS) provides a secure communication channel between a client and a server. A secure communication channel is offered by Transport Layer Security (TLS) between a client and a server.TLS is a cryptographic protocol that establishes a secure connection between the client and the server by negotiating different parameters through a handshake procedure. Once the handshake is finished, the client and server can communicate with ecncrypted messages, making it impossible for an attacker to listen in on any part of the conversation. Clients utilize an X509 certificate that servers offer to confirm the server's identity. TLS encrypts data transferred between the client and the server, guaranteeing that it cannot be intercepted and interpreted by unauthorized parties. It offers security and data integrity. TLS ensures that there hasn't been any tampering with the data during transmission between the client and server.

4.3.2 Message-level security

It refers to the practice of securing individual messages exchanged between software components or services within a distributed system. Unlike transport-level security, which focuses on securing the entire communication channel, message-level security is concerned with securing the contents of each individual message. Messages are encrypted before transmission and decrypted upon receipt, to ensure that only the devices/clients that are authorized will access the content of the message. This helps prevent eavesdropping and unauthorized access to important, sensitive information. Data integrity, non repudiation and interoperability are guaranteed by the message level security.

Even if an attacker can gain access to the transmission channel or network, they cannot access the contents of the messages without the appropriate cryptographic keys. It is particularly useful in environments where messages may pass through multiple servers or intermediaries, ensuring that only the intended recipients can read the content. It is commonly used in secure email communications, messaging applications, and any other scenarios where data security at the message level is crucial.

Table 4.1. Summary of different authentication mechanisms

Authentication Type	*Description*	*Pros*	*Cons*
Username/Password	Clients authenticate by sending a username and password with the connection.	Simple to implement and widely supported.	Less secure, passwords need to be stored and managed securely.
Client Certificates	Uses SSL/TLS with X.509 certificates for client authentication.	Highly secure, mutual authentication.	More complex setup, certificate management needed.
Pre-shared Keys (PSK)	Clients use pre-shared keys for authentication over SSL/TLS.	Secure and efficient.	Key distribution and management can be challenging.
OAuth 2.0	Uses OAuth 2.0 tokens for authentication.	Modern, flexible, integrates with many identity providers.	More complex implementation, requires additional infrastructure.
JWT (JSON Web Token)	Clients use JWTs for stateless authentication.	Compact, self-contained tokens, can carry metadata.	Token management and expiration handling needed.
SASL (Simple Authentication and Security Layer)	Provides a flexible framework for various authentication mechanisms.	Supports multiple authentication methods.	Complexity depends on the chosen mechanism.
Anonymous	No authentication, open access.	Very simple, no setup required.	Not secure, should only be used in trusted environments.

5. IMPLEMENTATION OF AUTHENTICATION TECHNIQUES IN DIFFERENT MQTT BROKERS

There are several MQTT brokers available, each with its own features, advantages, and specific use cases. Here are some of the popular MQTT brokers:

5.1 Mosquitto

Eclipse Mosquitto is an open-source MQTT broker that is widely used for its simplicity and compliance with MQTT standards. It is Lightweight and efficient, supports MQTT versions 3.1, 3.1.1, and 5.0., easy to set up and configure and available on multiple platforms including Windows, macOS, and Linux.

5.2 HiveMQ

HiveMQ is a commercial MQTT broker designed for high scalability and enterprise-grade deployments. It supports MQTT versions 3.1.1 and 5.0 It has high availability and clustering capabilities, provides an intuitive management interface and has extensive integration options with other enterprise systems.

5.3 Amazon AWS IoT Core

Amazon AWS IoT Core is a managed cloud service that provides MQTT broker functionality as part of the AWS IoT suite. It is a fully managed service with auto-scaling, supports MQTT versions 3.1.1 and 5.0. It integrates seamlessly with other AWS services and has built-in security features including mutual authentication and end-to-end encryption.

5.4 Azure IoT hub

Azure IoT Microsoft Azure offers a managed service called Azure IoT Hub, which serves as a central messaging hub for two-way communication between IoT applications and the devices it manages. It offers an extremely scalable and secure platform for managing, connecting, and keeping an eye on IoT devices.

