applied sciences

MDPI

Article

A Multi-Tier MQTT Architecture with Multiple Brokers Based on Fog Computing for Securing Industrial IoT

Hassan Kurdi [1,2,*] and Vijey Thayananthan [2]

[1] Department of Computer Science, College of Computer Science and Engineering, Taibah University, Madinah 42353, Saudi Arabia
[2] Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia; vthayanathan@kau.edu.sa
* Correspondence: hkurdi@taibahu.edu.sa or hkurdi0009@stu.kau.edu.sa

Abstract: With the rapid growth of internet-connected devices and their resource-constrained capabilities, the current authentication mechanisms are unable to meet the complex IoT application requirements, such as in the Industrial Internet of Things (IIoT), due to the increased computation, communication, and storage overhead arising from these mechanisms. In the IIoT, machine-to-machine (M2M) communication is an underlying technology where devices (e.g., sensors, actuators, and controllers) can be enabled to exchange information autonomously; thus, the massive data generated by these devices can increase latency, network congestion, and the complexity of security management. Message queue telemetry transport (MQTT) is one of the promising M2M protocols used in the IoT that could encounter such issues because it relies on a central broker in the cloud and implements a heavyweight authentication mechanism based on TLS. Therefore, this paper proposes an MQTT architecture with multi-tier brokers based on fog computing, where each broker is deployed with an authentication manager. In addition, the paper presents a lightweight mutual authentication scheme based on hash function and XOR operation. Comparing the results given in the benchmark, the overall performance of our scheme shows that storage and communication overheads are reduced to 89% and 23%, respectively. Furthermore, our system can resist against several cyberattacks and provide scalability.

Keywords: lightweight authentication; message queue telemetry transport (MQTT); Internet of Things (IoT); fog computing

Citation: Kurdi, H.; Thayananthan, V. A Multi-Tier MQTT Architecture with Multiple Brokers Based on Fog Computing for Securing Industrial IoT. *Appl. Sci.* **2022**, *12*, 7173. https://doi.org/10.3390/app12147173

Academic Editors: Antonella Petrillo and Agostino Forestiero

Received: 22 April 2022
Accepted: 14 July 2022
Published: 16 July 2022

1. Introduction

The evolution and expansion of networking technologies have enabled the establishment of large-scale connectivity among devices and applications, which has led to the emergence of the Internet of Things (IoT) concept. According to [1,2], the number of connected IoT devices in 2016 was 4.6 billion, then 10 billion in 2019, and is expected to be more than 14 and 27 billion in 2022 and 2025, respectively. This rapid growth of internet-connected devices can cause several challenges, such as high latency, network congestion, and bandwidth consumption, due to the fact that the cloud architecture does not meet the requirements of the massive growth of data that are generated by IoT devices [3].

The challenges will be more critical in applications that require fast responses, as in the Industrial Internet of Things (IIoT), in which machine-to-machine (M2M) communication is an underlying technology where devices (e.g., sensors, actuators, controllers, and gateways) are enabled to exchange data with each other in an autonomous way without human intervention. In industrial applications, low latency and real-time processing are fundamental features that are not supported sufficiently in cloud computing [4]. As a consequence, the fog computing paradigm emerged as a complementary technology with cloud computing to solve such challenges. Fog computing can provide characteristics such

as low latency, location awareness, geo-distribution, increased data security, and real-time processing [5]. In addition, it aims to bring computing, storage, and network capabilities closer to the end users.

Message queue telemetry transport (MQTT) is a messaging protocol and one of the promising M2M protocols in the IoT based on the publish–subscribe model, which allows multiple clients (publishers/subscribers) to communicate with each other through a broker, can encounter the aforementioned challenges. This is because the communication between clients relies on a central broker in the cloud. To mitigate and overcome these challenges, we need to leverage fog computing capabilities to develop MQTT architecture by deploying multiple brokers in the fog layer. In this case, the load of data transmission, processing, and storage will be coordinated between brokers in the fog and the cloud, and thus, the overhead on the cloud will be reduced.

However, this integration between the fog and the cloud for MQTT architecture will create new security challenges. Since this paper focuses on authentication, the major concern in designing the authentication system for MQTT based on fog computing is providing a reliable authentication mechanism while enhancing performance and scalability. The reasons are, firstly, that the current authentication mechanisms do not satisfy resource-constrained IoT devices due to the complex computational costs involved in these mechanisms, which consume the processing, memory, and storage capabilities of IoT devices. Secondly, since existing solutions utilize the central cloud entity as an authentication server, scalability issues will arise due to the growing number of devices, which may cause bottlenecks, network congestion, and high delays. Thirdly, the large computational load and high communication overhead on a remote central authentication entity might cause high transmission, queuing, or processing delays. The main objectives of this paper are as follows:

1. Propose a scalable MQTT architecture with multi-tier brokers that are based on fog computing.
2. Present a lightweight mutual authentication scheme that is based on hash function and XOR operation.
3. Build an authentication manager associated with each broker to conduct the authentication process for that broker and its connected group of clients/brokers independently.

The lightweight mutual authentication scheme and scalable authentication architecture for the publish–subscribe model (MQTT) that is based on multiple brokers' communication is still an open challenge, and this is the motivation for our study. Unfortunately, most of the current studies either focus on presenting authentication mechanisms for traditional MQTT that are based on a single broker without taking into account multiple brokers' communication or propose distributed MQTT architectures with multiple brokers without taking into account the security solutions, such as authentication, while the studies that provided authentication schemes for distributed MQTT architectures that are based on multiple brokers lack lightweight mutual authentication and a scalable authentication architecture.

The main contributions of this paper can be summarized as follows:

1. The proposed architecture contributes to handling the massive amount of M2M communication in IIoT by clustering the clients into groups that connect to a single local broker. Further, the architecture will reduce latency, network congestion, and bottlenecks.
2. The proposed architecture is designed in a hierarchical structure to enable delivering the requested data to the MQTT client from other brokers that the client is not connected to.
3. Implementing a lightweight mutual authentication scheme using hash function and XOR operation is more feasible for resource-constrained IoT devices to mitigate computation costs and communication overhead.
4. Deploying an authentication manager in each broker will allow the management of security independently for the broker and its group of connected clients/brokers.

This will reduce the possibility of security threats and decrease the complexity of security management.

The rest of the paper is structured as follows: Section 2 provides a brief background of MQTT and presents the related works. Section 3 introduces the proposed architecture of the multi-tier MQTT broker based on fog computing (Mu-TiMB) as well as the proposed lightweight authentication scheme. Section 4 presents an informal security analysis and the performance results. The evaluation is discussed in Section 5. Finally, the conclusion and directions for future work are provided in Section 6.

2. Background and Related Works

This section introduces a brief background of MQTT, including the type of messages, the quality-of-service levels (QoS), and the MQTT topic. In addition, the section on the related works presents current studies on MQTT authentication and distributed MQTT architecture with multiple brokers.

2.1. Message Queue Telemetry Transport (MQTT)

MQTT is a messaging protocol based on the publish–subscribe model, which is similar to the client–server model, but the server running MQTT is called a broker. The protocol is designed for resource-constrained devices and low-bandwidth and high-latency networks, and it attempts to provide reliability and ensure data delivery. In addition, it is ideal for mobile IoT applications and machine-to-machine (M2M) communication. MQTT provides bidirectional communication between clients (publisher/subscriber) and a broker. The publisher is decoupled from the subscriber, and there is no direct communication between the publisher and the subscriber, except through the broker, which handles, filters, and distributes the messages between the clients. As shown in Table 1, MQTT has 14 types of messages that are sent to and received from the broker. Security in MQTT is a username/password-based authentication in plain text that is protected by the cryptographic protocol Secure Sockets Layer/Transport Layer Security (SSL/TLS). This protocol is not considered to be lightweight for resource-constrained devices and increases the computational, communication, and storage overhead [6].

MQTT defines three quality-of-service (QoS) levels: QoS 0, QoS 1, and QoS 2. In QoS 0 (at most once), the publisher sends the message one time to the broker, which in turn sends it one time to the subscriber. No acknowledgment will be sent by the receiver, and the message will not be resent by the sender. QoS 1 (at least once) is the default level used in MQTT, and it guarantees that the PUBLISH message from the publisher to broker, and then from the broker to subscriber, is delivered at least one time. In other words, if an acknowledgment of the message (PUBACK) is not obtained by the receiver, the sender (either the publisher or broker) will resend the message. Finally, QoS 2 (exactly once) is utilized when message loss and duplication are not acceptable. QoS 2 performs a two-step acknowledgment (PUBLISH–PUBREC) and (PUBREL–PUBCOMP) process to ensure that the message is delivered one time. This level is more reliable but increases the communication overhead.

The clients' communication in MQTT using QoS 1 is depicted in Figure 1, where the subscriber is the client who requests specific topic(s) by sending the SUBSCRIBE message to the broker. The PUBLISH message is used by the publisher, who sends the topic to the broker, which in turn uses the PUBLISH message again to send the topic to the subscriber. The MQTT topic is a form of addressing and represents the type of data that are sent by the publisher and received by the subscriber. The topic name usually uses the forward slash (/), similar to folders in a file system, to separate each level within the topic tree. It is possible to subscribe to multiple topics using wildcards represented by # or +. The first symbol means a multi-level wildcard, and the latter means a single-level wildcard. For example, the topic "home/groundfloor/livingroom/#" allows the client to subscribe to all subtrees belonging to the topic such as "home/groundfloor/livingroom/temperature", "home/groundfloor/livingroom/humidity" and "home/groundfloor/livingroom/gas", whereas the topic "home/groundfloor/+/temperature" allows the client to subscribe to all topics related

to temperature from every single device, such as "home/groundfloor/livingroom/temperature", "home/groundfloor/kitchen/temperature" and "home/groundfloor/bedroom/temperature".

Table 1. MQTT message types.

Message Type	Value	Flow	Description
CONNECT	1	Client to broker	Request to connect
CONNACK	2	Broker to client	Connect acknowledgement
PUBLISH	3	Publisher to broker Broker to subscriber	Publish message
PUBACK	4	Broker to publisher Subscriber to broker	Publish acknowledgement (used in QoS 1)
PUBREC	5	Broker to publisher Subscriber to broker	Publish received (used in QoS 2)
PUBREL	6	Publisher to broker Broker to subscriber	Publish release (used in QoS 2)
PUBCOMP	7	Broker to publisher Subscriber to broker	Publish complete (used in QoS 2)
SUBSCRIBE	8	Subscriber to broker	Subscribe request
SUBACK	9	Broker to subscriber	Subscribe acknowledgement
UNSUBSCRIBE	10	Subscriber to broker	Unsubscribe request
UNSUBACK	11	Broker to subscriber	Unsubscribe acknowledgement
PINGREQ	12	Client to broker	Ping (keep alive) request
PINGRES	13	Broker to client	Ping (keep alive) response
DISCONNECT	14	Client to broker	Client disconnecting

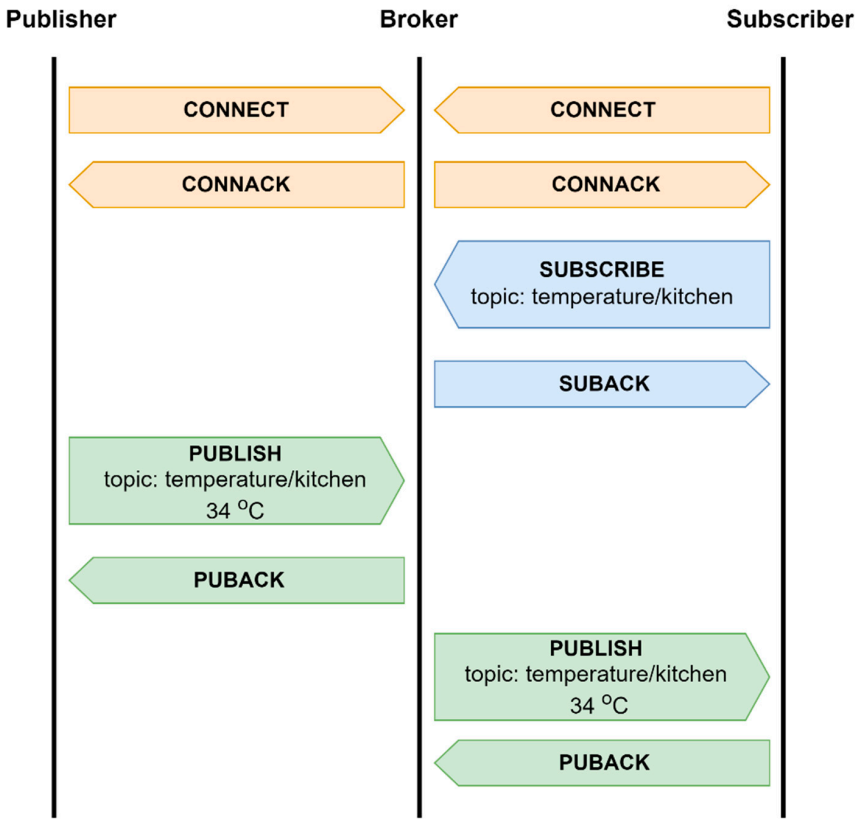


Figure 1. MQTT communication protocol (QoS 1).

