ENS 491-492 – Graduation Project
Final Report

Project Title: Privacy Preserving MQTT Messaging

Group Members:

Bilgesu Çakır 27889

Burcu Tanış 26919

Supervisor(s):

Albert Levi

Date: 04/06/2023

1. EXECUTIVE SUMMARY

MQTT is a communication protocol that is on the application layer which is highly used in IoT communications due to its nature of being a lightweight protocol. With the adopted publisher-subscriber mechanism, a middle node called broker controls the whole communication between clients connected to it, called publishers and subscribers. These parties communicate over the broker using a topic name only without requiring any other information, which is an upside of this protocol. But the nature of the broker creates serious security vulnerabilities as accessing the details of the whole communication between client parties. Although it provides some optional features like using a password and a username, the default protocol is not enough to ensure required security principles such as confidentiality, integrity, authentication, and authorization. To achieve these security principles, a high-level security protocol was developed in this project by modifying chosen modules, a broker and a client module. In the default authenticated client in this project, authenticated encryption is used on both the topic names and payloads of the PUBLISH packets and payload of the 'SUBSCRIBE' packets. The key used for encryption is established in the connection phase by using Ephemeral Diffie Hellman key exchange. Also, the payloads of PUBLISH packets are encrypted using topic based choice tokens. As an optional security scheme, topic hashing publisher and subscriber are implemented. Detailed design schemas and their corresponding implementation results are described in the following sections of this report. In terms of the findings of this project, all the security features implemented except the topic hashing scheme can be used in real applications.

2. PROBLEM STATEMENT

MQTT is an application-layer machine-to-machine communication protocol that is suitable for an Internet of Things (IoT) environment due to its lightweight structure. It uses the publish-subscribe mechanism which allows asynchronous communications between the publisher node and the subscriber node (Calabretta et al., 2018). The connection between these two parties is handled by a third node which is called MQTT Broker.

MQTT by default was not designed to provide enough security requirements It provides optional authentication using a username and password. But usernames and passwords are sent in clear text and this makes the protocol open to eavesdropping attacks (Calabretta et al., 2018). . Besides that MQTT does not provide any mandatory security in terms of confidentiality, integrity, authentication, and authorization (Calabretta et al., 2018). Regarding confidentiality, there is no proper payload encryption used (Hue et al., 202"). This may give access to unauthorized subscribers to reach private sensitive information. Regarding integrity, since any MQTT clients can access the MQTT broker and the topics, any adversary can intercept and modify the messages (Hue et al., 2021). Regarding authentication, MQTT can not prevent spoofing the identity of clients and sending unauthorized information. Since spoofing identity is possible, MQTT is also open to Denial of Service attacks (DoS), and this threatens availability (Ramos et al. 2018). Regarding authorization, even if some sort of client authentication is done, these clients can publish or subscribe to any topic without having appropriate authorization (Ramos et al. 2018).

Some studies have been devised, aiming to improve different security vulnerabilities of the MQTT protocol. Firstly, SSL/TLS protocol is now supported by many MQTT

implementations to provide encryption at the transport level. But this is not enough for network security issues alone. In another study (Singh et al., 2015), a secure MQTT protocol (SMQTT) is developed by using Attribute-Based Encryption (ABE), based both on a Key Policy (KP-ABE) and on a Ciphertext Policy (CP-ABE), and Elliptic Curve Cryptography. The disadvantages of this method are that it creates undesired overhead and it requires the broker to behave as Public Key Generator (Calabretta et al., 2018). In addition, another solution was proposed by Neisse et al. (2014) based on their Model-based Security Toolkit (SecKit). But it requires compliance with the EU security policy rules and it must be integrated in the broker directly (Hue et al., 2021). Moreover in the study of Upadhyay et al. (2016), they use Access Control Lists (ACLs) on the broker to provide security. The drawback of this solution is that it creates overhead because of using different usernames and passwords for different kinds of information (Calabretta et al., 2018). Furthermore, in the study of Niruntasukrat et al. (2016), it is developed using OAuth 1.0a. It provides authorization to access related topics. Clients have to use new signatures for each request and for this an authorization server is required. This mechanism creates overhead on all communication. Also, it does not provide confidentiality (Calabretta et al., 2018).

The motivation of the project is to improve the security and privacy requirements of the MQTT protocol. By default, any client can access the broker without proper authorization, and spoofing the identity of the clients is possible. Unauthorized access to the broker should be prevented by mandatory authentication of the clients. Also, in the current implementation of MQTT unauthorized clients can access private and sensitive information easily since there is no proper payload encryption. Therefore, payloads of packets should be encrypted to provide confidentiality. Moreover, there is no guarantee that there are no modifications in the messages. To prevent message modifications, the integrity of the messages should be achieved. In

summary, this project aims to provide authentication, authorization, confidentiality, and integrity to the default implementation of the MQTT protocol used for IoT devices.

2.1. **Objectives/Tasks**

The objectives/tasks of this project are to provide authentication, authorization, confidentiality, and integrity to the default implementation of the MQTT protocol used for IoT devices.

Providing authentication prevents unauthenticated publishers from publishing messages on any topic and prevents unauthenticated subscribers from subscribing to topics from which they want to get messages. It verifies the identity of communicating parties. Providing authorization prevents unauthorized publishers from publishing messages on unrelated topics and prevents unauthorized subscribers from reading messages that they don't have access to. Providing confidentiality and integrity prevents private data to be reached by unauthorized third parties and prevents private messages to be changed during the communication of the parties.

To achieve these objectives, aMQTT broker, implemented with Python programming language, has been used as the default broker and Eclipse Paho MQTT Python client library has been used as the client library providing the publishers and the subscribers necessary functionality during the communication. First, a long-term design process will be adopted in line with these modules and their specific characteristics. An implementation process will follow the design process and these mentioned modules will be modified to improve the security of the MQTT protocol with the aim of achieving mentioned objectives, according to design choices that have been brought to life during the design process.

2.2. **Realistic Constraints**

This complex problem was subject to some realistic constraints such as time frame, economical, ethical, and sustainability.

In terms of time frame, the design of this project is considered for the fall term of the 2022-2023 academic year, and implementation of this project is considered for the spring term of the 2022-2023 academic year of Sabancı University. This constraint has made the scheduling of the project and following the determined schedule much more crucial.

The ethical constraint of this project is also the motivation behind this project, providing necessary security principles such as authentication, confidentiality, etc. for the parties that use this protocol for communication. Sensitive information could be sent with this protocol by individuals or companies and the absence of the mentioned security principles could lead to ethical and legal issues. As an example, companies may face legal issues due to violating some laws regarding the protection of personal data, etc.

The sustainability constraint of this project concerns whether the result of this project could survive under social, economic, and environmental aspects of the current time and in IoT technology. In order to come up with a solution that aligns with this sustainability constraint, the solution must be applicable to a wide range of devices, these devices might be battery-powered small devices or computers for general use, etc.

In terms of the economical constraints, since no budget is available for this project, resources are considered as the measurement and the determinant. In order to stay within the limits of the economical constraint, open source and free resources like aMQTT broker and Eclipse Paho client library were to be used during the design process and during the implementation process which follows the design part.

Also, there exists specific engineering standards regarding MQTT protocol. The MQTT version that we have used (3.1.1) is an OASIS standard. Specification document provided by OASIS confirms this by the date of October 29, 2014 (“MQTT Version 3.1.1”, 2014). According to IBM, MQTT 3.1.1 also is an ISO standard as ”ISO/IEC PRF 20922:2016” (IBM, 2022). The mentioned ISO standard is explained in the related webpage of ISO (ISO/IEC, 2016).

There are engineering standards for methods we have been using such as Authenticated Encryption (AE), Diffie-Hellman Key Exchange Protocol, etc. For the Diffie-Hellman Key Exchange Protocol, RFC 2631 provides a standard that was published by IETF (Rescorla, 1999). For the methods and algorithms that will be used for the Authenticated Encryption scheme, RFC 5116 provides a standard, published by IETF (McGrew, 2008).

3. METHODOLOGY

As explained before, MQTT offers no authentication, authorization, confidentiality, and integrity in its default configuration. Although it has been used in IoT communication due to its lightweight nature with its default configuration, these weaknesses should be handled to provide secure communication between parties.

Firstly, authentication of publishers and subscribers must be provided. This is done by generating different secure session keys between the broker and publisher and between the broker and subscriber. Key exchange is based on the Ephemeral Diffie Hellman Key Exchange protocol. At this step, the Diffie Hellman public keys are signed by using the X509 certificate on both sides. DH public keys are accepted after the verification process**.** After the connection

setup, a session secret key SK would be shared between the broker and each client (publisher or subscriber).

Secondly, confidentiality, integrity, and authentication of the messages should be achieved. This is achieved by using authenticated encryption for the topic names and payloads of ‘PUBLISH’ messages and topic names of ‘SUBSRCIBE’ messages. For the payloads of ‘PUBLISH’ packet, the HMAC of the payload of ‘PUBLISH’ messages is computed with the shared session key between the broker and each client. This HMAC is appended at the end of the message. Then, this structure is encrypted using the shared session key between the broker and each client again. For the topic names of ‘PUBLISH’ packet, the HMAC of the topic name of ‘PUBLISH’ messages is computed and appended at the end of the topic name. Then, this structure is encrypted using the shared session key between the broker and each client again. For the topic name of ‘SUBSCRIBE’ packet, the HMAC of the topic name of ‘SUBSCRIBE’ messages is computed and appended at the end of the topic name. Then, this structure is encrypted using the shared session key between the broker and each client again.

Thirdly, authorization of the topics is needed. To give access to subscribers to subscribe on a certain topic, topic-based choice tokens are used. Only authorized subscribers with the correct choice token associated with that topic can access the data. The publisher encrypts each message with the choice token related to that topic. A subscriber is able to decrypt only the messages encrypted with the related choice token. This also provides some sort of confidentiality.

In addition to these security methodologies, there is an optional topic hashing scheme for providing privacy. Topic names are changed after each publish iteration based on a hash function on both sides. This hash function uses the previous hash values of that topic names and a random

polynomial for salting purposes to generate the next hash values of the topic names. This iteration is repeated until the counter reaches the predefined maximum value.

4. RESULTS & DISCUSSION

In this part of the report, first the detailed design of the security protocol with communication schemes will take place. Then, the outputs of the implementation process will be included. Thirdly, the libraries and their functions that we used in the implementation process will be referred to. In the fourth part, performance results that have been done so far will take place. Last part contains the difference between the initial objectives and the results.

4.1 Detailed Design

As explained before, MQTT offers no authentication, authorization, confidentiality, integrity, and privacy in its default configuration.

Regarding authentication, the broker will authenticate clients at the beginning of the communication in order to prevent unauthorized access to the broker. The first phase in the communication between a client and broker is the setup phase. As a result of this phase, a common secret key will be generated between the broker and the client by using Ephemeral Diffie Hellman Key Exchange protocol. The flow of the setup phase is shown in Figure 1.