5.5 Google Cloud IoT Core

Google Cloud IoT Core is a fully managed service that includes MQTT broker functionality within the Google Cloud Platform. It is a fully managed and scalable MQTT broker. It supports MQTT version 3.1.1.It integrates with other Google Cloud services and provides robust security and data analytics capabilities.

5.6 EMQX

EMQX (formerly known as EMQ) is an open-source, highly scalable, and extensible MQTT broker. It supports MQTT versions 3.1, 3.1.1, and 5.0. It has high throughput and low latency, clustering and load balancing capabilities and built-in support for various protocols (CoAP, LwM2M, etc.) and databases.

5.7 VerneMQ

VerneMQ is an open-source MQTT broker that focuses on high availability and scalability. It supports MQTT versions 3.1, 3.1.1, and 5.0. It is designed for clustering and horizontal scaling. It has robust plugin architecture and supports various authentication and authorization mechanisms.

Table 5.1. Summary of different MQTT broker implementations for security

MQTT Broker	***Authentication Methods***	***Description***
Mosquitto	Username and Password	Clients provide credentials during the connection process.
	Client Certificates	Devices authenticate using SSL/TLS certificates during the TLS handshake.
	External Authentication	Integration with external mechanisms through plugins (e.g., LDAP, OAuth 2.0).
	ACL (Access Control List)	Fine-grained access control policies based on client identities or topics.
HiveMQ	Username and Password	Clients provide credentials during the connection process.
	Client Certificates	SSL/TLS certificates required for client connections.
	OAuth 2.0	Clients obtain access tokens from OAuth 2.0 authorization servers.
	Custom Authentication Plugins	Custom logic or integration with LDAP/Active Directory.
	Token-Based Authentication	Clients present tokens (e.g., JWT) for authentication.
AWS IoT Core	X.509 Certificates	Devices use unique certificates signed by a trusted CA for authentication.
	AWS Signature Version 4	HTTP-based authentication using AWS credentials.
	AWS IAM Integration	Fine-grained access control with policies based on device ID, certificate attributes, or IAM roles.
Azure IoT Hub	SAS (Shared Access Signature) Tokens	Devices authenticate using SAS tokens in the MQTT CONNECT packet.
	X.509 Certificates	Client certificates used for secure connection with the IoT Hub.
	Azure Active Directory Integration	Fine-grained access control based on roles and permissions.
Google Cloud IoT	JWT (JSON Web Tokens)	Devices present JWTs signed by a Google Cloud service account.
	Mutual TLS (mTLS)	Both device and server present certificates during the TLS handshake.
	Google Cloud IAM Integration	Define access policies and permissions for IoT resources based on roles.

EMQX Cloud	Username and Password	Clients provide credentials during the connection process.
	Client Certificates	Devices present SSL/TLS certificates for mutual authentication.
	External Authentication	Integration with external systems through plugins (e.g., LDAP, OAuth 2.0).
	Token-Based Authentication	Clients authenticate using pre-generated tokens, such as JWT.
VerneMQ	Username and Password	Clients provide credentials during the connection process.
	Client Certificates	Devices authenticate using SSL/TLS certificates during the TLS handshake.
	External Authentication	Integration with external systems through plugins (e.g., LDAP, OAuth 2.0).
	Token-Based Authentication	Clients authenticate using pre-generated tokens, such as JWT.

6. IMPLEMENTATION CHALLENGES

Implementing authentication in MQTT presents several challenges. One key difficulty is achieving a balance between security and performance, as more robust authentication methods like TLS certificates enhance security but may strain the limited computational power of IoT devices, potentially slowing data flow. Additionally, managing and scaling device credentials, whether through usernames, passwords, or token-based approaches, can become complex and resource-intensive. Securing the storage and distribution of these credentials is another logistical challenge, particularly for devices in unsecured or remote locations. Some of the implementation challenges associated with MQTT authentication techniques which are encapsulated in the table below.