2.2. Related Works

This section reviews the current studies related to MQTT, which are categorized into three parts based on those involving only the authentication property, those involving only a distributed MQTT architecture with multiple brokers, and those applying an authentication scheme for a distributed MQTT architecture with multiple brokers.

Among the studies that only focused on authentication mechanisms for MQTT, the authors of [7] proposed an approach called MQTT-Auth, which used two tokens, one to authenticate published topics and the other to grant authorizations to enable subscribers to have different access privileges. MQTT-Auth is based on the augmented PAKE (AugPAKE) algorithm, which is a client–server session key establishment protocol. AugPAKE establishes two session keys, one between the publisher and the broker and the other between the broker and the subscriber. The presented solution costs a high computation load and message exchanges [8]. The paper in [9] presented a token-based authentication mechanism for resource-constrained devices in the MQTT protocol. The mechanism is based on the JSON web token (JWT) authentication server, which is responsible for issuing tokens to publishers and subscribers after validating their credentials. After receiving the connection, then obtaining the token from the publisher and subscriber, the broker checks the validity of the token with the authentication server. The number of message exchanges in this system is high because each token provided to clients needs to be validated by a broker through JWT.

In [10], a lightweight authentication mechanism was introduced using tokens and secret keys for secure connections. The secret key is periodically updated using the chaotic algorithm to provide high diversity among consecutive security keys. To maintain diversity among the consecutive keys, the appropriate chaotic parameters are selected based on the distance entropy. The paper did not calculate the performance of the chaotic algorithm for the broker. However, it can be said that the frequent updating of the key by the broker could cause communication and processing overhead. An extension for MQTT called authenticated publish and subscribe (AUPS) was proposed in [11]. AUPS ensures the authentication and authorization of data by integrating a policy enforcement mechanism that is based on attribute-based encryption (ABE), where the data are encrypted by the publisher based on a collection of conditions in relation to the access policy, and if the subscriber fulfills the access policy, the ciphertext can be decrypted. However, ABE is not recommended for resource-constrained devices because the cryptographic overhead increases as the number of attributes increases [8].

The paper in [12] provides an authentication and authorization mechanism for the MQTT protocol using an HMAC-based one-time password (HOTP) and the OAuth 2.0 framework. The authentication is conducted in two stages. The first stage is validating the access token, which is used as a username through the WSO2is identity server. Secondly, the password, which is updated using HOTP, will be authenticated every time the connection is established between clients and brokers through MongoDB. The issue in this scheme is the increased message exchanges because of the number of stages that require more than one authentication entity. The paper in [13] proposes a multi-tier authentication mechanism called S-MQTT for delivering secure communication between end devices. The mechanism is based on key policy/cipher policy attribute-based encryption (KP/CP ABE) using lightweight elliptic curve cryptography (ECC). However, the issue with ABE is that the registration of nodes in the broker requires sending a set of attributes, which may cause high computation overhead. In [14], the authors proposed the Secure Reliable Message Communication (SEC-RMC) protocol for MQTT using a lightweight encryption algorithm to secure the data transmission at the transport layer. The MQTT broker is responsible for the client's registration and key management. The transmitted messages are encrypted and decrypted using the Advanced Encryption Standard (AES) algorithm. The results showed that there is an improvement in the packet deliverance ratio (PDR) and energy utilization compared to TLS. However, the paper did not present a security analysis or performance evaluation (such as computation costs, communication overhead,

and storage requirements) of the proposed system. In addition, the proposed protocol did not consider broker bridging, where brokers can communicate with each other to deliver messages between MQTT clients.

The paper in [15] presented a lightweight security solution using elliptic curve cryptography (ECC) for the publish–subscribe protocol based on fog computing. The fog broker contains three modules: (1) The key management module is responsible for providing public parameters and generating secret keys. (2) The subscription module is used for authentication and authorization. (3) The publication module is responsible for encrypting sent data and transmitting them to subscribers. The scheme provides a shorter key length, reduces overhead, and improves scalability compared to RSA. However, the paper did not present a broker bridging mechanism, and the scheme was not implemented in an MQTT architecture with multiple brokers. In addition, one of the disadvantages of the scheme is that key management and key revocation issues were not addressed [16]. Thus, implementing the scheme in a large-scale MQTT network with brokers' communication is a complicated process. The study in [17] proposed a lightweight authentication and authorization framework for inter-device communication in a distributed environment using ECC with Transport Layer Security (TLS). The system utilized the MQTT protocol for broadcast-based data transmission. The cloud architecture has a combination of servers, such as the cloud authentication server (CAS), policy enforcement server (PES), network function virtualization, and broker server. The result showed that there is a high computation cost for encryption and decryption; moreover, the increased number of devices will increase the time of the authentication process [18].

There are also studies that proposed a distributed architecture for MQTT without focusing on security solutions. The authors of [19] modified the traditional MQTT protocol by presenting a software-defined multi-tier edge computing model. The system contains three layers. The first layer and the third layer are the sensing network (sensors and actuators) and data center (cloud), respectively. The second layer is the fog computing layer, which is divided into four tiers: brokers, critical nodes, edge nodes, and intermediate nodes. The system is designed with two types of brokers: remote brokers, which are located in the fog node, and main brokers, which are located in the cloud. The results showed that adopting the fog layer significantly improved the processing and transmission latency. In [20], the author proposed MQTT with a multicast mechanism, DM-MQTT (direct multicast-MQTT), to minimize data transfer delays and network congestion for the massive IoT communications in distributed edge networks. The mechanism relies on the following components: the software-defined networking (SDN) controller and master broker (in the cloud level), SDN switches, a slave broker and analysis server (in the edge level), and IoT devices. The slave broker collects the data of IoT devices and edge information and sends them to the master broker. The master broker collects all edge information from all slave brokers and sends them to the SDN controller. The SDN controller analyzes the edge information, creates a group table, and uses the table to set paths between different edge networks. This architecture does not apply any security solutions, such as authentication and authorization, which is one of the drawbacks.

The authors of [21] present the interworking layer of distributed MQTT brokers (ILDM), which enables multiple MQTT brokers to cooperate with each other. The proposed system relies on MQTT brokers, which are placed at the edges, and ILDM nodes, which are located between the brokers and their clients. The ILDM node can connect with other ILDM nodes; thus, multiple brokers can communicate with each other via ILDM nodes. The system is prone to different kinds of attacks, such as impersonation, MITM, and replay attacks, since no security solutions are presented. The author of [22] proposed the edge–cloud pub/sub broker model, which allows multiple brokers to dynamically coordinate among each other for data delivery to achieve high-scalability and low-latency properties in a large-scale IoT system. The model depends on coordination servers to manage subscription and publication requests among the brokers. The proposed model did not involve any security solutions.

Regarding studies that proposed security solutions, especially authentication for distributed MQTT architecture with multiple brokers, the authors in [23] presented a lightweight and secure authentication scheme for MQTT based on fog computing, which comprises multiple fog gateways as brokers. The scheme used the Elliptic Curve Diffie–Hellman Ephemeral (ECDHE) key exchange algorithm along with the pre-shared key (PSK). However, the paper did not present a broker bridging mechanism, which enables the broker to publish and subscribe the topic to/from other brokers, and the authentication scheme did not take into account brokers' communication. In addition, the results showed that the average transmission and processing times are extremely high, reaching 4.25 s and 2.5 s, respectively. Furthermore, there is a significant increase in the average packet size and the number of packets. The authors of [24] presented a scalable and secure MQTT communication framework for the Industrial IoT based on multi-stage brokers. The single trusted authentication and authorization server (AS) is responsible for clients' authentication and access token generation to define the right of clients to subscribe to or publish a topic. The cryptographic access token is signed through the asymmetric key RSA with a key size of 2048 bits, and the authentication is performed during the TLS handshake protocol. The limitations of this framework are that the central AS could not accommodate the rapid growth of IoT devices, which hinders the scalability and causes a single point of failure or bottleneck, and the high computation cost of RSA and TLS, resulting in heavyweight authentication, which is not suitable for resource-constrained IoT devices. Furthermore, mutual authentication is not provided between the clients and brokers.

To the best of our knowledge, the lightweight mutual authentication scheme and scalable authentication architecture for fog-based MQTT architecture that comprise multiple brokers communicating with each other have not been addressed. Compared to the current works, our proposed architecture implements lightweight mutual authentication between client–broker communication and broker–broker communication using hash function and XOR operation. This reduces the storage overhead, computation costs, and communication overhead of resource-constrained IoT devices. In our proposed architecture, deploying an authentication manager in each broker and clustering the clients and brokers into groups that connect to a single specific broker allows the independency of the authentication management for each group of clients/brokers. This will reduce the possibility of security threats and facilitate security management. Our system enables brokers and their deployed authentication managers to scale-up and scale-out in case clients' requests exceed the processing capacity of the broker. This will avoid network congestion and bottlenecks. Our proposed architecture is designed in a hierarchical structure to enable the MQTT clients obtaining the data from any broker if it is required, although each group of clients is connected to a single local broker in the fog layer.

3. The Proposed Fog-Based MQTT Architecture with an Authentication Scheme

In this section, we present the Mu-TiMB architecture leveraging fog computing to construct a hierarchical MQTT architecture for M2M communication in the IIoT. In addition, we introduce a multi-tier authentication architecture to perform a lightweight mutual authentication scheme using the authentication manager (AM) that is deployed in each broker. In other words, for each group with a direct client–broker or broker–broker connection, there will be an AM to manage authentication procedures independently.

3.1. Multi-Tier MQTT Broker Based on Fog Computing (Mu-TiMB)

An IIoT system based on the traditional MQTT paradigm may lead to a complex, undependable, and non-scalable framework [25] since the current infrastructure is not designed to cope with the rapid growth of data generated in IoT, and this will increase the consumption of resources and reduce the quality of services for the clients. Moreover, industrial applications in the cloud do not meet the requirements of low latency, network traffic reduction, and reliability [26], which are fundamental properties for the industrial

field where the process must be continuously monitored. Therefore, we need to solve such issues by adopting fog computing in smart industries [27].