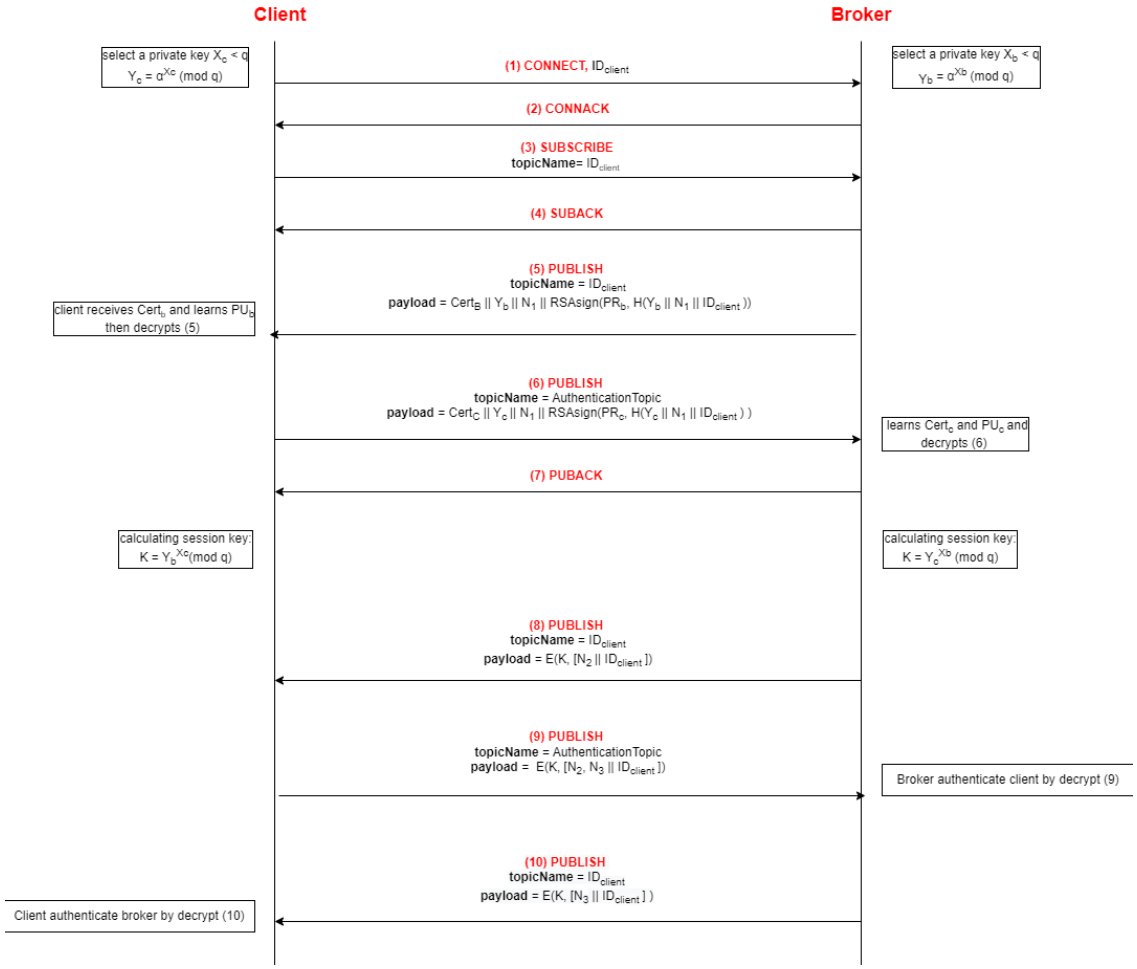

Figure 1

Diffie-Hellman Key Exchange

In this scheme every client and the broker should have X509 public key certificates before starting the communication.

The first packet is a CONNECT packet sent by the client. This packet includes the clientID which uniquely identifies each client. The broker sends a CONNACK packet in response to the CONNECT packet. After the connection established, both sides select a private key (X_c for the client and X_b for the broker) and calculate their public key (Y_c for the client and

Y_b for the broker). The α and q values that are used for this calculation should be known by both sides beforehand the setup phase.

The client private key X_c should be smaller than the q value and the public key is calculated by using $\alpha^{Xc} \, mod \, q$ formula. The private key of the broker X_b should be smaller than the q value and the public key is calculated by using $\alpha^{Xb} \, mod \, q$ formula.

The third packet in Figure 1 is a SUBSCRIBE packet sent by the client to the broker. The Client subscribes to their client ID in order to get messages from the broker which is going to publish in that topic.

The broker gets the SUBSCRIBE packet and sends SUBACK to acknowledge it as the fourth packet.

The fifth packet in Figure 1 is a PUBLISH packet sent by the broker to the client. The broker publishes on a topic same as the clientID. The payload of the packet includes X.509 certificate of the broker, Diffie Hellman public key (Y_b) calculated after the connection established, N_1 and a RSA sign. The broker signs the Diffie Hellman public key, nonce N_1 and client ID by using its own private key corresponding to its X.509 public key certificate.

After the client gets the fifth packet in Figure 1, first it learns the public key of the broker (PU_b) by looking at the X.509 public key certificate ($Cert_b$). Then, it calculates the hash of $Y_b || N_1 || ID_{client}$, decrypts the RSAsign by using PU_b and compares the hash values. If the hash values are not the same, the communication will not continue.

The sixth packet in Figure 1 is a PUBLISH packet sent by the client to the broker. The topic of this packet is AuthenticationTopic. The payload of the packet includes the X.509

certificate of the client ($Cert_c$), Diffie Hellman public key (Y_c) calculated after the connection established, N_1 and a RSA sign. The client signs the Diffie Hellman public key, N_1, and client ID by using its own private key corresponding to its X.509 public key certificate.

After the broker gets the sixth packet in Figure 1, first it learns the public key of the client (PU_c) by looking at the X.509 public key certificate $Cert_c$. Then, it calculates the hash of $Y_c || N_1 || ID_{client}$, decrypts the RSAsign by using PU_c and compares the hash values. If the hash values are not the same, the communication will not continue and the broker disconnects the client. If they are the same, the communication will continue.

After these phases, a common secret key is calculated on both sides. The client uses the formula $Y_b^{X_c}$ mod q and the broker uses the formula $Y_c^{X_b}$ mod q to calculate the common secret key. Now, a common session key K is known by both client and broker. The eighth, ninth and tenth packets in Figure 1 are used for authentication by using the common session key.

The eighth packet in Figure 1 is a PUBLISH packet sent by the broker to the client. The topic of this packet is ID_{client}. The payload of the packet includes N_2 and ID_{client} encrypted using the session key K.

After that, the client decrypts the eighth packet received by using their session secret key K and learns the N_2. Then, the client builds the ninth packet in Figure 1 which is a PUBLISH packet with topic name "AuthenticationTopic". The payload of the packet includes N_2, N_3 and ID_{client} encrypted using the session key K.

By decrypting the ninth packet using common session key K, the broker see the N_2 which was sent in the eighth packet and it authenticates the client. So, the client to broker authentication is completed at this step.

The broker builds the tenth packet in Figure 1 which is a PUBLISH packet with topic name ID_{client} . The payload of the packet includes N_3 and ID_{client} encrypted using the session key K.

The client decrypts the tenth packet using common session key K, and checks the N_3 which was sent in the ninth packet and it authenticates the broker. So, the broker to client authentication is completed at this step.

Regarding authorization, a topic related choice tokens will be used. The messages in the payloads of PUBLISH packets will be encrypted by publishers and decrypted by subscribers using choice tokens. These choice tokens will be located on the broker. The scheme related to getting the choice tokens from the broker is shown in Figure 2.

A publisher before sending any messages for a topic, it must get the topic related choice token from the broker. Also, a subscriber must get the topic related choice token in order to be able to decrypt the messages sent by the publisher for that topic.

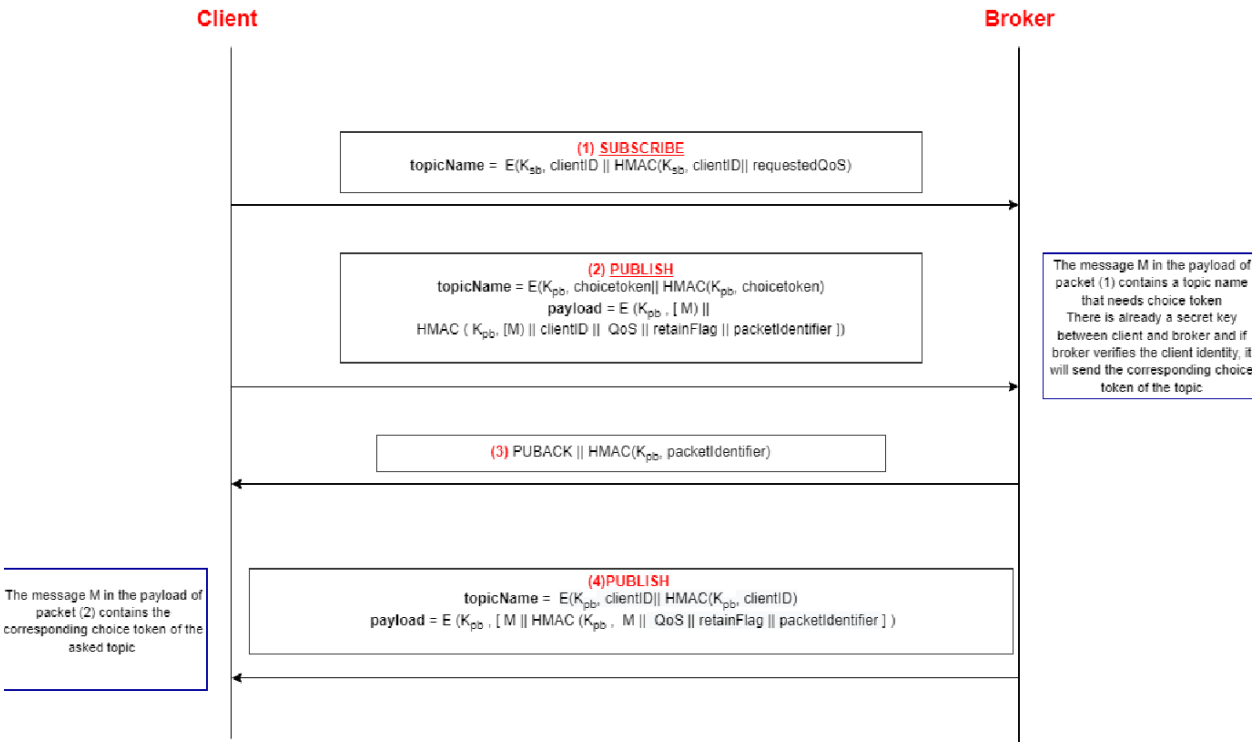

Figure 2

Getting Topic Based Choice Tokens from the Broker

The first packet in Figure 2 is a SUBSCRIBE packet. In this step, the client subscribes to the authenticated encryption version of its client ID.

The second packet in Figure 2 is get choice token PUBLISH packet sent by the client. The topic name of this packet is “choiceToken”. “choiceToken” is the topic name which is used by all clients to get the related choice token of the interested topic names. The topic name “choiceToken” and its HMAC is encrypted using the common session key which was generated at the setup phase. The payload of this packet includes the encrypted version of M (message itself) and the calculated HMAC of the (M || | QoS || retainFlag || packetIdentifier). QoS, retainFlag and packetIdentifier are part of the header fields of the PUBLISH message and they

are added to the HMAC calculation in the payload to make sure there is no modification for them.

The M message in the second packet contains the topic name for which the client wants to get a choice token from the broker. If the broker can decrypt the packet by using the session key between the client and the broker successfully, it compares the HMAC values. If everything is correct, the communication continues. If the HMAC values are not the same, an error message is sent by the broker to the client to inform the client that its subscription is not accepted.

The PUBACK packet, which is the third packet in Figure 2, is sent as a response to the PUBLISH. It includes packetIdentifer of the corresponding PUBLISH packet in its header. In the default design, there is no payload inside PUBACK packets but in our design, the PUBACK has a payload field which includes the MAC of the packetIdentifer. This is done to provide integrity of the acknowledgement.

The fourth packet in Figure 2 is a PUBLISH packet sent by the broker. The topic name of this packet is the authenticated encryption version of the clientID. The payload of this packet includes the encrypted version of M (message itself) and the calculated HMAC of the (M || clientID || QoS || retainFlag || packetIdentifier). The common session key is used as the key for the encryption and HMAC. The M message itself contains the topic name and its corresponding choice token of the asked topic.