Table 6.1. Different Research challenges in MQTT broker security

Challenge	*Description*
Resource Constraints	Many MQTT clients run on resource-constrained devices with limited processing power, memory, and energy. Because of the memory requirements and computational complexity, implementing sophisticated authentication mechanisms like public key infrastructure (PKI) or digital certificates might be difficult.
Compatibility and Interoperability	MQTT is designed to be interoperable across different platforms and programming languages. Authentication mechanisms need to be compatible with various MQTT client libraries and brokers to ensure seamless integration. Achieving compatibility while maintaining security can be a challenging task.
Key Management	In environments with a large number of MQTT clients, managing authentication credentials such as usernames, passwords, client certificates, or API keys can be cumbersome. Key management becomes particularly challenging in scenarios where devices are frequently added, removed, or updated.
Secure Communication Channels	MQTT typically relies on TCP/IP as its underlying transport protocol. Ensuring the security of communication channels is essential to prevent eavesdropping and man-in-the-middle attacks. However, securing TCP/IP connections using Transport Layer Security (TLS) can introduce additional complexity, especially on resource-constrained devices.
Scalability	As the number of MQTT clients and brokers in a network grows, the authentication infrastructure must scale accordingly to handle the increased load. Scalability considerations include authentication server performance, database capacity, and network bandwidth requirements.
Security Risks	Poorly implemented authentication mechanisms can introduce security vulnerabilities such as brute-force attacks, credential stuffing, or authentication bypasses. Misconfigured authentication settings can inadvertently expose sensitive data or compromise the integrity of the MQTT network.
Organizational Best Practices	Organizations deploying MQTT-based solutions should conduct thorough security assessments and follow best practices to mitigate potential risks and ensure the confidentiality, integrity, and availability of their MQTT deployments.

7. Conclusion

Authentication in MQTT is a critical aspect of securing IoT environments and messaging applications. This paper has reviewed the foundational concepts of MQTT, various authentication mechanisms, and their implementation in different MQTT brokers. From simple username/password authentication to advanced methods like OAuth 2.0 and JWTs, each approach offers different levels of security and complexity. By understanding the strengths and weaknesses of these methods, organizations can better select and implement the most appropriate authentication strategy for their specific use case. while traditional authentication methods provide a foundational layer of security, they may not suffice in scenarios requiring higher levels of security and scalability. Advanced techniques like OAuth 2.0 and token-based authentication offer more flexibility and security but introduce additional complexity and infrastructure requirements. The analysis further reveals that no single authentication method can address all security challenges posed by diverse IoT environments. As such, MQTT broker implementations often provide a combination of authentication methods, allowing for layered security strategies tailored to specific use cases and risk profiles. However, challenges such as managing certificates, tokens, and integrating external authentication services remain significant, particularly in resource-constrained environments typical of many IoT applications

References

[1] Karthikeyan, S.; Patan, R.; Balamurugan, B. Enhancement of Security in the Internet of Things (IoT) by Using X. 509 Authentication Mechanism. In Recent Trends in Communication, Computing, and Electronics; Springer: Singapore, 2019; pp. 217–225.

[2] M. Calabretta, R. Pecori and L. Veltri, "A Token-based Protocol for Securing MQTT Communications," 2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 2018, pp.1-6,doi: 0.23919/SOFTCOM.2018.8555834.

[3] Calabretta, Marco & Pecori, Riccardo & Vecchio, Massimo & Veltri, Luca. (2018). MQTT-Auth: a Token-based Solution to Endow MQTT with Authentication and Authorization Capabilities. Journal of Communications Software and Systems. 14. 10.24138/jcomss.v14i4.604.

[4] 4.A. Bhawiyuga, M. Data and A. Warda, "Architectural design of token based authentication of MQTT protocol in constrained IoT device," 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA), Lombok, Indonesia, 2017, pp. 1-4.

[5] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul and A. Panya, "Authorization mechanism for MQTT-based Internet of Things," 2016 IEEE International Conference on Communications Workshops (ICC), Kuala Lumpur, 2016, pp. 290-295. DOI: 10.1109/ICCW.2016.7503802.

[6] S. Shin, K. Kobara, Chia-Chuan Chuang and Weicheng Huang, "A security framework for MQTT," 2016 IEEE Conference on Communications and Network Security (CNS), Philadelphia, PA, 2016,pp.432-436.DOI: 10.1109/CNS.2016.7860532.

[7] M. A. A. da Cruz, J. J. P. C. Rodrigues, P. Lorenz, V. V. Korotaev and V. H. C. de Albuquerque, "In.IoT—A New Middleware for Internet of Things," in IEEE Internet of Things Journal, vol. 8, no. 10, pp. 7902-7911, 15 May15, 2021, doi: 10.1109/JIOT.2020.3041699.