The idea of Mu-TiMB is based on dividing the clients into groups to interact with the nearest fog broker, called the local broker. As shown in Figure 2, the architecture comprises three layers: the IoT devices layer, where MQTT clients can PUBLISH and SUBSCRIBE messages to/from the local broker; the fog layer, which is composed of two broker tiers, called local brokers and aggregation brokers; and the third layer is the cloud broker, which performs traditional MQTT broker functions with the aggregation brokers. The notation used in Figure 2 is described in Table 2.

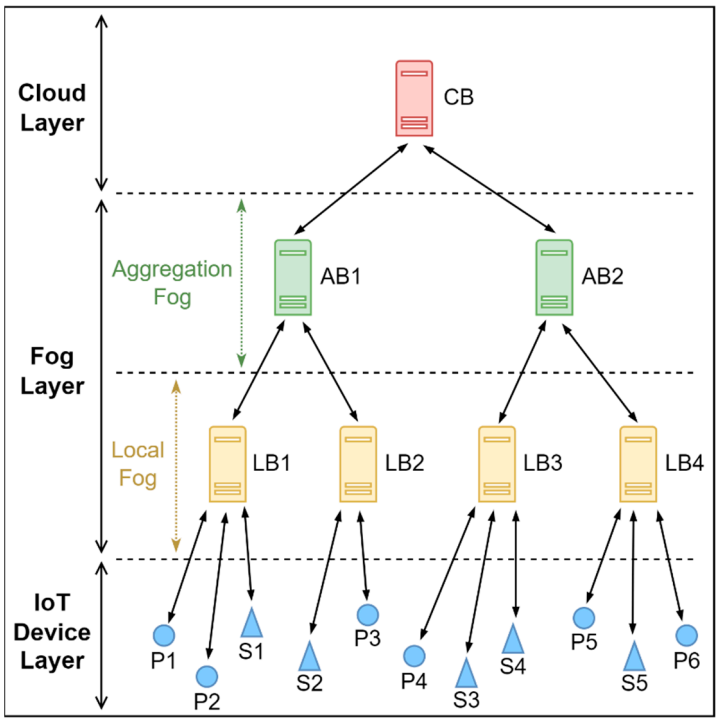


Figure 2. Multi-tier MQTT broker based on fog computing (Mu-TiMB).

Table 2. Notation used in Mu-TiMB architecture.

Symbols	Meaning
P_i	Publisher
S_i	Subscriber
LB_i	Local Broker
AB_i	Aggregation Broker
CB_i	Cloud Broker
T_i	Topic

The messages sent by clients to subscribe to or publish topics are conducted via the local broker. In other words, there is no direct connection between clients and aggregation brokers or between clients and the cloud broker; the topic will be published and subscribed to by the clients only through the local broker which, in turn, either handles and provides the requested topic or obtains the topic from the corresponding upper-layer brokers. As a result, delivering the topic from the publisher to the subscriber will create three communication scenarios: (1) the clients connect to the same local broker; (2) the clients connect to different local brokers, but the same aggregation broker; and (3) the clients connect to different local brokers and different aggregation brokers but share the same cloud broker. Since the clients are clustered and connected directly to different local brokers, the requested topic cannot be found in the local broker but can be found in the

aggregation broker or the cloud broker. The details of the three scenarios are described in Section 3.1.2.

The first scenario is where the publisher and subscriber are connected to the same local broker. In this scenario, the local broker performs the traditional functions of the MQTT broker, which receives the PUBLISH message on a topic from the publisher and delivers the requested topic to the subscriber. In addition to this function, the local broker acts as an MQTT client to publish the topic to the corresponding upper aggregation broker, which in turn, sends it to the cloud broker. This step allows for saving a copy of the published message, including the topic and its data, in all the corresponding upper-layer brokers in case the topic is subscribed to by clients from other brokers. This process is performed each time the published data are received by the local broker.

The second scenario is that the publisher and the subscriber are connected to two different local brokers but have the same aggregation broker. In other words, the subscribed topic is provided by the aggregation broker and not by the local broker that the subscriber is connected to because the requested topic was published to a different local broker, but both local brokers share the same aggregation broker, which stores all topics of the corresponding connected local brokers. In this case, the local broker, where the subscriber is connected, will act as an MQTT client and subscribe to the topic from the corresponding aggregation broker. Since the aggregation broker saves all topics received from the local broker where the publisher is connected, it will forward the topic to the local broker where the subscriber is connected. Then, the local broker sends the requested topic to the subscriber.

The last scenario is where the publisher and the subscriber are connected to two different local brokers and two different aggregation brokers but share the same cloud broker. This scenario is somewhat similar to the second scenario. However, the required topic in this scenario is obtained from the cloud broker; thus, the local broker where the subscriber is connected needs to send the SUBSCRIBE request to the corresponding aggregation broker, which in turn sends the SUBSCRIBE request to the cloud broker. As mentioned before, the local broker where the publisher is connected usually sends the topic to the corresponding upper-layer broker until reaching the cloud broker. Therefore, the cloud broker will forward the topic to the corresponding lower-layer brokers until it is delivered to the subscriber.

3.1.1. Broker Bridging

To implement the proposed architecture, which requires brokers to communicate with each other for publishing and subscribing to topics, a bridging mechanism needs to be presented by MQTT brokers. When a broker is configured as a bridge, in addition to performing the traditional broker functions, it acts as a client by subscribing to some pre-configured topics from another broker (or multiple brokers) and relaying the PUBLISH message targeted to those topics to the selected brokers. As a consequence, the given topics can be shared among more than one broker. This allows clients to subscribe to topics via a broker different from the one that the publisher sends the message to.

Since the client can subscribe to or publish messages into topics handled by brokers different from the broker where the subscriber or publisher is connected, as in the second and third communication scenarios, the client needs to determine the broker that is responsible for handling the topic. This can be achieved by several approaches [24], including defining a new MQTT header field that carries the required information or extending some of the current MQTT header fields. In this study, we are focusing on extending the current MQTT header fields as proposed in [24] by exploiting the topic name string to encode both the identifier of brokers and the topic. The advantage of this approach is that MQTT specification and implementations do not need to be modified. In other words, the topic name will use the symbol "/" to separate the topic and the identifier of brokers that are responsible for receiving and sending the topic.

The topic name (TN) is divided into four parts: the topic (T), the identifier of the local broker (LB), the identifier of the aggregation broker (AB), and the identifier of the

cloud broker (CB). Therefore, the general structure of the MQTT topic name is introduced as follows: TN = CB/AB/LB/T. In the publishing process, LB, AB, and CB indicate to the brokers where the topic will be published and stored. On the other hand, in the subscription process, LB, AB, and CB indicate to brokers where the topic can be subscribed. This structure of TN is used by all clients to send SUBSCRIBE or PUBLISH messages to their corresponding local brokers. Additionally, it is used by brokers when acting as a client to publish and subscribe topics to/from other brokers.

3.1.2. Communication Scenarios

This section explains the processes of the three communication scenarios. We refer to the notations in Figure 2 and Table 2 to present an example for each scenario.

The first scenario is where the subscriber (S1) and publisher (P1) are connected to the same local broker (LB1). In other words, the requested topic name (TN1) is sent to S1 by LB1 where P1 is connected. In addition to the traditional MQTT broker function, LB1 acts as an MQTT client to publish the requested topic to the corresponding aggregation broker (AB1), which in turn publishes the topic to the cloud broker (CB). The scenario is explained in an example below and is shown in Figure 3:

- After successfully connecting to LB1 by P1 and S1, P1 publishes a topic name (TN1) "CB/AB1/LB1/temperature" to LB1.
- To publish the topic TN1 to the corresponding upper-layer brokers, LB1 acts as an MQTT client, connects to AB1, and publishes TN1 to AB1.
- Then, AB1 acts as an MQTT client, connects to CB, and publishes TN1 to CB.
- S1 sends the SUBSCRIBE message asking to subscribe to TN1 "CB/AB1/LB1/temperature" from LB1.
- In this case, LB1, which received the subscribe request, matches with LB1, which is involved in TN1 because TN1 was published by P1 to the same LB1 where S1 is connected. Thus, LB1 publishes TN1 to S1.

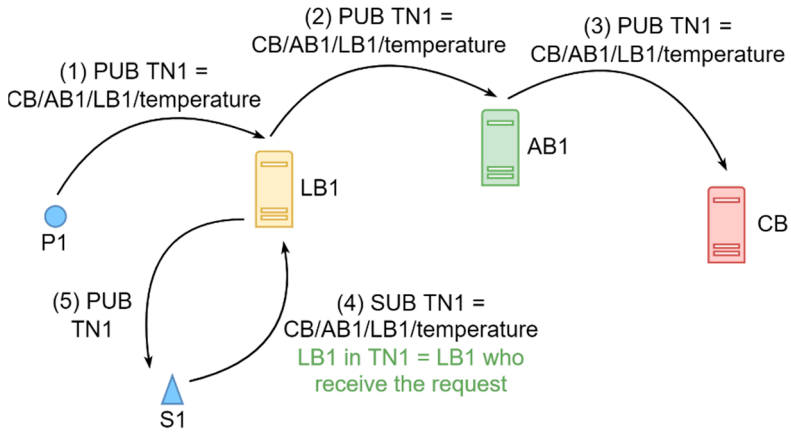


Figure 3. First communication scenario.

In the second scenario, the connected subscriber (S2) and publisher (P1) are located in two different local brokers (LB2 and LB1, respectively) but share the same aggregation broker (AB1). The topic TN1 is obtained from AB1 since TN1 is sent to the corresponding upper-layer brokers (as explained in the first scenario) from LB1 where P1 is connected. Therefore, LB2, where S2 is connected, acts as a client and subscribes to TN1 from AB1. LB2 then receives TN1 and publishes it to S2. Figure 4 introduces the second scenario, which is described as follows:

- After a successful connection from P1 to LB1 and from S2 to LB2, P1 publishes the topic name (TN1) "CB/AB1/LB1/temperature" to LB1.
- To publish the topic TN1 to the corresponding upper-layer brokers, LB1 acts as an MQTT client, connects to AB1, and publishes TN1 to AB1.

- Then, AB1 acts as an MQTT client, connects to CB, and publishes TN1 to CB.
- S2 sends the SUBSCRIBE message to LB2 asking to subscribe to the topic TN1 "CB/AB1/LB1/temperature". Since TN1 is published to LB1 by P1, the requested topic TN1 is not found in LB2 because the identifier of LB1 involved in TN1 is not identical to the identifier of the LB2 that received the SUBSCRIBE request.
- To subscribe to TN1 from AB1, LB2 acts as an MQTT client, connects to AB1, and send the SUBSCRIBE request to AB1.
- In this case, the identifier of AB1 who received the request is identical to the AB1 that is involved in TN1. Thus, AB1 publishes TN1 to LB2 where S2 is connected.
- LB2 publishes TN1 to S2.