Regarding confidentiality, authentication and integrity of the messages the authenticated encryption will be used. Figure 3 shows an example scheme for this communication phase.

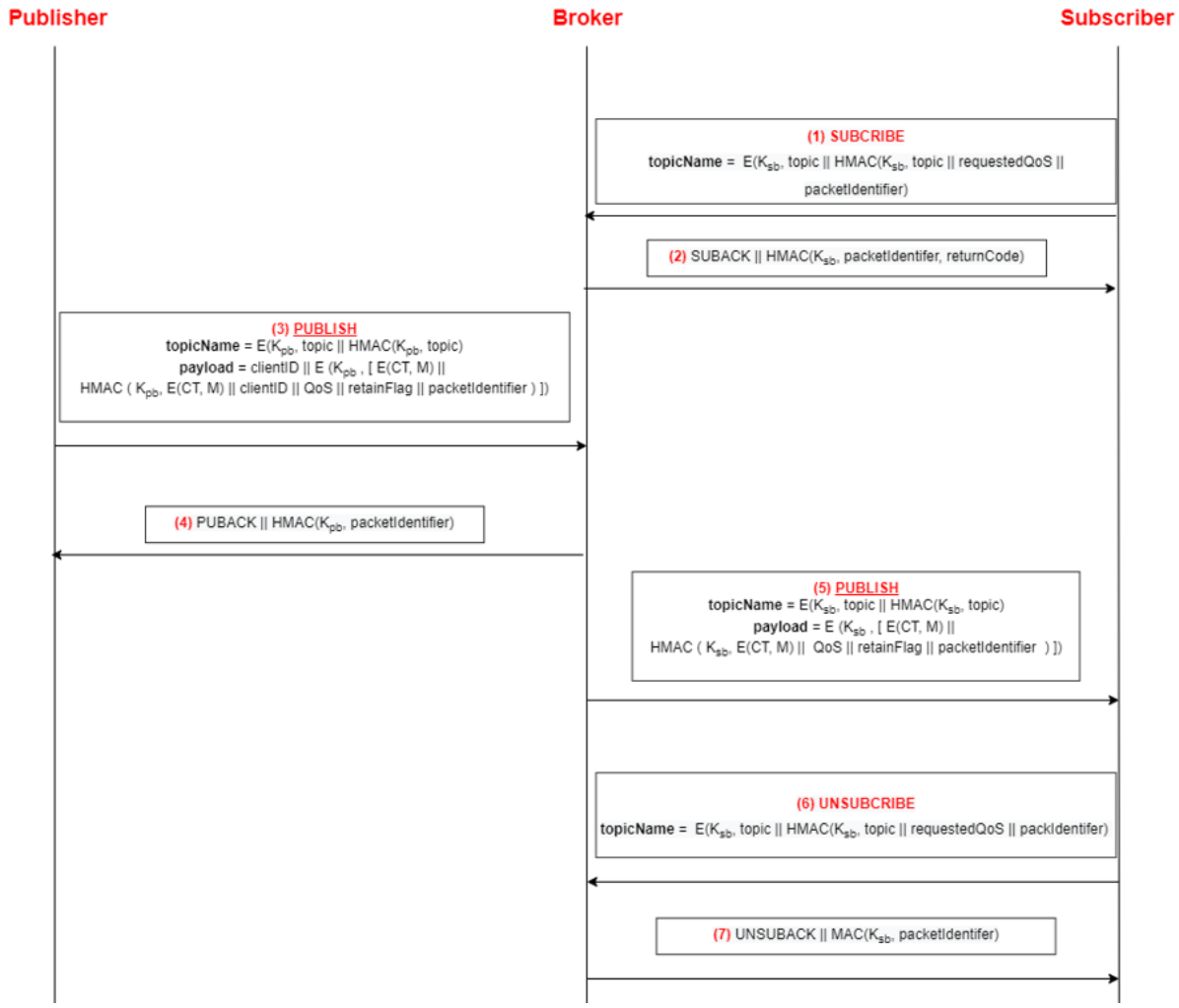

Figure 3

Confidentiality, Message Authentication, and Integrity

The first packet in Figure 3, is the SUBSCRIBE packet sent by the subscriber to the broker. The topic name and the HMAC of the (topic name || requestedQoS || packetIdentifier) of the SUBSCRIBE packets is encrypted by the session key between the broker and the subscriber (K_{sb}) which was generated at the setup phase in order to prevent unauthorized clients from seeing it by intercepting the communication. The HMAC is used to guarantee that there is no modification on the topic name, QoS and packet identifier fields. When the broker gets the

SUBSCRIBE packet, first it tries to decrypt it by using the unique session key between itself and the client subscribing to it. Then, it calculates the HMAC of the (topic name || requestedQoS || packetIdentifier) and compares two HMAC values. If they are the same, it adds this subscription to the subscription list. If they are not the same, the broker informs the client by sending an error message.

The second packet in Figure 3, is the SUBACK packet sent by the broker to the subscriber after it receives the SUBSCRIBE packet. In our design, the SUBACK has a payload field which includes the HMAC of the packetIdentifer and return code. These are added to provide integrity of the acknowledgement.

The third packet of Figure 3 is a PUBLISH packet sent by the publisher. The topic names of the PUBLISH packets are encrypted by the session key between the broker and the publisher (K_{pb}) which was generated at the setup phase in order to prevent unauthorized clients from seeing it by intercepting the communication. The HMAC of the topic name is calculated by using K_{pb} in order to guarantee that there is no modification on the topic name field. Also, authenticated encryption is used for the payload of the PUBLISH packets. The payload includes the M (message itself) which is encrypted by the choice token CT and the calculated HMAC of the (E(CT, M) || QoS || retainFlag || packetIdentifier). These are encrypted using the common session key. QoS, retainFlag and packetIdentifier are part of the header fields of the PUBLISH message and they are added to the HMAC calculation in the payload to make sure there is no modification for them.

The broker gets the PUBLISH packet and decrypts the topic name by using common session key K_{pb}between the broker and the publisher. Then it checks the HMAC of the topic name. If it is correct, it continues to check the payload. The broker decrypts the payload by using

K_{pb} and then it calculates the HMAC and compares it with the received HMAC value. If they are the same, it adds this publish packet to the queue in order to forward it to the related subscribers. If they are not the same, the broker sends an error message to the publisher to inform it that it will not relay this publish message.

The fourth packet in Figure 3, is the PUBACK packet sent by the broker to the publisher after it receives the PUBLISH packet. It includes packetIdentifer of the corresponding PUBLISH packet in its header. In the default design, there is no payload inside PUBACK packets but in our design, the PUBACK has a payload field which includes the MAC of the packetIdentifer. This is done to provide integrity of the acknowledgement.

The fifth packet in Figure 3, is a PUBLISH packet that relays the message sent by the publisher in the third packet. This packet is similar to the third packet in Figure 3 which was sent by the publisher. One difference from it, is the common session key used. Here, common session key is the one used between the subscriber and the broker (K_{sb}).

The sixth packet in Figure 3, is the UNSUBSCRIBE packet sent by the subscriber to the broker. The topic name and the HMAC of the (topic name || requestedQoS || packetIdentifier) of the UNSUBSCRIBE packets is encrypted by the session key between the broker and the subscriber (K_{sb}) . When the broker gets the UNSUBSCRIBE packet, first it tries to decrypt it by using the session key between them. Then, it calculates the HMAC of the (topic name || requestedQoS || packetIdentifier) and compares two HMAC values. If they are the same, it removes the topic name from the subscription list for the corresponding client. If they are not the same, the broker informs the client by sending an error message.

The seventh packet in Figure 3, is the UNSUBACK packet sent by the broker to the subscriber after it receives the SUBSCRIBE packet. In our design, the UNSUBACK has a

payload field which includes the HMAC of the packetIdentifer. This is added to provide integrity of the acknowledgement.

Regarding providing privacy, besides confidentiality, topic name hashing scheme will be used as an optional mechanism. The example scheme is shown in Figure 4, Figure 5 and Figure 6. This scheme is implemented on top of the previous schemes. So, all the topic names and payload are encrypted using the session key between the broker and the client. Also, related HMAC values are calculated as mentioned at the previous schemes.

Subscribers must publish on the “newParticipant” to join the scheme which is shown in the second packet of Figure 4. The third packet is the relay of the second packet sent to the real publisher. Then, the subscriber subscribes to “topicHashing” which is shown as the fourth packet of Figure 4. In the fifth message of Figure 4, the publisher sends a PUBLISH packet on the topic “topicHashing” that will be relayed by the broker to all related subscribers. The payload of this packet contains the seed mapping and a random polynomial for salting. Seed mapping is a set of initial random strings, each one corresponding to a different topic. The seed mapping and random polynomial will be used in the hashing function in the later stages of this scheme. The sixth packet is the relay of the fifth packet.

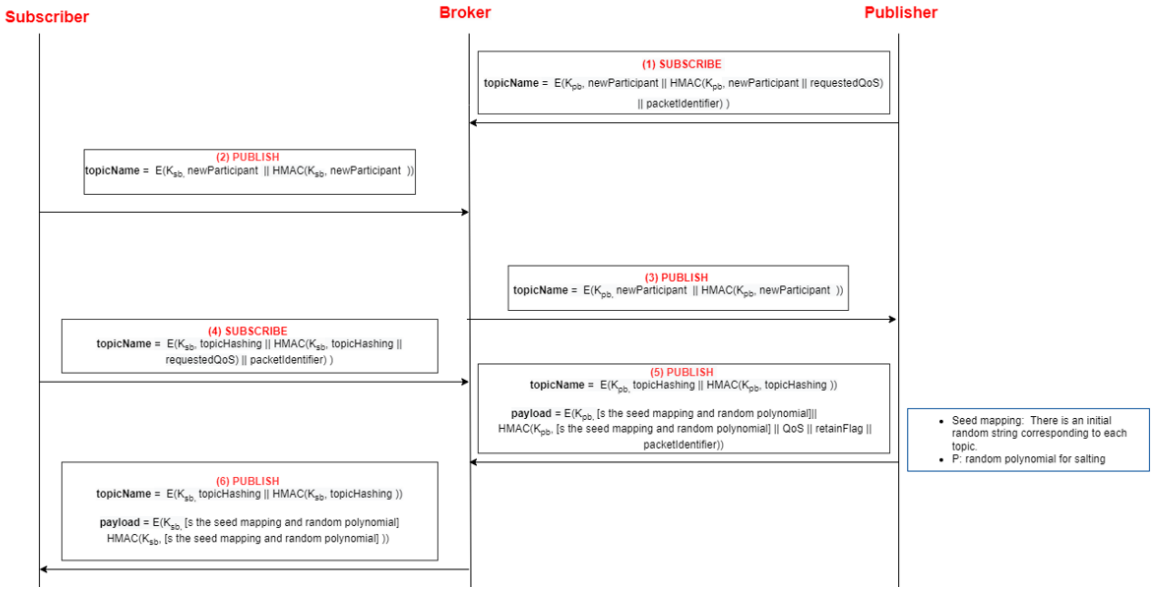

Figure 4

Topic Name Hashing 1

The flow of Figure 4 continues in Figure 5. Figure 5 starts with calculation of the S_t^1 for all topics which are the first hash values of the topic names. They are calculated using the function $S_t^j = F(num(S_t^{j-1}).P(j)) \forall t \in Topics$. F is the predetermined hash function. P is the random polynomial which will be used for salting. "j" can take values between 1 and the seed expiration counter. j value starts from 1. Each time, publisher publishes on one of its topics, j is incremented by 1 and a new hash value for each topic is calculated until j reaches the seed expiration counter.