[8] F. A. Shodiq, R. R. Pahlevi and P. Sukarno, "Secure MQTT Authentication and Message Exchange Methods for IoT Constrained Device," 2021 International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA), Bandung, Indonesia, 2021, pp. 70-74.

[9] Shingala, "JSON Web Token (JWT) based client authentication in Message Queuing Telemetry Transport (MQTT)," 2019, doi.org/10.48550/arXiv.1903.02895.

[10] B.S.Bali, F. Jaafar, P.Zavarasky, "Lightweight authentication for MQTT to improve the security of IoT communication" ICCSP '19: Proceedings of the 3rd International Conference on Cryptography, Security and Privacy, January 2019, Pages 6–12, https://doi.org/10.1145/3309074.3309081.

[11] A. A. Wardana and R. S. Perdana, "Access Control on Internet of Things based on Publish/Subscribe using Authentication Server and Secure Protocol," 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE), Bali, Indonesia, 2018, pp. 118-123, doi: 10.1109/ICITEED.2018.8534855.

[12] Bersani, Florent, and Hannes Tschofenig. The EAP-PSK protocol: A pre-shared key extensible authentication protocol (EAP) method. No. rfc4764. 2007.

[13] Clancy, T., and H. Tschofenig. *Extensible Authentication Protocol-Generalized Pre-Shared Key (EAP-GPSK) Method*. No. rfc5433. 2009.

[14] Nguyen, Kim Thuat, Nouha Oualha, and Maryline Laurent. "Authenticated key agreement mediated by a proxy re-encryptor for the internet of things." *Computer Security–ESORICS 2016: 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part II 21*. Springer International Publishing, 2016.

[15] Ashibani, Yosef, and Qusay H. Mahmoud. "A multi-feature user authentication model based on mobile app interactions." *IEEE Access* 8 (2020): 96322-96339.

[16] Salman, Ola, et al. "Identity-based authentication scheme for the Internet of Things." *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 2016.

[17] P. Kumar, A. Braeken, A. Gurtov, J. Iinatti and P. H. Ha, "Anony-mous secure framework in connected smart home environments", IEEE Transactions on Information Forensics and Security, vol. 12, no. 4, pp. 968-979, 2017.

[18] W. Xi, C. Qian, J. Han, K. Zhao, S. Zhong, X.-Y. Li, et al., "Instant and robust authentication and key agreement among mobile devices", Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 616-627, 2016.

[19] H. Yan, Y. Wang, C. Jia, J. Li, Y. Xiang and W. Pedrycz, "IoT-FBAC: Function-based access control scheme using identity-based encryption in IoT", Future Generation Computer Systems, vol. 95, pp. 344-353, Jun. 2019.

[20] B. B. Gupta, A. Gaurav, K. T. Chui and C. -H. Hsu, "Identity-Based Authentication Technique for IoT Devices," *2022 IEEE* International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2022, pp. 1-4, doi: 10.1109/ICCE53296.2022.9730173.

[21] Annashree Nivethitha, S., Chanthini Baskar, and Manivannan Doraipandian. "Mutual Authentication Scheme for the Management of End Devices in IoT Applications." Advances in Electrical and Computer Technologies*:* Select Proceedings of ICAECT 2019. Singapore: Springer Singapore, 2020. 221-231.

[22] Lu, Yanrong, et al. "A secure and efficient mutual authentication scheme for session initiation protocol." Peer-to-Peer Networking and Applications 9 (2016): 449-459.

[23] Qingru Ma, Haowen Tan, Tianqi Zhou, Mutual authentication scheme for smart devices in IoT-enabled smart home systems,Computer Standards & Interfaces,Volume 86,2023,

[24] Zhang, Yanbin, et al. "A mutual authentication scheme for establishing secure device-to-device communication sessions in the edge-enabled smart cities." *Journal of Information Security and Applications* 58 (2021): 102683.

[25] Ma, Qingru, Haowen Tan, and Tianqi Zhou. "Mutual authentication scheme for smart devices in IoT-enabled smart home systems." *Computer Standards & Interfaces* 86 (2023): 103743.

[26] Bisma, Mariam, et al. "A model-driven framework for ensuring role based access control in IoT devices." Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence. 2020.