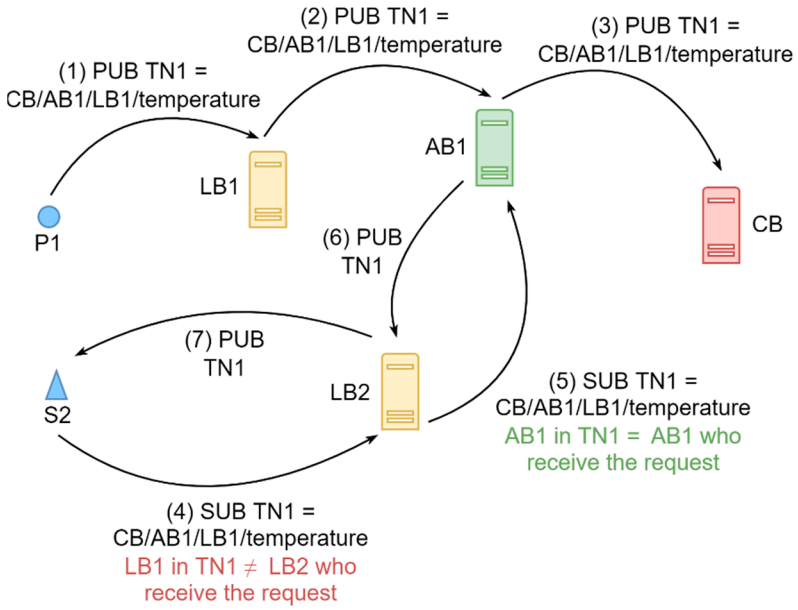


Figure 4. Second communication scenario.

The third scenario is quite similar to the second scenario. However, the cloud broker is in charge of providing the requested topic to the subscriber instead of the aggregation broker because the subscriber and publisher are connected to different local brokers and different aggregation brokers but share the same cloud broker. As explained previously, all topics published by the publisher are sent to the cloud broker via the local broker, then the aggregation broker. In this scenario, we assume that P1 is the publisher and S5 is the subscriber. Thus, the local broker (LB4) to which the subscriber (S5) is connected and from which it requested the topic needs to act as a client and send the SUBSCRIBE message to the corresponding aggregation broker (AB2), which in turn, subscribes to the topic from the cloud broker (CB). After that, the topic is published to the corresponding lower-layer brokers (AB2, then LB4) until it is delivered to S5. Figure 5 depicts the third scenario, which is described as follows:

- After the successful connection from P1 to LB1 and from S5 to LB4, P1 publishes the topic name (TN1) "CB/AB1/LB1/temperature" to LB1.
- To publish the topic TN1 to the corresponding upper-layer brokers, LB1 acts as an MQTT client, connects to AB1, and publishes TN1 to AB1.
- Then, AB1 acts as an MQTT client, connects to CB, and publishes TN1 to CB.
- When S5 sends the SUBSCRIBE message to LB4 asking to subscribe to the topic (TN1) "CB/AB1/LB1/temperature", the requested topic is not found in LB4 because it is published to LB1 by P1. In this case, the identifier of LB1 involved in TN1 is not matched with the identifier of LB4, which received the SUBSCRIBE request.

- Therefore, LB4 acts as an MQTT client, connects to AB2, and sends the SUBSCRIBE request asking to subscribe to TN1 from AB2. In this case, the broker AB2, which received the request, is not identical to AB1, which is involved in TN1.
- As a result, AB2 acts as an MQTT client, connects to CB, and subscribes to the topic TN1 from CB.
- CB publishes the topic TN1 to AB2.
- Then, TN1 is published from AB2 to LB4.
- Finally, LB4 delivers TN1 to S5.

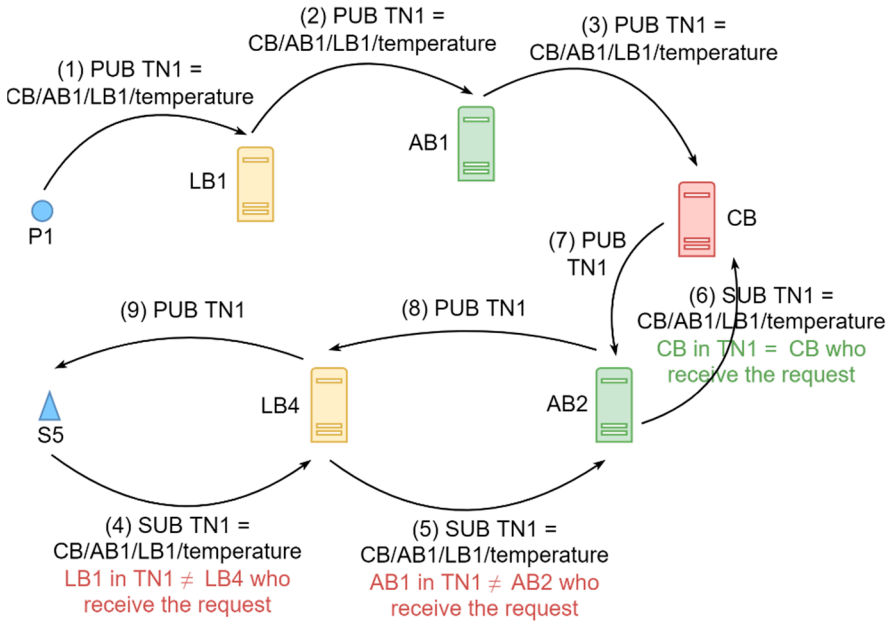


Figure 5. Third communication scenario.

3.2. Authentication System

This paper proposes an authentication system based on multiple authentication managers (AMs), where each AM is deployed in each broker to manage the authentication process independently between clients (which include traditional MQTT clients as well as brokers that act as clients) and their corresponding brokers. In addition, this paper presents a lightweight mutual authentication scheme based on hash function and XOR operation that are performed by an AM. Because IoT devices have resource-constrained capabilities, we tried to propose a lightweight authentication scheme, and we found that hash function and XOR operation is one of the most suitable solutions compared with the symmetric and asymmetric cryptographic algorithms. There are many studies [28–31] that presented lightweight authentication schemes based on hash function and XOR operation, and these schemes have proven their feasibility and efficiency.

The authentication scheme introduced in [28], which is based on hash function and XOR operation, is extended to be compatible with the Mu-TiMB architecture. The scheme considers the hierarchal fog-based MQTT architecture developed with multiple brokers and is implemented by the AM deployed in each broker. The AM is responsible for generating the authentication parameters and verifying the authenticity of clients and brokers. In addition, the obtained parameters are used to conduct mutual authentication between client–broker and broker–broker connections.

The scheme has several advantages. First, the encryption mechanism used for authentication based on hash function and XOR operation has lower storage requirements, communication costs, and computation overhead compared with symmetric and asymmetric encryption. Second, it resists serious authentication attacks such as impersonation, replay, and eavesdropping attacks, and it performs mutual authentication with less over-

head. Third, since the scheme is considered a lightweight cryptographic based on software ciphers [32], it is feasible for it to be implemented in a large-scale and heterogeneous network such as the IoT compared to hardware-based ciphers, and it is more flexible and less costly to deploy in multiple brokers rather than using hardware-based authentication.

3.2.1. Authentication Architecture

This section presents the authentication architecture of Mu-TiMB as well as the overview of the interaction between clients, brokers, and AMs.

As shown in Figure 6, each AM is deployed in each broker, and the AM is in charge of performing an authentication scheme for the associated broker and the group of clients/brokers connected to that broker. For example, the authentication manager of LB2 (AM_{lb2}) manages the authentication for LB2 and its connected clients S2 and P3, while the authentication manager of AB1 (AM_{ab1}) is responsible for authenticating AB1 and its connected local brokers LB1 and LB2, which act as clients. Another example, the authentication manager of CB (AM_{cb}) manages the authentication for CB and its connected aggregation brokers AB1 and AB2. In Mu-TiMB, the authentication process starts once the client/broker sends the normal MQTT CONNECT message to the corresponding broker and completes it successfully after receiving the normal MQTT connection acknowledgment, CONNACK.

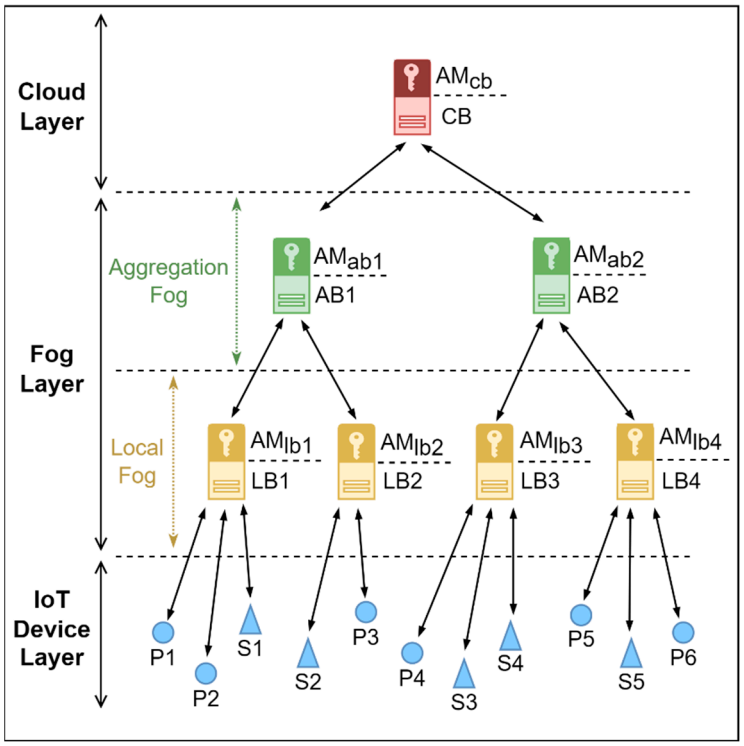


Figure 6. Authentication architecture for Mu-TiMB.

In Mu-TiMB, the AM independently implements the authentication scheme for the associated broker, and their connected group of clients (publishers, subscribers, and brokers that act as clients), as depicted in Figure 7. This creates a multi-tier authentication process, where LB and AB can be authenticated by two different AMs; thus, there are two different authentication parameters, one where they act as a broker and one where they act as a client. For example, LB will be authenticated by AM_{lb} when it receives a connect message from clients and acts as a traditional broker. On the other hand, the same LB will be authenticated by AM_{ab} when it sends a connect message to AB and acts as a client.

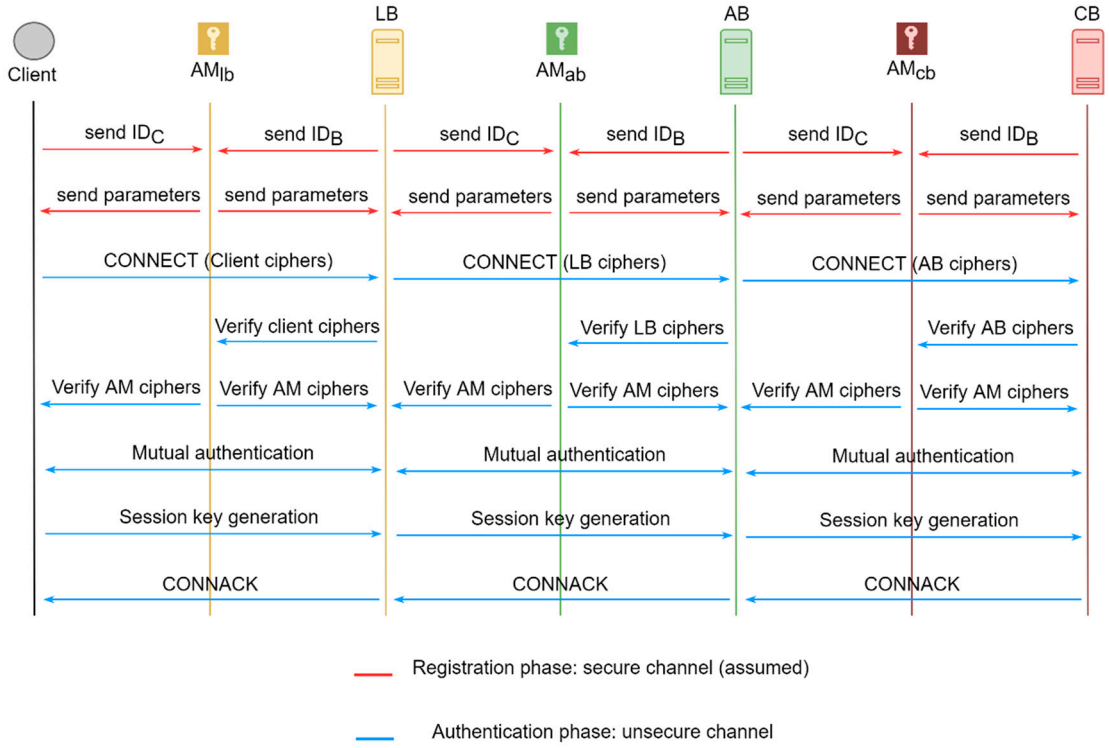


Figure 7. General interaction of the authentication scheme.