After the calculation of the S_t^1 in Figure 5, subscriber subscribes to the hash of the interested topic names. In the example scheme, the interested topics are $S_{t1}^1, S_{t3}^1, S_{t4}^1, S_{t6}^1$.

Then the publisher sends a PUBLISH packet in this figure as the eighth packet. In the example scheme, publisher wants to publish on the topic S_{t1}^1 but it also publishes on all the topics

in its publication set. Since the real published topic is S^1_{t1}, all of the messages are encrypted using the choice token of topic S^1_{t1} .

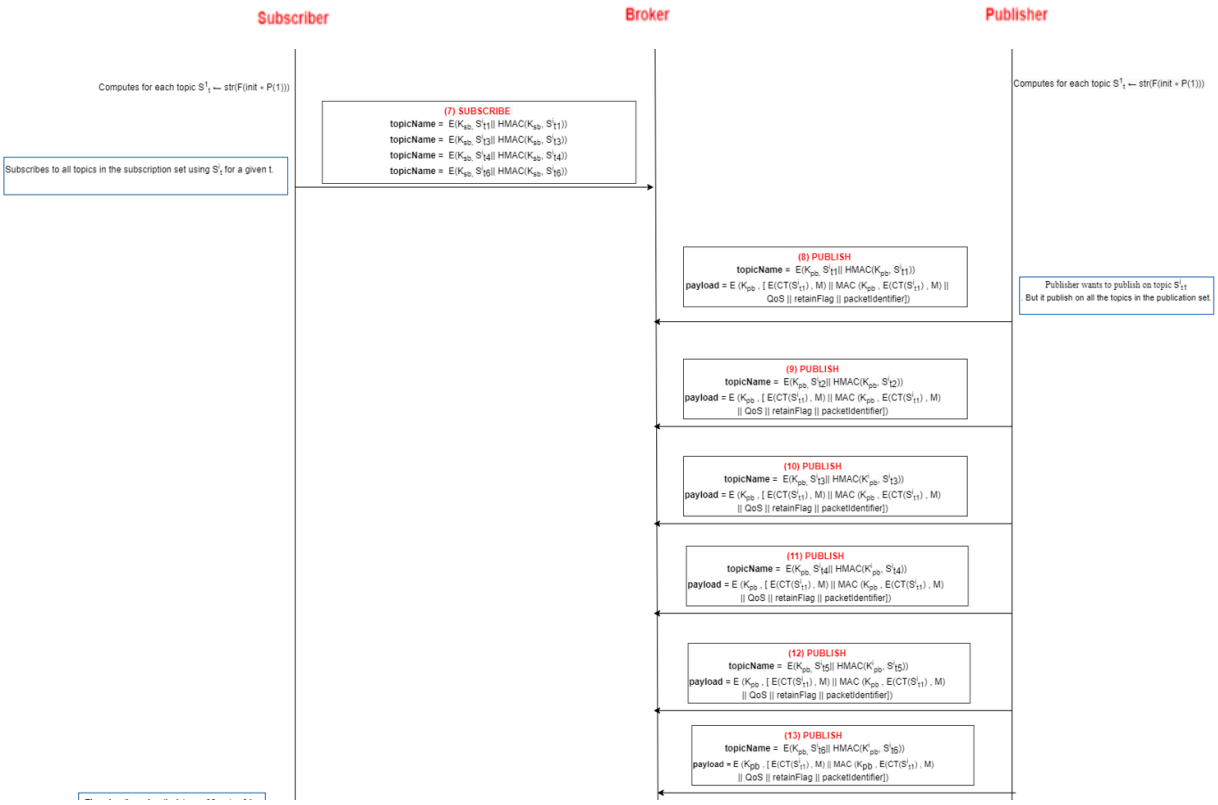

Figure 5

Topic Name Hashing 2

The flow of Figure 5 continues in Figure 6. As can be seen in Figure 5 and 6, the publisher is publishing on the topics S^1_{t1}, S^1_{t2}, S^1_{t3}, S^1_{t4}, S^1_{t5}, S^1_{t6} and all of them are encrypted by using the choice token of S^1_{t1}. The broker relays these messages to related subscribers. In our example scheme our subscriber gets the messages for topics S^1_{t1}, S^1_{t3}, S^1_{t4}, S^1_{t6}.

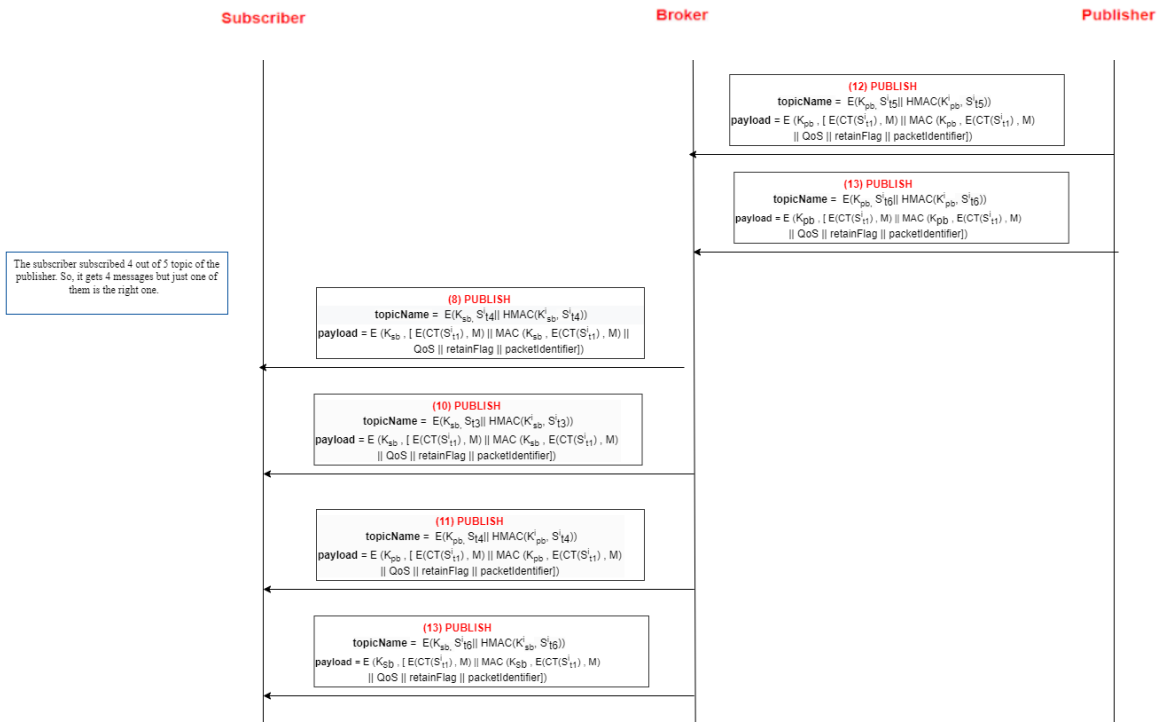

Figure 6

Topic Name Hashing 3

As can be seen in the Figure 6, the subscriber gets 4 messages for topics S_{t1}^{1}, S_{t3}^{1}, S_{t4}^{1}, and S_{t6}^{1}. But just topic S_{t1}^{1} includes the real payload and the other three include random payloads. All of them are encrypted using the choice token of topic S_{t1}^{1} which is the real one.

Client try to decrypt these topics and S_{t3}^{1}, S_{t4}^{1} , and S_{t6}^{1} are discarded due to the wrong post-decryption attempt. S_{t1}^{1} can be decrypted since it is the real one for this iteration.

Then both publisher and the subscribers update the given topic name hashing with $S_t^j = F(num(S_t^{j-1}).P(j)) \; \forall t \in Topics$ and increment the j by one. Also, the publisher

updates the counter. When the counter reaches the predefined maximum value, the publisher sends the new seed mapping for the topics and a new hash session starts.

2.2 Implementation

Implementation part of the Diffie Hellman Key Exchange will be explained with outputs in the following part. Due to the complexity of the outputs, we will run the broker in INFO-logging mode instead of DEBUG-logging mode which we normally use during implementation and testing.

Figure 7

Starting the Broker

Figure 7 contains the output in the broker logs after it was started. At the opening stage, the broker reads its X509 certificate which was created before.

Figure 8

Starting the Client

Figure 8 contains the output in the client logs after it was started. At the opening stage, the client reads its X509 certificate which was created before.

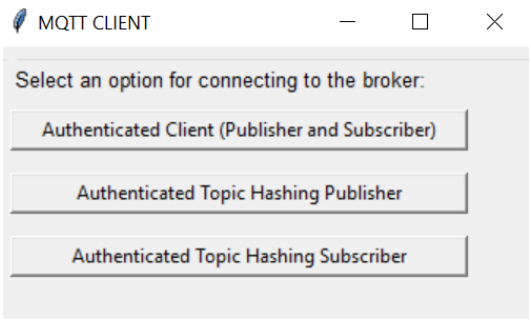

Figure 9

Starting GUI

When the program is executed, first the client should select an option from the GUI. As can be seen in Figure 9, there are 3 options. First option is for default authentication which is explained in Figure 1, Figure 2 and Figure 3. The second and third options are for the topic name hashing scheme which is explained in Figure 4, Figure 5 and Figure 6. First, default authenticated client's console outputs will be mentioned. When the client clicks the first button, this option screen will open another GUI which can be seen in Figure 10.

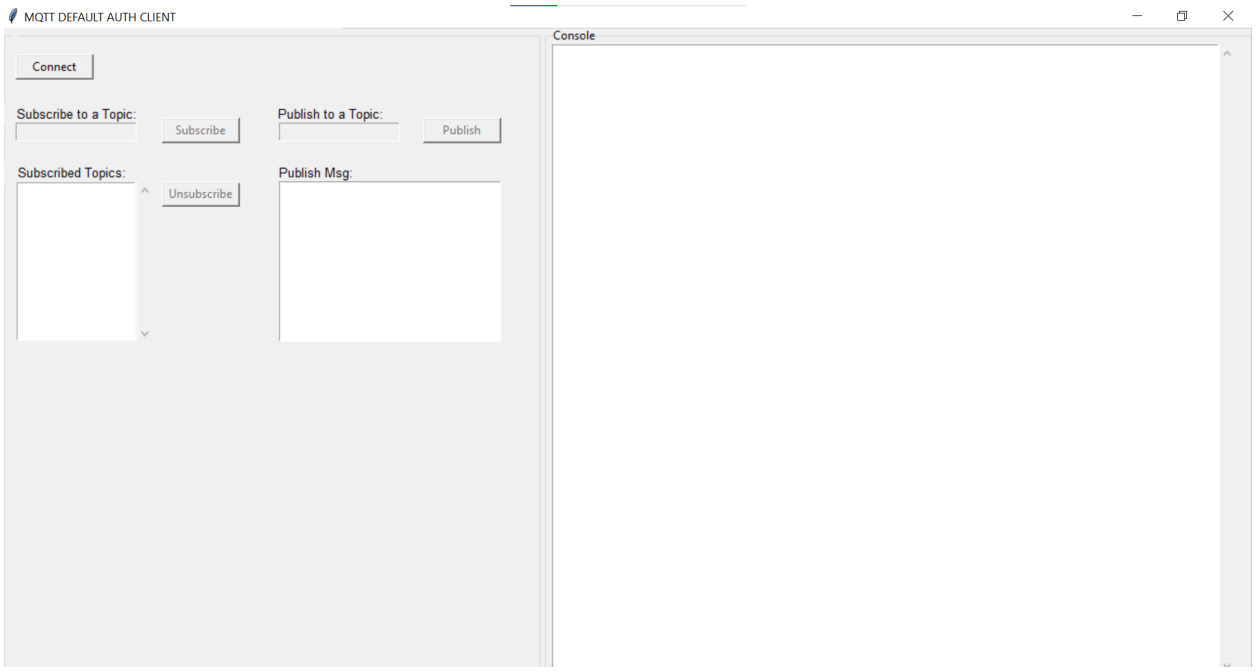

Figure 10

GUI of Default Authenticated Client

Figure 10 contains the graphical user interface of the default-auth client. At the beginning only the connect button is enabled, the other buttons and text boxes are disabled. When the client pushes the connect button Diffie Hellman key exchange is started. When the Diffie Hellman key exchange ends successfully, the connect button is disabled, subscribe and publish buttons and their text entries are enabled.

```
 20:26:46,431   CLIENT ID: 71816975
 20:26:46,440   ---Connection message send to broker (step 1)---
 20:26:46,451   Connection successful (step 2), return code: 0
 20:26:46,548   ----Client subscribed to its client id (step 3 of the DH Key Exchange
)----
 20:26:46,556   Suback received, message id: 1
```