3.2.2. Lightweight Authentication Scheme

The authentication scheme comprises two main phases: registration and authentication. The complete steps of the authentication scheme are illustrated in Figure 8, and all notations used in the scheme are listed in Table 3. In the registration phase in which the channel is assumed to be secure, the identity of C (ID_C) and the identity of B (ID_B) are sent to the AM with which B is associated. Then, the AM generates the authentication parameters for both C and B. These parameters contribute to initializing the mutual authentication process between C and B. In the authentication phase, since the connection in the traditional MQTT must be directly established from the client to the broker by sending the client ID, the authentication in our proposed architecture starts by sending the CONNECT message. Thus, C is required to send some authentication parameters to B, including its ID_C, which is wrapped with ID_B in the XOR operation. B, in turn, extracts ID_C and passes the parameters to the AM to authenticate C. The AM then generates and sends other authentication parameters to C and B, which should acknowledge the authenticity of these parameters. After this, a mutual authentication is conducted between C and B. At the end, a session key is generated by C to secure the communication between C and B.

Table 3. Notation used in the authentication scheme.

Symbol	Description
C	Client (publisher, subscriber, or broker acting as a client)
B	Broker
AM	Authentication manager
ID_C	Identity of client
ID_B	Identity of broker

Table 3. *Cont.*

Symbol	Description
AID_C	Alias ID of client
AID_B	Alias ID of broker
SN	Secret number
Na	Nonce
R1, R2, R3, R4	Random number
MK1	Master key 1
MK2	Master key 2
MID_C	Master identity of client
MID_B	Master identity of broker
SK_{AM_C}	Secret key between AM and C
SK_{AM_B}	Secret key between AM and B
$K\text{-}S_{C\_B}$	Session key between C and B
H(.)	One-way hash function
$\oplus$	XOR operator
| |	Concatenation
Ci, Verifier	Ciphers

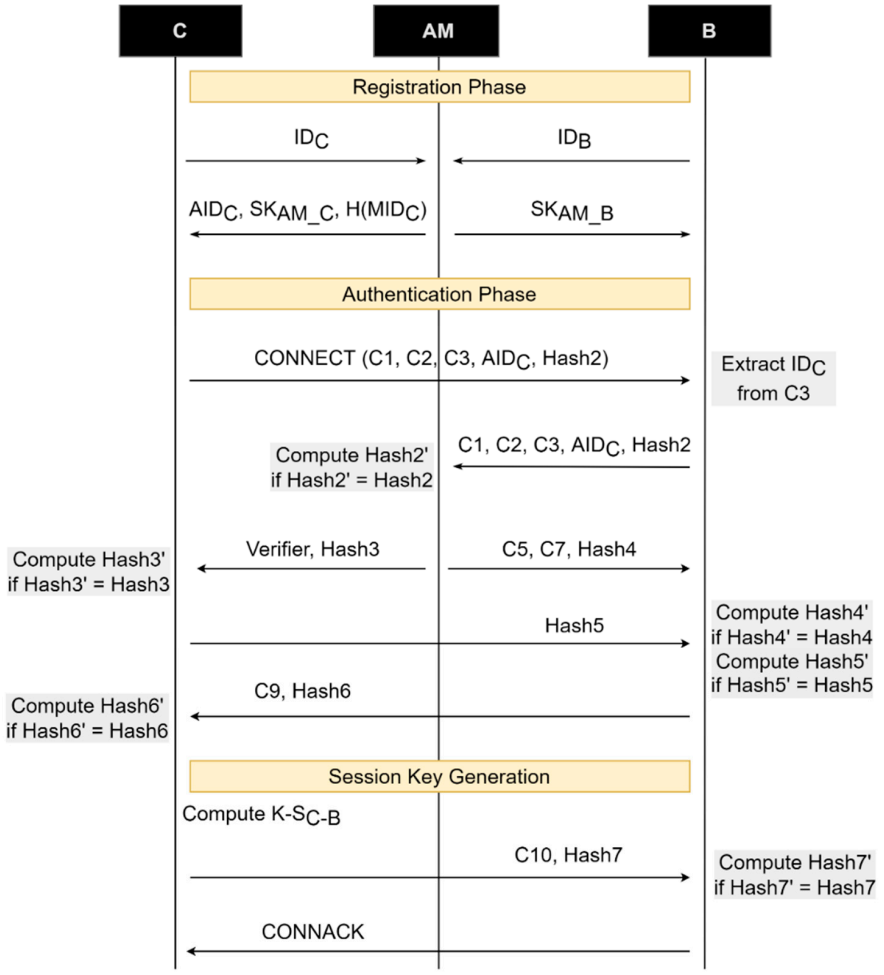


Figure 8. Authentication scheme's steps.

- Registration Phase

In the registration phase, C and the corresponding B share their IDs with the AM. After receiving both IDs, the AM generates and sends authentication parameters for C and B independently. It is assumed that all messages in the registration phase are exchanged through a secure channel.

Upon receiving ID_C and ID_B, the AM creates the following parameters for both C and B: alias identities (AID_C and AID_B) using SN, master identities (MID_C and MID_B) using MK1, and secret keys (SK_{AM_C} and SK_{AM_B}) using MK2. At the end, the AM computes the message digest of the master identity of client H (MID_C), then sends AID_C, SK_{AM_C}, and H (MID_C) to C and SK_{AM_B} to B.

$$AID_C = H\ (ID_C \oplus SN) \tag{1}$$

$$MID_C = AID_C \oplus MK1 \tag{2}$$

$$SK_{AM\_C} = MID_C \oplus MK2 \tag{3}$$

$$Hash1 = H\ (MID_C) \tag{4}$$

$$AID_B = H\ (ID_B \oplus SN) \tag{5}$$

$$MID_B = AID_B \oplus MK1 \tag{6}$$

$$SK_{AM\_B} = MID_B \oplus MK2 \tag{7}$$

- Authentication Phase

To begin the authentication process, C generates a random number (R1), then computes and sends the ciphers C1, C2, and C3 as well as AID_C and Hash2 to B to establish the connection. The ciphers and Hash2 are calculated according to the following equations:

$$C1 = SK_{AM\_C} \oplus R1 \tag{8}$$

$$C2 = H\ (MID_C) \oplus ID_C \oplus R1 \tag{9}$$

$$C3 = ID_C \oplus ID_B \tag{10}$$

$$Hash2 = H\ (AID_C \,||\, R1 \,||\, C1 \,||\, ID_C \,||\, ID_B) \tag{11}$$

Upon receiving C1, C2, C3, AID_C, and Hash2, B extracts ID_C from the cipher C3 using the following equation:

$$ID_C = C_3 \oplus ID_B \tag{12}$$

Then, B passes C1, C2, C3, AID_C, and Hash2 to the AM, which in turn, computes Hash2′ using the following equations and compares it with the received Hash2. If both hash digests are identical, C is authenticated; otherwise, the authentication process is terminated.

$$MID_C = AID_C \oplus MK1 \tag{13}$$

$$SK_{AM\_C} = MID_C \oplus MK2 \tag{14}$$

$$R1 = C1 \oplus SK_{AM\_C} \tag{15}$$

$$ID_C = H\ (MID_C) \oplus C2 \oplus R1 \tag{16}$$

$$ID_B = C3 \oplus ID_C \tag{17}$$

$$Hash2' = H\ (AID_C \,||\, R1 \,||\, C1 \,||\, ID_C \,||\, ID_B) \tag{18}$$

After the successful matching of Hash2 and Hash2′, C needs to prove the authenticity of the AM. Therefore, the AM generates nonce (Na), then computes and sends the cipher Verifier and Hash3 to C according to the following equations:

$$R2 = H\ (R1\ ||\ H\ (MID_C)) \tag{19}$$

$$Verifier = R2 \oplus Na \tag{20}$$

$$Hash3 = H\ (R2\ ||\ Verifier) \tag{21}$$

C, in turn, calculates Hash3′, compares it with the received Hash3, and extracts Na using the equations presented below. If both hashes (Hash3 and Hash3′) are matched, the AM is verified by C.

$$R2 = H\ (R1\ ||\ H\ (MID_C)) \tag{22}$$

$$Hash3' = H\ (R2\ ||\ Verifier) \tag{23}$$

$$Na = Verifier \oplus R2 \tag{24}$$

To start the mutual authentication between C and B, the AM computes Hash4 using the secret key of B that was previously created in the registration phase (SK_{AM_B}) and the following ciphers. Then, it sends C5, C7, and Hash4 to B.

$$C4 = H\ (SK_{AM\_C}\ ||\ R2) \tag{25}$$

$$C5 = C4 \oplus ID_B \tag{26}$$

$$C6 = H\ (ID_B\ ||\ SK_{AM\_B}) \tag{27}$$

$$C7 = C6 \oplus Na \tag{28}$$

$$Hash4 = H\ (C5\ ||\ C7\ ||\ ID_B\ ||\ SK_{AM\_B}) \tag{29}$$

At the same time, C computes Hash5 using ID_C, ID_B, and Na and sends it to B.

$$Hash5 = H\ (ID_C\ ||\ ID_B\ ||\ Na) \tag{30}$$

Upon receiving C5, C7, and Hash4, B computes Hash4′ and compares it with the received Hash4. If both hashes are identical, the AM is authenticated by B. After that, B computes Hash5′ according to the following equations and compares it with the Hash5 received from C.

$$C6 = H\ (ID_B\ ||\ SK_{AM\_B}) \tag{31}$$

$$Na = C6 \oplus C7 \tag{32}$$

$$Hash5' = H\ (ID_C\ ||\ ID_B\ ||\ Na) \tag{33}$$

If both hashes (Hash5 and Hash5′) are matched, B authenticates C. In the next step, B generates a random number (R3) to compute the cipher C8 and Hash6, as shown in the following equations, and sends them to C.

$$C4 = C5 \oplus ID_B \tag{34}$$

$$C8 = H\ (C4\ ||\ ID_B) \tag{35}$$

$$C9 = C8 \oplus R3 \tag{36}$$

$$Hash6 = H\ (C8\ ||\ C9\ ||\ R3) \tag{37}$$

In turn, C computes Hash6′ using the following equations and compares it with the received Hash6.

$$C8 = H\ (H\ (SK_{AM\_C}\ ||\ R2)\ ||\ ID_B) \tag{38}$$

$$R3 = C8 \oplus C9 \tag{39}$$

$$\text{Hash6}' = \text{H}(\text{C8} \,||\, \text{C9} \,||\, \text{R3}) \tag{40}$$

If Hash6′ matches Hash6, C authenticates B, and thus, mutual authentication is achieved successfully.

- Session key generation

After the authentication phase is terminated successfully, C creates the session key ($\text{K-S}_{\text{C\_B}}$) using R3 and the random number (R4), which is generated by C. In addition, C creates and sends cipher C10 and Hash7 to B, as shown in the following equations:

$$\text{K-S}_{\text{C\_B}} = \text{H}(\text{R3} \,||\, \text{R4}) \tag{41}$$

$$\text{C10} = \text{C8} \oplus \text{R4} \tag{42}$$

$$\text{Hash7} = \text{H}(\text{K-S}_{\text{C\_B}} \,||\, \text{C10}) \tag{43}$$

Upon receiving C10 and Hash7 from C, B computes the session key ($\text{K-S}_{\text{C\_B}}$) using R3 and R4, which is extracted from C10. In the next step, B computes and compares Hash7′ with Hash7, which is received from C.

$$\text{R4} = \text{C8} \oplus \text{C10} \tag{44}$$

$$\text{K-S}_{\text{C\_B}} = \text{H}(\text{R3} \,||\, \text{R4}) \tag{45}$$

$$\text{Hash7}' = \text{H}(\text{K-S}_{\text{C\_B}} \,||\, \text{C10}) \tag{46}$$

If Hash7 matched with Hash7′, B verifies that the session key ($\text{K-S}_{\text{C\_B}}$) has been generated and sent from C. The session key aims to secure the communication between C and B.

4. Analysis and Results

4.1. Informal Security Analysis

In this section, we present an informal security analysis to prove that the proposed authentication scheme is able to withstand various malicious attacks and provide several security features.

4.1.1. Resistance to Impersonation Attacks

Impersonation attacks are a type of forgery where an adversary pretends to be or tries to impersonate a legitimate entity by computing legitimate messages leading to the exposure of an entity's identity. In the proposed scheme and during the registration phase, the identities of C and B are shared to the AM in a secure channel; moreover, the AM generates alias identities (AID) for C and B by applying XOR operation and hash function on the real identity and the secret number (SN), which is only known to the AM. Since the hash function is an irreversible operation, the adversary is unable to extract the real identity or the secret number (SN) from the alias identity. Let us assume that the adversary obtained the alias identity of C (AID_{C}) and communicates with B. In this case, the adversary cannot generate the ciphers C1, C2, C3, and Hash2 correctly because it does not know the values used to generate these ciphers, such as $\text{SK}_{\text{AM\_C}}$ and H (MID_{C}). Figure 9 explains this scenario of the attack.

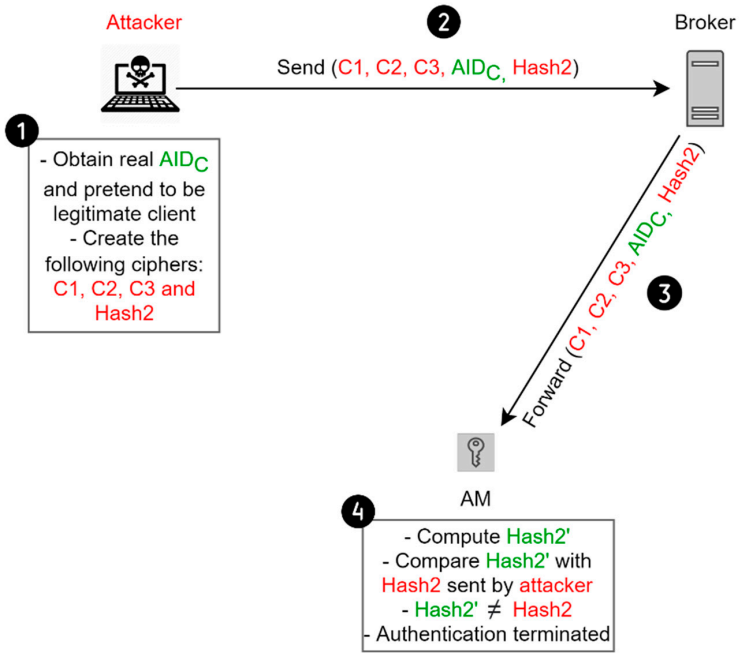


Figure 9. Impersonation attack.

4.1.2. Resistance to Replay Attacks

In the proposed scheme, a replay attack is prevented due to the random numbers involved in or entered into the formation of every transmitted message during the authentication and session key phases, where random numbers cannot be extracted by the adversary. For example, let us assume that the adversary captures and replays the message containing the ciphers C1, C2, C3, AID_C, and Hash2 to B. Upon responding to this message by the AM, the adversary will receive the Verifier, which is generated from the random numbers R2 and Na. In this case, the adversary is unable to extract Na because it cannot obtain R2. Furthermore, R2 cannot be computed because it is generated from H (MID_C), which is unknown to the adversary. Therefore, the adversary is unable to compute Hash3 and Verifier and cannot proceed to start the mutual authentication with B. This scenario of the attack is depicted in Figure 10.

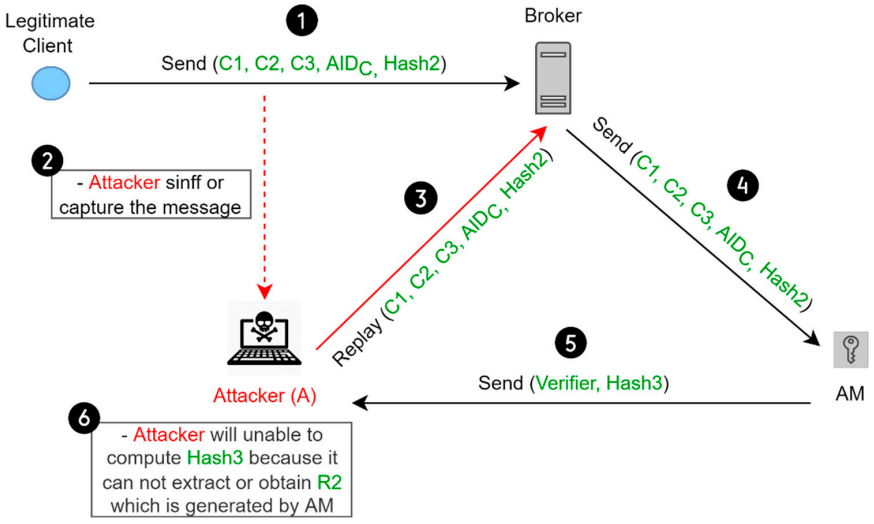


Figure 10. Replay attack.

4.1.3. Resistance to Eavesdropping Attacks

This attack allows the adversary to intercept the communication between two entities and possibly manipulate the messages exchanged between these entities who believe that they are directly communicating with each other, as in a man-in-the-middle attack, which is considered a type of eavesdropping attack. In the presented authentication scheme, an eavesdropping attack can be resisted, as there is no sensitive or valuable information transmitted during the authentication phase, which is conducted on an unsecure channel. This is because the messages are not exchanged in plain text and the parameters involved in these messages cannot be extracted. For instance, as shown in Figure 11, let us assume that the adversary obtains the C1, C2, and C3. The parameters such as the secret key SK_{AM_C} or H (MID_C), which are used to generate these ciphers, cannot be extracted, as they are merged in an XOR operation with other parameters, resulting in ciphertext. Thus, the adversary cannot interpret the message and its sensitive contents. On the other hand, in cases where the adversary modifies the contents of transmitted messages, such as a digest hash, which is involved in each authentication message, the receiving entity of the message will detect the attack. For example, if the adversary changes the Na that was used to generate Hash5, the digest hash result, Hash5′, computed by B will not match the received Hash5, and thus, the authentication process will be terminated.

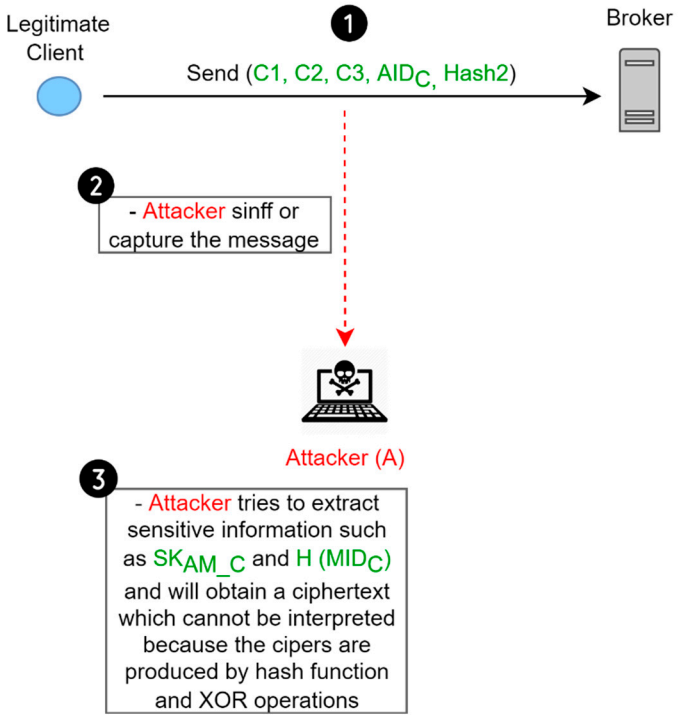


Figure 11. Eavesdropping attack.

4.1.4. Secure Session Key

The session key is a feature to ensure secure communication between two entities, where the key is used for only one session. It is then discarded, and a new key is randomly generated for the next session. In the proposed scheme, the session key $K\text{-}S_{C\_B}$ is established by the C using the digest hash of the concatenation of the random number (R3) previously generated by B with a random number (R4) newly generated by C. The adversary is unable to extract R3 and R4 without obtaining the cipher C8, which is not transmitted or shared at all over the communication channel. Therefore, the session key cannot be computed by the adversary.

4.1.5. Mutual Authentication

In mutual authentication, any two entities must authenticate each other's identities prior to communicating and exchanging messages to avoid adversary breaches. In the proposed scheme, the mutual authentication between B and C is achieved based on correctly computing the received Hash5 and Hash6, respectively. Before the mutual authentication begins, the AM plays a significant role in creating the authentication parameters and verifying the authenticity of C and B to ensure that the messages are not manipulated and to prevent any adversary from becoming a part of the network.

4.2. Performance Results

This section analyzes the performance and overhead of the proposed authentication scheme in terms of computation cost, storage overhead, and communication overhead. The scheme was implemented using the network simulator environment OMNeT++ 5.6.2 that runs on the Windows 10 operating system with an Intel(R) Core (TM) i7-6500U CPU @ 2.50 GHz and 8 GB of RAM. The proposed authentication scheme is based on hash function and XOR operation. The type of hash used in the scheme is SHA-1, which produces a 160-bit hash value known as a message digest.

4.2.1. Computation Cost

The proposed authentication scheme is computationally lightweight and designed for the Industrial IoT environment. The scheme provides high security, utilizing only simple hash and XOR computations, and thus consumes less storage compared with other schemes that are based on symmetric and asymmetric encryption. The scheme uses two operations, namely, the XOR operation and the one-way hash function. To identify the computation cost, we indicate the times that a hash function is conducted using the symbol T_h, and XOR operations are indicated by the symbol T_{xor}.

Considering the authentication scheme shown in Figure 8, the client (publisher or subscriber) performs 11 hash functions and 6 XOR operations, which result in a total computation cost of 11 T_h + 6 T_{xor}. Usually, the computation cost of an XOR operation is negligible and can be discarded, so it is assumed that $T_{xor} \approx 0$. On the other hand, the broker performs seven T_h and five T_{xor}, which results in a total computation cost of 7 T_h + 5 T_{xor}. Therefore, the total computation costs that should be taken into account for both the client and broker after ignoring the cost of T_{xor} are $T_h \approx 11$ and $T_h \approx 7$, respectively. However, in our proposed Mu-TiMB architecture, since fog brokers (FB) (including the local broker (LB) and the aggregation broker (AB)) can act as traditional MQTT brokers and as clients, both LB and AB bear the total computation cost of the client and broker combined, which yields 11 T_h + 7 $T_h \approx 18$ T_h. The computation cost is illustrated in Table 4.