Figure 11

Connection Established and Client Subscribe to its client ID

Figure 11 contains the connection establishment state of the client. Client connects to the broker with its clientID which is described in the step 1 of Figure 1 (DH scheme). After the client receives the connack packet, it subscribes to its client ID which is the step 3 of Figure 1.

```
[2023-05-21 20:26:46,441] :: INFO :: amqtt_folder.broker :: Listener 'default': 1 connections acquired
[2023-05-21 20:26:46,442] :: INFO :: amqtt_folder.broker :: Connection from 127.0.0.1:6770 on listener 'default'
[2023-05-21 20:26:46,450] :: INFO :: amqtt_folder.broker :: Client ID: 71816975 Start messages handling
[2023-05-21 20:26:46,550] :: INFO :: amqtt_folder.broker :: Client ID: 71816975 handling subscription
[2023-05-21 20:26:46,551] :: INFO :: amqtt_folder.broker :: ADD SUBSCRIPTION: ('71816975', 1)
[2023-05-21 20:26:46,553] :: INFO :: amqtt_folder.mqtt.suback :: Signature of the suback packet: None
[2023-05-21 20:26:46,554] :: INFO :: amqtt_folder.mqtt.suback :: ###SUBACK WAS SENT###
[2023-05-21 20:26:46,556] :: INFO :: amqtt_folder.broker :: ----CLIENT WAS SUBSCRIBED TO ITS CLIENT ID(STEP 3 of DH)----
[2023-05-21 20:26:46,557] :: INFO :: amqtt_folder.mqtt.protocol.handler :: ----FUNCTION: PREPARATION OF THE PUBLISH MESSAGE AT STEP 5 OF DH for CLIENT ID: 71816975----
[2023-05-21 20:26:46,558] :: INFO :: amqtt_folder.mqtt.protocol.handler :: TOPIC NAME (client ID):  71816975
[2023-05-21 20:26:46,709] :: INFO :: amqtt_folder.mqtt.protocol.handler :: BROKER DH PUBLIC KEY: b"\xea\xed\xe7'\xd3t\x11\x005>\xa5\xe6\x1c;\xbc(\xa7\xf5u4\xcb\xda\xb8
C\xb1a\xc9z>\xab\x14\xf5\x85\xba\xc2\xc0\xf1e@\xb7\xb7\x87\x8f)\xd0\x8a?D\xd3\xd2\x03\xa4\x80N]\x9e\x8a\xf7x\xc0\xc6f\xd8\xa9S\xcd\xc6\xdd\x08)&'\xcb\xbff\xc5\x01U(\xb
5^{ieQ_\xfe\xabL-\xea[V\xf7v\xd5\xa2\xb7\x0b\x0c_\x98\xe8Wn\x9c\xec\x02[\xfd\x163\xbe\xb4\xa2\xeau\x17~\x13\x12\x85\x01\xc3\x00l\xa5\xb9_.Fd\x16\x9a\xb2\x92\xe9\xa0B\x
95\xcfv\x08\xbc\x97\xfcr\xf8\xa2\xe5\xb7\xc1\xc3\xdb\x82\xa3oR\x87\xa9\x90y\xee\xe6\xe7\xf9\xb4\\Q\x80\xdd\x99d%\xcc\xb5dS v\xc2\r\xeeL\xd5\x11\xd6\x9f+x\xa7\xe7\xabu\
xe6sH\xb1Z\xfb\xcd@\x81\x88\xe9\xc3\xae\x80\x933t/\xea\x19\x1dM!\xe1$r\x1aR\xfb\x04\x1f\xc8r\xb6zR\xaa9\x8dp5.U\x8a\xec\xc1#\xc0T}\xf4\x03P\xdd?!S\x82\x7f\xc8\xbc?"
[2023-05-21 20:26:46,714] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, NONCE 1: BNuubQ31_6fTSwmCnYRB48O-TWbv4uBYAtG-_q9HmOI
[2023-05-21 20:26:46,721] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, RSA SIGN: b'_F\x85\x80\\\xefJ}\xdb\xa2\xe1\xf8e<!\xe8\xac`\x87\xc3\x91\xa0
\x80\xcd\xac\xe3\xd4\xa1\x0eKG/9\x11.z\xde\xd5\xaa}\xda\xf7\xb5\xa8\xcb0L \xc6\xea\xd1q*\x84\xaaA\x11U\x90[\xd4:\xa3\x8b\x1a\x8b\x95_\xef\x0c\x85\xf47O\x05\x95\xf3\x9d
\xb7\x8d\xf1\x13\x15\x88o\x93\xb5\xb3\n-\xa0\xb6\xd6\xe0\xebs\x10-$\xe3\xa0\xa7\x9d\xe1\xfcD\xdd\x8e\xbd\x0fJ\'\xc0\x9d+\x8b\x95%\xefh\t|\xa4\xea\x83,A\xa6\xe6\xa3-_\x
c5\xb9\x1beK\xcc\xba\xbbM\xca\xaa\x1bqy\x94\xdd\xcek\x04\xfe\xae\xa8\x14\x95\xab\xe2\x96\x17\x1b\xc66\xca\xc2\xd7=C1\xd0\xffh\xc2\x12\xa7\xf2\n\xd2\x9d][\xfd\xc3\x1b 4
\x0ff\xfa\xf04\xb4";\x8d\xa4\xef\xb7\x1eJ1\x9eQ?\xef1\xea\xf5\xfe\x7f\r_VhV=\xf5\x86P\x1bB?O%\xa0\x90\'\xd4j,\x80h\\\xd4\x19\x83\xf7\x81\x0e\xc9\x19\xd8\xff9$\xce\xb5\
x81\xeb.E\xe0\xcc\xe7\xfcY'
```

Figure 12

Broker side, first 5 steps of DH scheme

In Figure 12, the broker acquires the connection from the client. Then, it handles the subscription coming from the client to the topic name same as the client ID. When the broker sees the client subscribe to its own id, it starts to prepare the publish message for the step 5 of the Figure 1 (DH scheme). This publish message contains the X509 certificate of the broker, broker DH public key, generated nonce 1, and RSA signature. For the RSA signature part, the broker signs the Diffie Hellman public key of itself, nonce 1 and client ID by using the private key corresponding to its X509 certificate. After the preparation of the message, the broker publishes it using the topic name equal to the client ID.

Figure 13

Client handling the publish message from broker for step 5 of DH

In Figure 13, the client receives the publish message coming from the broker. Client learns the X509 certificate of the broker, broker DH public key, generated nonce 1, and RSA signature. From the X509 certificate of the broker, the client gets the X509 public key of the broker. Then, by using the X509 public key of the broker, the client tries to verify the coming RSA signature. In Figure 13, the client can verify the rsa sign since everything is correct. If the signature is not verified successfully, the client disconnects itself from the broker and connection finishes here.

Figure 14

Preparation of the publish message for step 6 of DH scheme by client

In Figure 14, the client prepares the publish message for step 6 of the Figure 1 (DH scheme). This publish message contains the X509 certificate of the client, DH public key of the client, received nonce 1 from the previous message, and RSA signature. For the RSA signature part, the client signs its own Diffie Hellman public key, nonce 1 and client ID by using the private key corresponding to its X509 certificate. After the preparation of the message, the client publishes it using the topic name 'AuthenticationTopic'. Also, in this step, the Diffie-Hellman

shared key is calculated by the client. At the end of Figure 14, a 256 bit session key is derived from the DH shared key.

Figure 15

Handling of the step 6 publish message of DH scheme by the broker

In Figure 15, the broker receives the publish message coming from the client. The broker learns the X509 certificate of the client, DH public key of the client, nonce 1 (which should be the same as the nonce that was generated in the previous step), and RSA signature. From the X509 certificate of the client, the broker gets the X509 public key of the client. Then, by using the X509 public key of the client, the broker tries to verify the coming RSA signature. In Figure 15, the broker can verify the RSA signature since everything is correct. If the signature is not verified successfully, the broker disconnects the client. If the signature is verified successfully, then the broker compares the nonce received from the client with the nonce generated at the previous step. If nonces are different, the broker disconnects the client and communication finishes here. If they are the same, the shared DH key is calculated by the broker. At the end of Figure 15, a 256 bit session key is derived from the DH shared key.

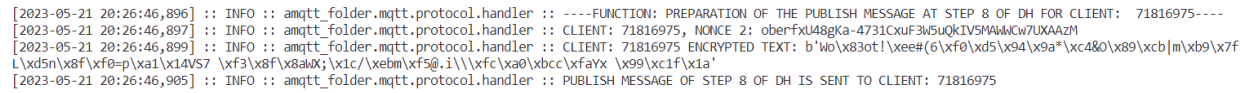

Figure 16

Preparation of the step 8 publish message of DH scheme by the broker

After the broker verifies the RSA signature and compares the nonces, it will send the publish message 8 of Diffie Hellman scheme (Figure 1). This publish message contains nonce 2 and client ID. The message will be encrypted using the session key. For encryption AES256 is used. The session key is generated from the DH shared key between the broker and the client. After the encryption of the message, the broker publishes it using the topic name equal to the client ID which can be shown in Figure 16.

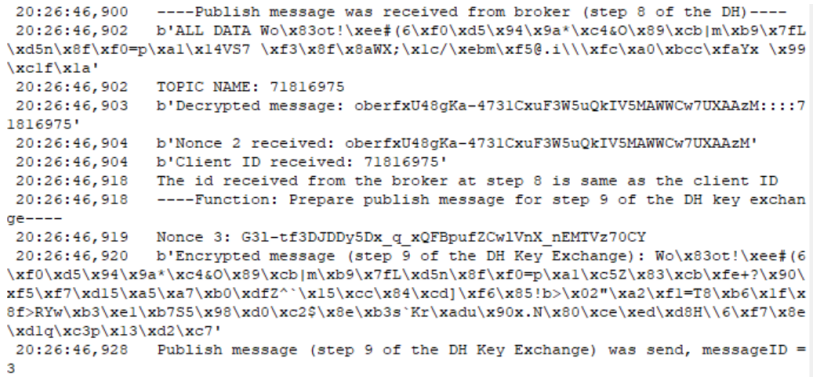

Figure 17

Preparation of the step 9 publish message of DH scheme by the client

At Figure 17, the client receives the step 8 publish message of DH scheme (Figure 1). Then the client decrypts the message by using the session key generated from the DH shared key. The decrypted message contains nonce 2 and client ID. If the received client id and its client id are different, then the client disconnects itself. If two client IDs are the same, then the client prepares the step 9 publish message of DH scheme (Figure 1). The message of step 9 publish

message contains received nonce 2, generated nonce 3 and client ID. This message is encrypted with the session key between the client and the broker. The topic name of this publish packet is ‘AuthenticationTopic’.

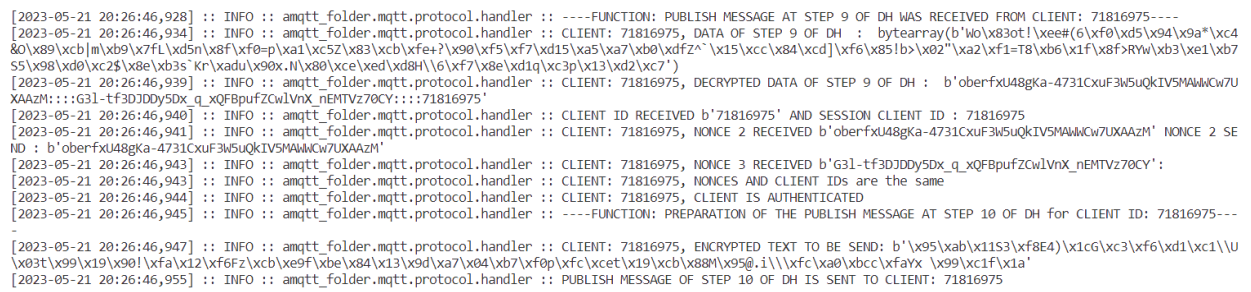

Figure 18

Preparation of the step 10 publish message of DH scheme by the broker

At Figure 18, the broker receives the step 9 publish message of DH scheme (Figure 1). Then the broker decrypts the message by using the session key between the client and the broker. The decrypted message contains nonce 2 (which should be the same as the nonce was generated at the previous step), nonce 3 and client ID. If received client ID and session client ID are the same, and if received nonce 2 and generated nonce 2 are the same, then the client is authenticated by the broker. If any of the nonces or client IDs do not match, the broker disconnects the client. After the client is authenticated, the broker prepares the step 10 publish message of DH scheme (Figure 1). The message of step 10 publish message contains received nonce 3, and client ID. This message is encrypted with the session key between the client and the broker. The topic name of this publish packet is client ID of the receiving client.

```
 20:26:46,949   ----Publish message was received from broker (step 10 of the DH)----
 20:26:46,950   b'Encrypted message: \x95\xab\x11S3\xf8E4)\x1cG\xc3\xf6\xd1\xc1\\U\x0
3t\x99\x19\x90!\xfa\x12\xf6Fz\xcb\xe9f\xbe\x84\x13\x9d\xa7\x04\xb7\xf0p\xfc\xcet\x19\
xcb\x88M\x95@.i\\\xfc\xa0\xbcc\xfaYx \x99\xc1f\x1a'
 20:26:46,951   Topic name: 71816975
 20:26:46,952   b'Decrypted message: G3l-tf3DJDDy5Dx_q_xQFBpufZCwlVnX_nEMTVz70CY::::7
1816975'
 20:26:46,953   b'Received Nonce 3: G3l-tf3DJDDy5Dx_q_xQFBpufZCwlVnX_nEMTVz70CY'
 20:26:46,954   b'Nonce 3 sent before: G3l-tf3DJDDy5Dx_q_xQFBpufZCwlVnX_nEMTVz70CY'
 20:26:46,954   Received nonce 3 and sent nonce 3 are the same
 20:26:46,954   BROKER IS AUTHENTICATED
 20:26:47,045   ----Client was subscribed to its encrypted clientID (step 1 of the ch
oice token schema)
 20:26:47,046   Authenticated Encryption version of the clientID: c84b69c900f8ec0286e
5f11885da7e08aa28d88ca7023ec55049bbab64bb604a6ab57402ed7dd753278c444336223fcf
 20:26:47,063   Suback received, message id: 4
 20:26:47,065   Signature of SUBACK is verified.
```

Figure 19

Handling of the step 10 publish message of DH scheme by the client

At Figure 19, the client receives the step 10 publish message of DH scheme (Figure 1). Then the client decrypts the message by using the session key between the client and the broker. The decrypted message contains nonce 3 (which should be the same as the nonce that was generated at the previous step), and client ID. If received client ID and its own client ID are the same, and if received nonce 3 and generated nonce 3 are the same, then the broker is authenticated by the client. If any of the nonces or client IDs do not match, the client disconnects itself. If authentication finishes successfully, the client subscribes to the authenticated encryption version of the client ID in order to prepare for the choice token communication schema. At the end of the Figure 19, Suback is received from the broker. This Suback packet contains the MAC of the QoS and packet identifier which is the same as the packet identifier and Qos level of the SUBSCRIBE packet. If the MAC of these two packets are the same, then the signature is verified. Otherwise, the client is informed about that error in the case it wants to send the subscribe packet again.

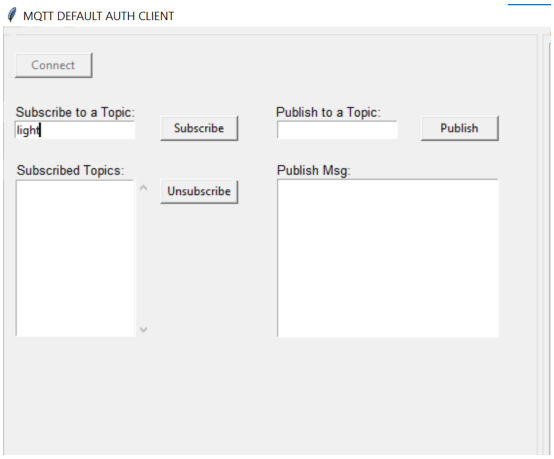

Figure 20

GUI of Client when subscribe to the 'light' topic

At Figure 20, the client subscribes to the topic 'light'. When the client pushes the subscribe button, the choice token of the topic 'light' is requested from the broker as in Figure 2. Then, the client subscribes to the 'light' topic as in Figure 20.

```
 20:35:28,242   Subcribe topic name: light
 20:35:28,248   ----Function: Prepare publish message for step 2 of the choice token
schema----
 20:35:28,250   b'Payload contains the authenticated encryption version of the topic
name for which a choice token is asked: light'
 20:35:28,255   ----Publish was sent to 'choiceToken' topic (step 2 of choice token s
chema)----
 20:35:28,260   ----Puback was received---- (step 3 of choice token schema)
 20:35:28,262   Signature of PUBACK is verified.
```

Figure 21

First figure of client for choice token scheme

At Figure 21, the client program takes the input from the GUI. Then, the client prepares the publish message for step 2 of the choice token schema (Figure 2). The topic name of the publish packet is "choiceToken". The topic name "choiceToken" and its MAC is encrypted

using the session key between the client and the broker. Payload of the publish packet contains the topic name for which the client requests to get a choice token from the broker and the MAC of the topic name. This publish message is sent to the broker and Puback is received as shown in Figure 21. This Puback packet contains the MAC of the packet identifier which is the same as the packet identifier of the publish packet. If the MAC of these packet identifiers are the same, then the signature is verified. Otherwise, the client gets an error message about that situation in the case it wants to send the publish packet again.

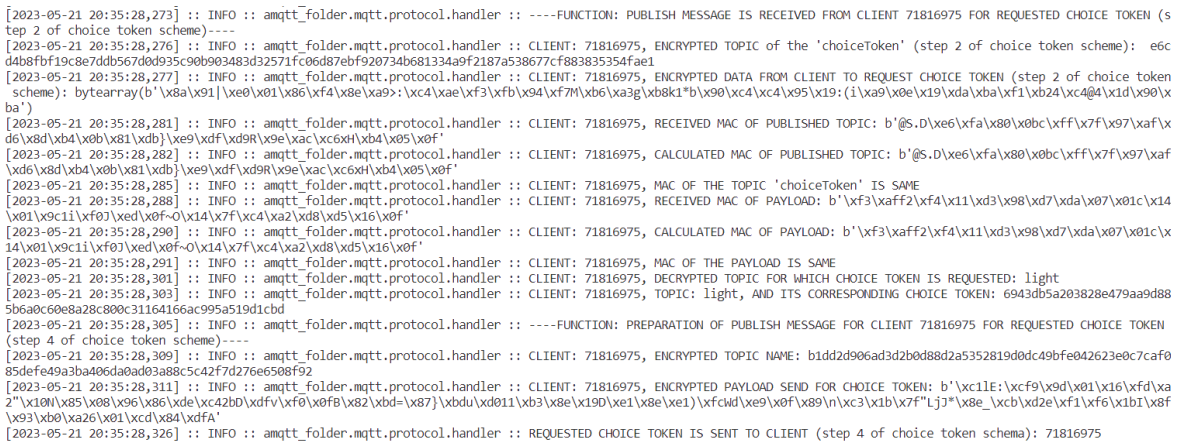

Figure 22

First figure of broker for choice token scheme

At Figure 22, the broker receives the step 2 publish message of choice token scheme (Figure 2). Then the broker decrypts the topic name by using the session key between the client and the broker. After topic name decryption, the received MAC of topic name and calculated MAC of topic name are compared. If they are the same, the broker decrypts the payload of the publish packet. After payload decryption, the received MAC of payload and calculated MAC of payload are compared. If they are the same, the broker checks the database table which contains the topic names and their corresponding choice tokens. If the requested choice token exists in the

database table, the broker gets it from the database. If it is not in the database table, the broker generates a choice token for the requested topic, then inserts this choice token and its topic name to the database table. After the preparation of the choice token, the broker prepares the publish message for step 4 of the choice token schema. The topic name of this publish packet is the authenticated encryption version of the clientID. The payload of the publish packet contains the encrypted version of the requested topic name, its corresponding choice token and calculated MAC of the message. This publish message is sent to the client at the end of Figure 22.

```
 20:35:28,314   ----Publish message was received from broker (step 4 of choice token
schema)---- from topic: bldd2d906ad3d2b0d88d2a5352819d0dc49bfe042623e0c7caf085defe49a
3ba406da0ad03a88c5c42f7d276e6508f92
 20:35:28,320   The content of the message has not been changed. Mac is correct.
 20:35:28,321   b'Topic name: light'
 20:35:28,323   Its choiceToken: 6943db5a203828e479aa9d885b6a0c60e8a28c800c31164166ac
995a519d1cbd
 20:35:28,371   ----Function to subscribe to topic: light
 20:35:28,373   b'MAC of the topic: \xc7\xad\xbb\x07k\xdd\xf8\t\xf6\xdd\xe4{;Iqa\xb6\
xd1\xda\x80e\xf3w\xaae\xaa\x9bPy\xb6#\n'
 20:35:28,375   Authenticated encryption version of the topic:412bf98d280210351e56d13
8063722775e2e8f46323d5bffbea5f681d9a8bb1d5986dab9b22e14a1ccb2b9bc908335af
 20:35:28,377   Subscribed to: 412bf98d280210351e56d138063722775e2e8f46323d5bffbea5f6
 20:35:28,397   Suback received, message id: 6
 20:35:28,399   Signature of SUBACK is verified.
```

Figure 23

Second figure of the client for choice token scheme

At Figure 23, the client receives the step 4 publish message of choice token scheme (Figure 2). After decrypting data, the client program learns the corresponding choice token of requested topic name. If MAC codes are the same, the client calls the function to subscribe to topic ‘light’. Topic name and its MAC code are encrypted using the session key between the client and the broker. Then, the client subscribes to this authenticated encryption version of the topic name. At the end of the Figure 23, Suback is received from the broker. This Suback packet contains the MAC of the QoS and packet identifier which is the same as the packet identifier and Qos level of the SUBSCRIBE packet. If the MAC of these two packets are the same, then the

signature is verified. Otherwise, the client gets an error message about that situation in the case it wants to send the subscribe packet again.