Table 4. Computation cost for the client, LB, AB, and CB.

Entity	Computation Cost
Client	11 T_h
LB	18 T_h
AB	18 T_h
CB	7 T_h

4.2.2. Storage Overhead

The storage overload specifies the total cost of bits required to be stored in each entity. Table 5 shows the storage overhead for the client, LB, AB, and CB. In the proposed scheme, each client (publisher or subscriber) is required to store its actual identity (ID_C), ID_B, AID_C, $H(MID_C)$, and SK_{AM_C}. Since we apply SHA-1 as a type of hash function that produces a hash digest of 160 bits, the size of bits for $|AID_C| = |H(MID_C)| = |SK_{AM\_C}| = 160$ bits. On the other hand, we assume the bits in $|ID_C| = |ID_B| = 16$ bits. Therefore, the total storage cost for the client is $(3 \times 160) + (2 \times 16) = 512$ bits.

Table 5. Storage overhead for the client, LB, AB, and CB.

Entity	Storage Overhead
Client	512 bits
LB	784 bits
AB	784 bits
CB	272 bits

On the other hand, the broker is required to store its actual identity (ID_B) = 16 bits and SK_{AM_B} =160 bits. Since the AM is deployed with B on the same server, the storage cost of the secret AM parameters SN, MK1, and MK2 are incurred by B. Assuming $|SN| = |MK1| = |MK2| = 32$ bits, the total storage cost for B is $16 + (3 \times 32) + 160 = 272$ bits. However, the storage requirements for FB (LB and AB) will bear the cost of both C and B combined. Thus, the total size of bits for FB is 512 + 272 = 784 bits.

4.2.3. Communication Overhead

The communication overhead is measured by the number of packets (total size in bits) transferred between two entities. The size of bits transferred in our authentication scheme is shown in Table 6. In the communication between C (which indicates to the publishers, subscribers, and brokers that act as clients) and B (which includes the AM), C transmits the parameters C1, C2, C3, AIDC, and Hash2. Assume C3, which is a result of an XOR operation between ID_C and ID_B, is 16 bits. The total size of these parameters is $4 \times 160 + 16 = 656$ bits. However, in the communication between C and AM, C sends the parameters C10, Hash5, and Hash7 with a size of $3 \times 160 = 480$ bits. Therefore, the total bit size of the client cost is 656 + 480 = 1136 bits.

Table 6. Communication overhead.

Entity	Storage Overhead
Client → LB + AM	1136 bits
LB +AM → Client	640 bits
LB → AB + AM	1136 bits
AB + AM → LB	640 bits
AB → CB + AM	1136 bits
CB + AM → LB	640 bits

On the other hand, the communication cost of B, including the AM, is less overhead than C. B must send 320 bits to C, which is the total size of C9 (160 bits) and Hash6 (160 bits). On the other hand, its AM sends Verifier and Hash3 to C, which cost 160 bits each. Therefore, the total size of bits is $4 \times 160 = 640$ bits. In our scheme, no communication overhead occurs between B and AM because both entities are deployed in the same server. Since FB (LB and AB) can act as a traditional broker and as a client, it bears the communication overhead of C and B combined; thus, the total bit size cost is 1136 + 640 = 1776 bits.

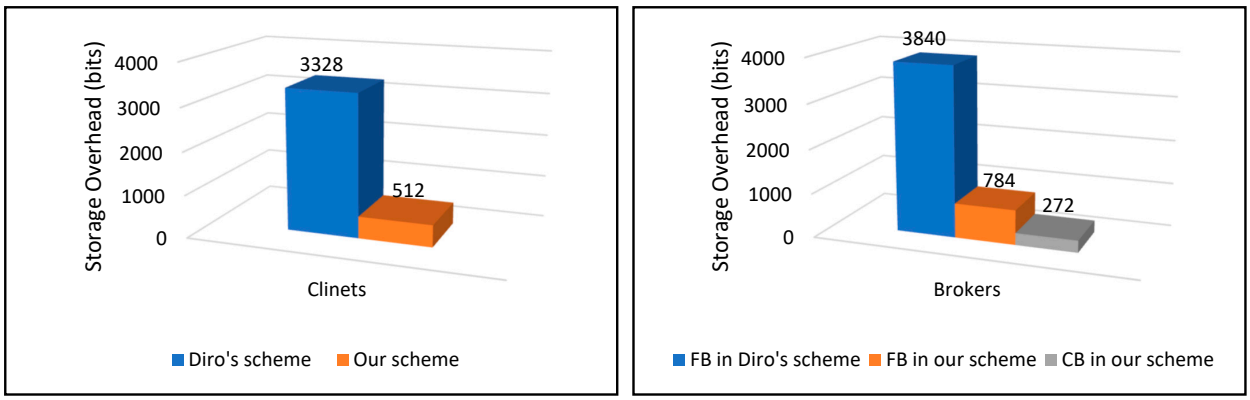
5. Discussion

Our scheme is compared in terms of storage overhead and communication overhead with the authentication scheme of Diro A. et al. [15], which is treated as a benchmark scheme, and compared in term of communication overhead with the scheme introduced in [23] by Amanlou S. et al.

The authors of [15] presented a lightweight security solution including authentication using elliptic curve cryptography (ECC), hashing, and XOR operation for the publish–subscribe protocol based on fog computing. However, broker bridging, which allows a broker to act as a client and communicate with other brokers, was not addressed in Diro's scheme. We achieved a secure authentication level, including mutual authentication and

resistance to attacks such as eavesdropping, impersonation, and replay attacks, and our scheme outperforms in terms of storage requirements and communication costs.

In Diro's scheme, the results showed that the total storage requirements cost 7168 bits: 3328 bits for clients and 3840 bits for the fog broker (FB). The FB in Diro's scheme is equivalent to the cloud broker (CB) in our scheme because both brokers act only as a broker. The storage overhead in our scheme incurs 512 bits for clients and 272 bits for CB for a total of 784 bits. In other words, our scheme reduces the storage overhead to 89%. However, the functions of the FB in Diro's scheme differ from the FB in our scheme, which includes local and aggregation brokers, which act as clients and brokers. Although the FB in our scheme (local and aggregation brokers) incurs the storage overhead of the broker and client combined, they are almost five times lower than the FB in Diro's scheme. The storage requirements for the local broker and the aggregation broker cost 512 + 272 = 784 bits each. Figure 12 shows the storage overhead of clients and brokers for Diro's scheme and our scheme.

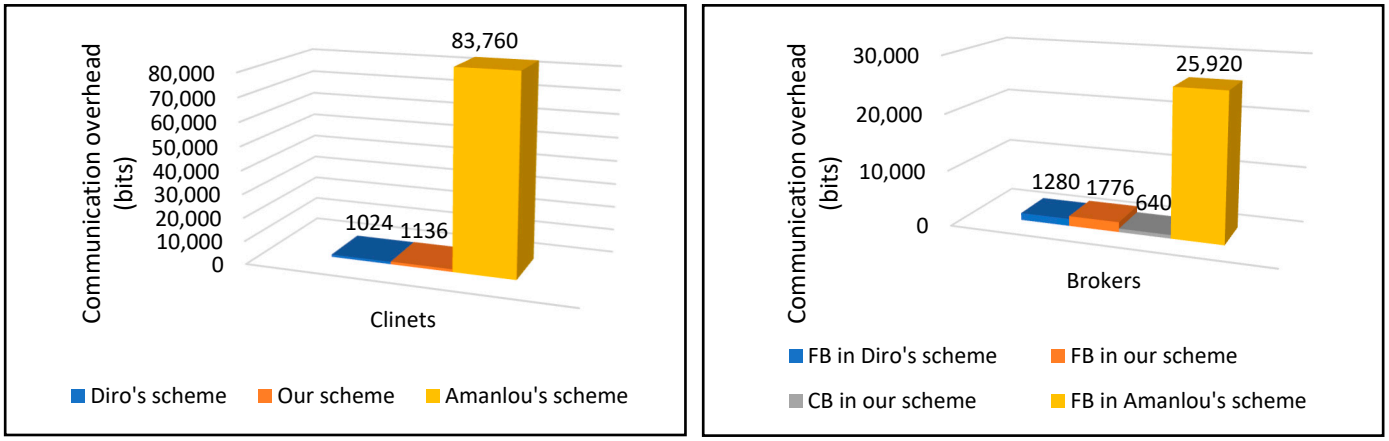


(**Clients**) (**Brokers**)

Figure 12. Storage overhead for clients and brokers in Diro's scheme and our scheme.

On the other hand, the communication overhead of a client in our scheme costs 1136 bits, which is slightly greater than the client in Diro's scheme, which costs 1024 bits. Moreover, our scheme reduces the communication overhead for CB to the half with 640 bits. Taking into the consideration that the FB in Diro's scheme is equivalent to the CB in our scheme, since both brokers perform only the traditional broker functions and do not act as a client, the communication overhead of our scheme costs 1776 bits, while in Diro's scheme it costs 2304 bits. This means our scheme decreases communication overhead almost to 23%. However, there is a noticeable contrast between the FB in our scheme and Diro's scheme. The communication overhead for the FB in our scheme costs 1776 bits, whereas it costs 1280 bits in Diro's scheme. The increased number of bits in our scheme is attributable to the FB playing the role of the client and broker together, requiring each FB to have authentication parameters when it acts as a broker and different authentication parameters when it acts as a client, because each FB communicates with two different authentication managers. Figure 13 illustrates the difference in the communication overhead.

The paper of Amanlou [23] proposed a lightweight and secure authentication scheme between brokers and IoT devices for a publish–subscribe protocol (MQTT) in a distributed fog computing architecture using the Elliptic Curve Diffie–Hellman Ephemeral (ECDHE) key exchange algorithm along with the pre-shared key (PSK). Based on the proposed design of the distributed MQTT architecture, a broker can connect multiple brokers. However, the paper did not handle the authentication mechanism between brokers. In addition, broker communication requires mutual authentication, which is a complex and non-feasible operation with the proposed architecture due to each broker being authenticated more than once at the same time (based on the number of brokers it is connected to). This increases communication overhead, network congestion, and security threats.



(**Clients**) (**Brokers**)

Figure 13. Communication overhead for clients and brokers in Diro's scheme, our scheme, and Amanlou's scheme.

Although Amanlou's scheme uses the pre-shared key authentication instead of the heavyweight certificate-based authentication, the communication overhead is extremely high. Compared with our scheme, there is a vast disparity in the communication overhead. As shown in Figure 13, the communication overhead in Amanlou's scheme require 83,760 bits and 25,920 bits for the client and fog gateway (broker), respectively. This is much greater than our scheme which cost 1136 bits for client, 1776 bits for FB, and 640 bits for CB.

The level of overhead in our scheme is highly acceptable for the FB because it can offer processing and storage resources with much higher capabilities than resource-constrained devices. Furthermore, the condition of the increased communication overhead of our scheme can be justified since it presents several advantages. Beside the lightweight mutual authentication, our proposed architecture provides scalability by scaling the brokers and their associated AMs horizontally and vertically so that the proposed architecture can accommodate the rapid growth of IoT devices and to handle the massive amount of communication, processing and storage requirements. In addition, deploying the authentication manager in each broker facilitates security management and reduces the flow of a large number of authentication requests because each group of Cs is handled independently. This leads to decreased network congestion and bottlenecks, and in cases where a broker and its AM are compromised, the whole system is not compromised.

6. Conclusions and Future Work

This paper proposed a lightweight mutual authentication scheme for distributed MQTT architecture based on fog computing to establish scalable and secure M2M communications for the IIoT. Our proposed architecture is designed in a hierarchical structure with a multi-tier MQTT broker (Mu-TiMB). The authentication scheme is implemented by an authentication manager (AM) that is deployed in each broker to enable the independency of security management for each broker and its group of connected clients/brokers.

Compared to the existing works, our proposed architecture mitigates security performance overhead by implementing a lightweight mutual authentication scheme between client–broker communication and broker–broker communication using hash function and XOR operation. The authentication scheme is characterized by low storage and communication overhead, which have been reduced to 89% and 23%, respectively, compared with the results given in the benchmark. Our system enables the management of the authentication for each broker and its group of connected clients/brokers independently. This decreases the possibility of security threats and facilitates security management. Our proposed architecture can avoid latency, network congestion, and bottlenecks due to clustering the clients

into groups that connect to a single local broker as well as due to the flexibility of scaling the broker and its associated AM horizontally and vertically. In our proposed architecture, the hierarchical structure allows the MQTT client to obtain data from any broker, despite it being connected to a single local broker in the fog layer. One of the authentication scheme disadvantages is the lack of anonymity and untraceability.

This paper can pave the way for the researchers to develop the proposed architecture with the other security properties, such as authorization, availability, and data confidentiality, so that there will be a comprehensive independent security manager for each broker that provides the required security properties for the group of connected clients/brokers. There are two limitations in this paper that could be addressed in future by conducting additional tests for the performance of the authentication scheme, including computation time and transmission time, and by presenting a formal security analysis using security verification tools such as AVISPA and Ban logic.

In the future, we plan to implement an authorization system by deploying an authorization manager for each broker to enforce access control on topics. Moreover, we intend to achieve the availability and avoid the single point of failure for a broker and its associated AM by adding load-balancing solutions. In case any failures occur, another broker and AM can take over the load. In addition, we plan to achieve the confidentiality and integrity of publishing and subscribing to topics using lightweight encryption techniques.

Author Contributions: Conceptualization, H.K. and V.T.; methodology, H.K. and V.T.; software, H.K.; validation, H.K. and V.T.; writing—original draft preparation, H.K.; writing—review and editing, H.K. and V.T.; visualization, H.K. and V.T.; supervision, V.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. State of IoT 2021: Number of Connected IoT Devices Growing 9% to 12.3 B. Available online: https://iot-analytics.com/number-connected-iot-devices/ (accessed on 31 March 2022).
2. Statista. Global IoT and Non-IoT Connections 2010–2025. Available online: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/ (accessed on 31 March 2022).
3. Basir, R.; Qaisar, S.; Ali, M.; Aldwairi, M.; Ashraf, M.I.; Mahmood, A.; Gidlund, M. Fog Computing Enabling Industrial Internet of Things: State-of-the-Art and Research Challenges. *Sensors* **2019**, *19*, 4807. [CrossRef] [PubMed]
4. Sittón-Candanedo, I.; Alonso, R.S.; Rodríguez-González, S.; García Coria, J.A.; De La Prieta, F. Edge Computing Architectures in Industry 4.0: A General Survey and Comparison. In Proceedings of the 14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019), Seville, Spain, 13–15 May 2019; Springer: Cham, Switzerland, 2020; Volume 950, pp. 121–131.
5. Kahvazadeh, S.; Souza, V.B.; Masip-Bruin, X.; Marn-Tordera, E.; Garcia, J.; DIaz, R. Securing Combined Fog-to-Cloud System through SDN Approach. In Proceedings of the CrossCloud 2017 4th Workshop on CrossCloud Infrastructures and Platforms, Colocated with EuroSys 2017, Belgrade, Serbia, 23 April 2017; Association for Computing Machinery Inc.: New York, NY, USA, 2017.
6. Haripriya, A.P.; Kulothungan, K. Secure-MQTT: An Efficient Fuzzy Logic-Based Approach to Detect DoS Attack in MQTT Protocol for Internet of Things. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 90.
7. Calabretta, M.; Pecori, R.; Vecchio, M.; Veltri, L. MQTT-AUTH: A Token-Based Solution to Endow MQTT with Authentication and Authorization Capabilities. *J. Commun. Softw. Syst.* **2018**, *14*, 320–331. [CrossRef]
8. Park, C.S.; Nam, H.M. Security Architecture and Protocols for Secure MQTT-SN. *IEEE Access* **2020**, *8*, 226422–226436. [CrossRef]
9. Bhawiyuga, A.; Data, M.; Warda, A. Architectural Design of Token Based Authentication of MQTT Protocol in Constrained IoT Device. In Proceedings of the 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA), Lombok, Indonesia, 26–27 October 2017; pp. 1–4. [CrossRef]

10. Bali, R.S.; Jaafar, F.; Zavarasky, P. Lightweight Authentication for MQTT to Improve the Security of IoT Communication. In Proceedings of the ACM International Conference Proceeding Series, Kuala Lumpur, Malaysia, 19–21 January 2019; Association for Computing Machinery: New York, NY, USA, 19 January, 2019; pp. 6–12.
11. Rizzardi, A.; Sicari, S.; Miorandi, D.; Coen-Porisini, A. AUPS: An Open Source AUthenticated Publish/Subscribe System for the Internet of Things. *Inf. Syst.* **2016**, *62*, 29–41. [CrossRef]
12. Erlikaya, O.Y.; Dalkiltc, G. Authentication and Authorization Mechanism on Message Queue Telemetry Transport Protocol. In Proceedings of the 2018 3rd International conference on computer science and engineering (UBMK), Sarajevo, Bosnia and Herzegovina, 20–23 September 2018; pp. 145–150. [CrossRef]
13. Rahman, A.; Roy, S.; Kaiser, M.S.; Islam, M.S. A Lightweight Multi-Tier S-MQTT Framework to Secure Communication between Low-End IoT Nodes. In Proceedings of the 2018 5th International Conference on Networking, Systems and Security (NSysS), Dhaka, Bangladesh, 18–20 December 2018; p. 1. [CrossRef]
14. Shilpa, V.; Vidya, A.; Pattar, S. MQTT Based Secure Transport Layer Communication for Mutual Authentication in IoT Network. *Glob. Transit. Proc.* **2022**, *3*, 60–66. [CrossRef]
15. Diro, A.A.; Chilamkurti, N.; Veeraraghavan, P. Elliptic Curve Based Cybersecurity Schemes for Publish-Subscribe Internet of Things. In Proceedings of the International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, online, 9 August 2017; Springer: Cham, Switzerland, 2017; Volume 199, pp. 258–268.
16. Anantharaman, P.; Palani, K.; Smith, S. Scalable Identity and Key Management for Publish-Subscribe Protocols in the Internet-of-Things. In Proceedings of the ACM International Conference Proceeding Series, Bilbao, Spain, 22–25 October 2019; Association for Computing Machinery: New York, NY, USA, 2019.
17. Lohachab, A.; Karambir. ECC Based Inter-Device Authentication and Authorization Scheme Using MQTT for IoT Networks. *J. Inf. Secur. Appl.* **2019**, *46*, 1–12. [CrossRef]
18. Khalid, U.; Asim, M.; Baker, T.; Hung, P.C.K.; Tariq, M.A.; Rafferty, L. A Decentralized Lightweight Blockchain-Based Authentication Mechanism for IoT Systems. *Cluster Comput.* **2020**, *23*, 2067–2087. [CrossRef]
19. Veeramanikandan, M.; Sankaranarayanan, S. Publish/Subscribe Based Multi-Tier Edge Computational Model in Internet of Things for Latency Reduction. *J. Parallel Distrib. Comput.* **2019**, *127*, 18–27. [CrossRef]
20. Park, J.H.; Kim, H.S.; Kim, W.T. DM-MQTT: An Efficient MQTT Based on SDN Multicast for Massive IoT Communications. *Sensors* **2018**, *18*, 3071. [CrossRef] [PubMed]
21. Banno, R.; Sun, J.; Fujita, M.; Takeuchi, S.; Shudo, K. Dissemination of Edge-Heavy Data on Heterogeneous MQTT Brokers. In Proceedings of the 2017 IEEE 6th International Conference on Cloud Networking (CloudNet), Prague, Czech Republic, 25–27 September 2017. [CrossRef]
22. Pham, V.N.; Nguyen, V.D.; Nguyen, T.D.T.; Huh, E.N. Efficient Edge-Cloud Publish/Subscribe Broker Overlay Networks to Support Latency-Sensitive Wide-Scale Iot Applications. *Symmetry* **2020**, *12*, 3. [CrossRef]
23. Amanlou, S.; Hasan, M.K.; Bakar, K.A.A. Lightweight and Secure Authentication Scheme for IoT Network Based on Publish–Subscribe Fog Computing Model. *Comput. Netw.* **2021**, *199*, 108465. [CrossRef]
24. Amoretti, M.; Pecori, R.; Protskaya, Y.; Veltri, L.; Zanichelli, F. A Scalable and Secure Publish/Subscribe-Based Framework for Industrial IoT. *IEEE Trans. Ind. Inform.* **2021**, *17*, 3815–3825. [CrossRef]
25. Ashrafi, T.H.; Hossain, M.A.; Arefin, S.E.; Das, K.D.J.; Chakrabarty, A. IoT Infrastructure: Fog Computing Surpasses Cloud Computing. In *Intelligent Communication and Computational Technologies*; Lecture Notes in Networks and Systems; Springer: Singapore, 2018; Volume 19, pp. 43–55. [CrossRef]
26. Seitz, A.; Buchinger, D.; Bruegge, B. The Conjunction of Fog Computing and the Industrial Internet of Things—An Applied Approach. In Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications Workshops, Athens, Greece, 19–23 March 2018; pp. 812–817. [CrossRef]
27. Bouzarkouna, I.; Sahnoun, M.; Sghaier, N.; Baudry, D.; Gout, C. Challenges Facing the Industrial Implementation of Fog Computing. In Proceedings of the 2018 IEEE 6th international conference on future internet of things and cloud (FiCloud), Barcelona, Spain, 6–8 August 2018; pp. 341–348. [CrossRef]
28. Adeel, A.; Ali, M.; Khan, A.N.; Khalid, T.; Rehman, F.; Jararweh, Y.; Shuja, J. A Multi-attack Resilient Lightweight IoT Authentication Scheme. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e3676. [CrossRef]
29. Esfahani, A.; Mantas, G.; Matischek, R.; Saghezchi, F.B.; Rodriguez, J.; Bicaku, A.; Maksuti, S.; Tauber, M.G.; Schmittner, C.; Bastos, J. A Lightweight Authentication Mechanism for M2M Communications in Industrial IoT Environment. *IEEE Internet Things J.* **2019**, *6*, 288–296. [CrossRef]
30. Amin, R.; Kumar, N.; Biswas, G.P.; Iqbal, R.; Chang, V. A Light Weight Authentication Protocol for IoT-Enabled Devices in Distributed Cloud Computing Environment. *Future Gener. Comput. Systems* **2018**, *78*, 1005–1019. [CrossRef]
31. Alshahrani, M.; Traore, I.; Woungang, I. Design and Implementation of a Lightweight Authentication Framework for the Internet of Things (IoT). In Proceedings of the 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), Granada, Spain, 22–25 October 2019; pp. 185–194. [CrossRef]
32. Roma, C.A.; Tai, C.E.A.; Anwar Hasan, M. Energy Efficiency Analysis of Post-Quantum Cryptographic Algorithms. *IEEE Access* **2021**, *9*, 71295–71317. [CrossRef]