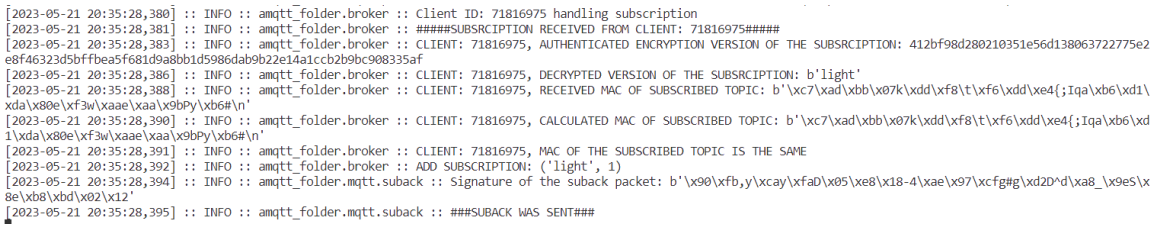

Figure 24

Handling subscription by broker

Figure 24 shows how subscription is handled on the broker side. The broker receives the subscribed topic name as authenticated and encrypted. It decrypts it by using the session key between the broker and the client sending the subscribe packet. Then, it compares the received MAC and calculated MAC . If they are the same, the broker adds the decrypted topic to the subscription list. If they are not the same, an error message is sent to the client in order to inform the client that the subscription failed.

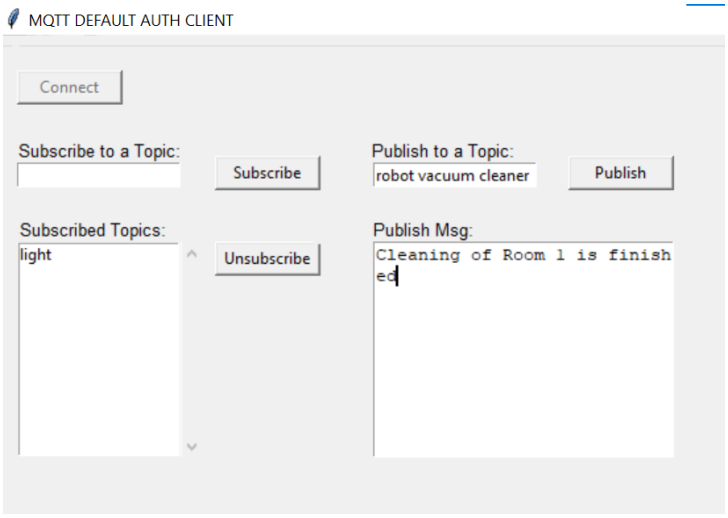

Figure 25

GUI of Client when publishing on the 'robot vacuum cleaner'

At Figure 25, the client publishes on the topic 'robot vacuum cleaner' with the message 'Cleaning of room 1 is finished." When the client pushes the publish button, the choice token of the topic 'robot vacuum cleaner' is requested from the broker as in Figure 25. Then, the client publishes on the 'robot vacuum cleaner' topic as in Figure 25. Also, before this step another client connected to the broker and subscribed to the 'robot vacuum cleaner' topic, in order to see the publish message relay mechanism at the below outputs.

```
 20:45:11,770   Publish topic name: robot vacuum cleaner
 20:45:11,775   Topic name received from the gui:robot vacuum cleaner
 20:45:11,775   Message received from the gui:Cleaning of Room 1 is finished

 20:45:11,777   ----Function: Prepare publish message for step 2 of the choice token
schema----
 20:45:11,779   b'Payload contains the authenticated encryption version of the topic
name for which a choice token is asked: robot vacuum cleaner'
 20:45:11,785   ----Publish was sent to 'choiceToken' topic (step 2 of choice token s
chema)----
 20:45:11,788   ----Puback was received---- (step 3 of choice token schema)
 20:45:11,789   Signature of PUBACK is verified.
```

Figure 26

Publish message by client for requesting choice token

At Figure 26, the client program takes the input from the GUI. Figure 26 is similar to Figure 21. The difference is at Figure 21, the client requests the choice token of 'light' topic in order to subscribe to the topic. Here, the client requests the choice token of 'robot vacuum cleaner' topic in order to publish for that topic.

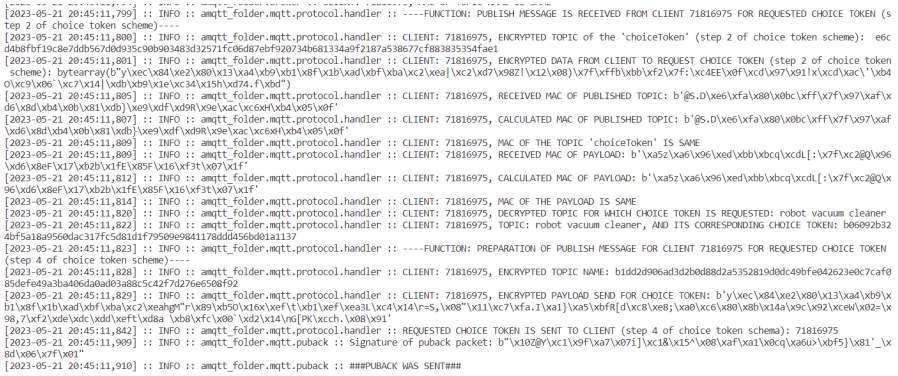

Figure 27

Publish message by broker for sending choice token

Figure 27 is similar to Figure 22. The difference is at Figure 22, the broker sends the choice token of 'light' topic. Here, the broker sends the choice token of 'robot vacuum cleaner'.

Figure 28

Publish on the 'robot vacuum cleaner' topic

At Figure 28, the client receives the step 4 publish message of choice token scheme (Figure 14). After decrypting data, the client program learns the corresponding choice token of requested topic name. If MAC codes are the same, the client calls the function to publish a message to the topic 'robot vacuum cleaner'. Topic name and its MAC code are encrypted using the session key between the client and the broker. Also, the payload is encrypted using the choice token. Then, the authenticated encryption version of the payload is generated. This publish message is sent to the broker at the end of Figure 28. Then Puback is received and MAC of the Puback is verified.

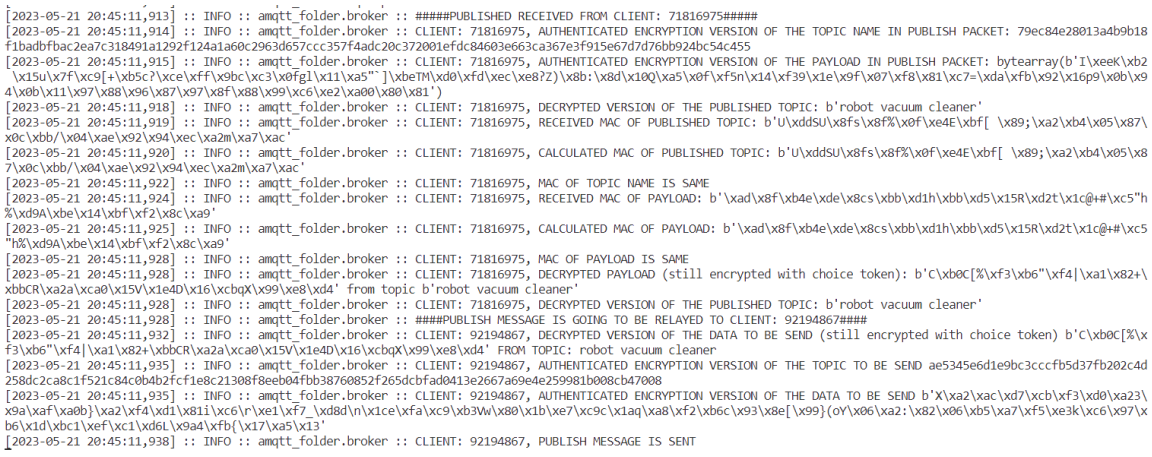

Figure 29

Handling publish message received from client at broker

At Figure 29, the broker receives the publish message from the client. Then the broker decrypts the topic name by using the session key between the client and the broker. After topic name decryption, the received MAC of topic name and calculated MAC of topic name are compared. If they are the same, the broker decrypts the payload of the publish packet. After payload decryption, the received MAC of payload and calculated MAC of payload are compared. If they are the same, the broker adds this publish message to the broadcast queue in

order to forward to interested subscribers. Since, at the previous steps another client subscribed to the published topic, message is relayed to that client. Before sending the publish packet, topic name and payload are signed and encrypted with the session key between the broker and target client.

```
 20:45:11,941   ----Publish message was received from broker
 20:45:11,943   b'Encrypted payload: \x00PX\xa2\xac\xd7\xcb\xf3\xd0\xa23\x9a\xaf\xa0b
}\xa2\xf4\xdl\x81i\xc6\r\xel\xf7_\xd8d\n\xlce\xfa\xc9\xb3Vw\x80\xlb\xe7\xc9c\xlaq\xa8
\xf2\xb6c\x93\x8e[\x99}(oY\x06\xa2:\x82\x06\xb5\xa7\xf5\xe3k\xc6\x97\xb6\xld\xbcl\xef
\xcl\xd6L\x9a4\xfb{\xl7\xa5\xl3'
 20:45:11,946   Encrypted topic: ae5345e6dle9bc3cccfb5d37fb202c4d258dc2ca8clf521c84c0
b4b2fcfle8c21308f8eeb04fbb38760852f265dcbfad0413e2667a69e4e259981b008cb47008
 20:45:11,949   b'Received MAC of topic name: <\xc0\xdlj\xb7m\x9c~\xd2\xe6\x0b.\xae$#
\xa8\xe9\xfc\x94\xa7\xaf-C8\x98\xbe8\xb0_\xl7\x8e\xf6'
 20:45:11,951   b'Calculated MAC of topic name: <\xc0\xdlj\xb7m\x9c~\xd2\xe6\x0b.\xae
$#\xa8\xe9\xfc\x94\xa7\xaf-C8\x98\xbe8\xb0_\xl7\x8e\xf6'
 20:45:11,951   The content of the topic name is not changed. Mac of the topic name i
s correct
 20:45:11,953   Decrypted topic name: robot vacuum cleaner
 20:45:11,956   Choice token of the topic: b06092b324bf5a18a9560dac317fc5d81dlf79509e
9841178ddd456bd01al137
 20:45:11,958   b'Message after decryption with session key: C\xb0C[%\xf3\xb6"\xf4|\x
al\x82+\xbbCR\xa2a\xca0\xl5V\xle4D\xl6\xcbqX\x99\xe8\xd4'
 20:45:11,962   b'Received MAC of payload: \x80.o\x97\xaa\xa2v\x9b\x01\x01\x92\xc67)\
xa6\x94\xel\xb4\xa0|,\xbdo\xf8\x03$\x07\xc9\xb5\xbb\xd0&'
 20:45:11,964   b'Calculated MAC of payload: \x80.o\x97\xaa\xa2v\x9b\x01\x01\x92\xc67
)\xa6\x94\xel\xb4\xa0|,\xbdo\xf8\x03$\x07\xc9\xb5\xbb\xd0&'
 20:45:11,964   The content of the payload is not changed. Mac of the payload is corr
ect
 20:45:11,964   b'Message after decryption with choice token: Cleaning of Room 1 is f
inished\n'
```

Figure 30

Second client who receives the relayed publish message

At Figure 30, the second client which is subscribed to topic ‘robot vacuum cleaner’ receives the relayed publish message. First, it decrypts the topic name by using the session key between the client and the broker. After topic name decryption, the received MAC of topic name and calculated MAC of topic name are compared. If they are the same, the client decrypts the payload of the publish packet. After payload decryption, the received MAC of payload and calculated MAC of payload are compared. If they are the same, the client decrypts the payload using the topic related choice token and learns the message.

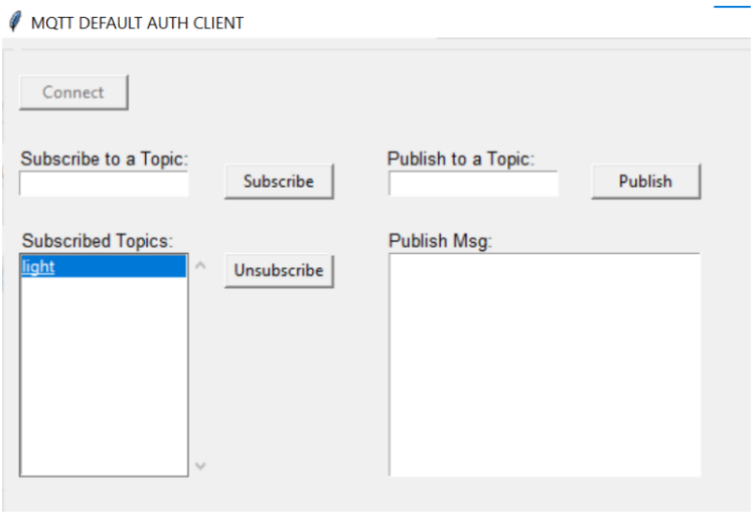

Figure 31

GUI of Client when unsubscribe from the topic 'light'

As can be seen in Figure 31, after the client subscribes to a topic that topic name can be seen from the 'Subscribed Topics' list of the GUI. When the client wants to unsubscribe from a topic, the client should select that topic from the 'Subscribed Topics' list and push the unsubscribe button. More than one topic can be selected from the list for sending unsubscribe.

```
 20:52:31,411   Topic names to unsubscribe received from the gui:light
 20:52:31,413   ---- Function to unsubscribe to topic: light----
 20:52:31,415   b'HMAC of the Topic: \xl4jZ0@\x00\xe5\xbc%\xd7\xcfB \xldb\xel\xee\xaa
PH\x08\xe4\xfbEmpr\xa9\xf37\x00\x8c'
 20:52:31,417   Authenticated Encryption Version of the Topic: 723ed77f7d6773c7eff93d
5bc916f5ad526044c8e165c72ddlfafc6557acbdc52f6246elc3ee6603dl88e95934e2c738
 20:52:31,419   Unsubscribe List : 723ed77f7d6773c7eff93d5bc916f5ad526044c8e165c72ddl
fafc6557acbdc52f6246elc3ee6603dl88e95934e2c738
 20:52:31,421   Unsubscribe to: ['723ed77f7d6773c7eff93d5bc916f5ad526044c8e165c72ddlf
afc6557acbdc52f6246elc3ee6603dl88e95934e2c738']
```

Figure 32

Client side when the client pushes the unsubscribe button from the GUI

In Figure 32, the client unsubscribes to previously subscribed topics. Client program calculates the authenticated encryption version of the topic names that the client wants to unsubscribe and then this packet is sent to the broker.

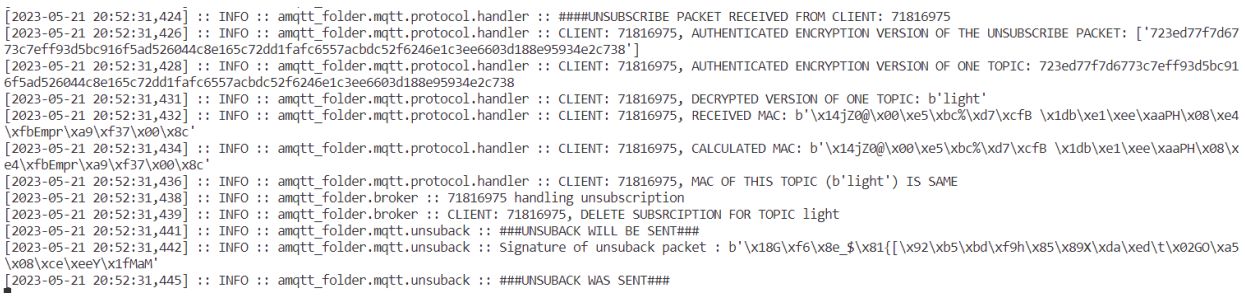

Figure 33

Broker side when client push the unsubscribe button from the GUI

In Figure 33, the broker gets the unsubscribe packet and decrypts it by using the session key between the client and the broker. Then HMAC of the topic names is calculated and compared with the HMAC received in the packet. If they are the same, topic names that the client wants to unsubscribe are deleted from the subscription list and unsuback is sent. Otherwise, an mac error message is sent to client to inform it their unsubscription is not completed

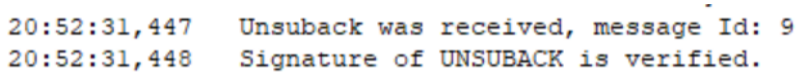

Figure 34

Client side when gets the unsuback packet

In Figure 34, as a response to unsubscribe, the broker sends the Unsuback packet. This Unsuback packet contains the MAC of the packet identifier which is the same as the packet identifier of the Unsubscribe packet. If the MAC of these packet identifiers are the same, then the signature is verified. Otherwise, the client gets an error message about that situation in the case it wants to send the unsubscribe packet again.

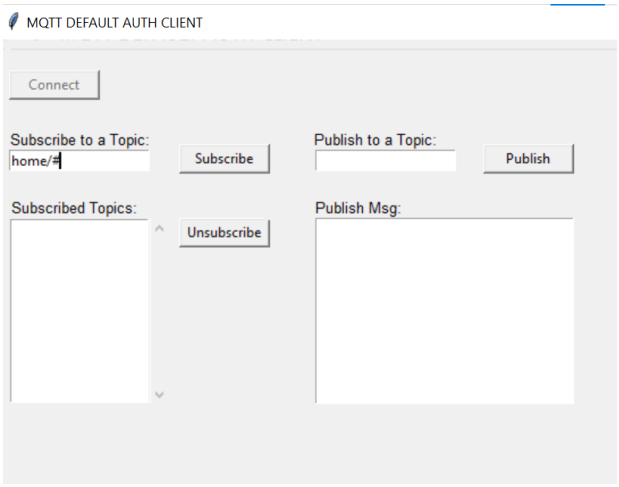

Figure 35

GUI of Client when subscribe to home/#

In Figure 35, the client subscribes to a wildcard topic 'home/#'. Since it subscribes to a wildcard topic, the client has been subscribed to all the topics under the 'home/' such as 'home/light', 'home/room/temperature' etc.

```
 20:57:01,030   ----Function to subscribe to topic: home/#
 20:57:01,032   b'MAC of the topic: \xbe\xcc\x84)\xf7\xfa\xf6\x7fb\\\x8d\xc1\x99\xc5\
x1f\x03\xbbB\xady\x93#\xc8\xf7\xd8\xe0\x1e\x8fT+\xc4:'
 20:57:01,034   Authenticated encryption version of the topic:266b79a4fe1a99f595a1692
a3d3be9fd85365e2e59266264bffc9753ce85462c660c3d2d8f118ed05bfc613abad3e596
 20:57:01,036   Subscribed to: 266b79a4fe1a99f595a1692a3d3be9fd85365e2e59266264bffc97
53ce85462c660c3d2d8f118ed05bfc613abad3e596
```

Figure 36

Handling subscription on client side for wildcard topics

In Figure 36, the client takes the 'home/#' from the GUI and prepares the authenticated encryption version of the topic and sends the subscribe packet to the broker.

```
[2023-05-21 20:57:01,039] :: INFO :: amqtt_folder.broker :: Client ID: 71816975 handling subscription
[2023-05-21 20:57:01,041] :: INFO :: amqtt_folder.broker :: #####SUBSRCIPTION RECEIVED FROM CLIENT: 71816975#####
[2023-05-21 20:57:01,043] :: INFO :: amqtt_folder.broker :: CLIENT: 71816975, AUTHENTICATED ENCRYPTION VERSION OF THE SUBSRCIPTION: 266b79a4fe1a99f595a1692a3d3be9fd853
65e2e59266264bffc9753ce85462c660c3d2d8f118ed05bfc613abad3e596
[2023-05-21 20:57:01,046] :: INFO :: amqtt_folder.broker :: CLIENT: 71816975, DECRYPTED VERSION OF THE SUBSRCIPTION: b'home/#'
[2023-05-21 20:57:01,047] :: INFO :: amqtt_folder.broker :: CLIENT: 71816975, RECEIVED MAC OF SUBSCRIBED TOPIC: b'\xbe\xcc\x84)\xf7\xfa\xf6\x7fb\\\x8d\xc1\x99\xc5\x1f\
x03\xbbB\xady\x93#\xc8\xf7\xd8\xe0\x1e\x8fT+\xc4:'
[2023-05-21 20:57:01,048] :: INFO :: amqtt_folder.broker :: CLIENT: 71816975, CALCULATED MAC OF SUBSCRIBED TOPIC: b'\xbe\xcc\x84)\xf7\xfa\xf6\x7fb\\\x8d\xc1\x99\xc5\x1
f\x03\xbbB\xady\x93#\xc8\xf7\xd8\xe0\x1e\x8fT+\xc4:'
[2023-05-21 20:57:01,050] :: INFO :: amqtt_folder.broker :: CLIENT: 71816975, MAC OF THE SUBSCRIBED TOPIC IS THE SAME
[2023-05-21 20:57:01,051] :: INFO :: amqtt_folder.mqtt.protocol.handler :: ----FUNCTION: send_choice_token_wildDB topicname b'home/#' ----
[2023-05-21 20:57:01,053] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, PREPARATION OF PUBLISH MESSAGE FOR CHOICE TOKEN
[2023-05-21 20:57:01,068] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, topicname_str_exact: home
[2023-05-21 20:57:01,070] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, TOPIC: home/room1/temperature, AND ITS CORRESPONDING CHOICE TOKEN: 15f6f4a
c7fecee63c52b3e60f55551c2d1b7d30720f9852683e4a3f5fc0a415a
[2023-05-21 20:57:01,074] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, PLAIN VERSION OF TOPIC NAME: 71816975
[2023-05-21 20:57:01,075] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, AUTHENTICATED ENCRYPTION VERSION OF TOPIC NAME: b1dd2d906ad3d2b0d88d2a5352
819d0dc49bfe042623e0c7caf085defe49a3ba406da0ad03a88c5c42f7d276e6508f92
[2023-05-21 20:57:01,077] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, AUTHENTICATED ENCRYPTION VERSION OF PAYLOAD SEND FOR CHOICE TOKEN: b'\xe1\
xa3`aL\x05\x80SnIje8\xaf\x9aO9\xceY4\x0fhA4\x84\x84\xfa\xc7\xa3j\x91\xfak\x94h\xe3g\x97V4$\xd9\xd2\xe1&\x7f\xceb\xcb(d\xb4T\xa9\x92\r\xf8\x9aP\x90\xf4(\xb4\xa8\xcd\xfb
"\xf0zW\xdaC\xc1;,\x8e\xaa\x188\xe4=C\xddk\x87\xc6It\xbcD\xed\x03\xebK\xf2%\xdb\x13\xf4\x13\xc2\xb6\xa5\'\xc6.\xfdT\x14\xcc\rc\xe1o\x96 \xce\xce\x9e\xb6*oU\xb4\x7f\x9d
V\x05'
[2023-05-21 20:57:01,093] :: INFO :: amqtt_folder.mqtt.protocol.handler :: REQUESTED CHOICE TOKEN IS SENT TO CLIENT (step 4 of choice token schema): 71816975
[2023-05-21 20:57:01,094] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, TOPIC: home/room2/temperature, AND ITS CORRESPONDING CHOICE TOKEN: f856cf8
3fab9f2b3a0937e2a66e4663ebe312b7cd8dca10238d1f502577eaa6f
[2023-05-21 20:57:01,098] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, PLAIN VERSION OF TOPIC NAME: 71816975
[2023-05-21 20:57:01,099] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, AUTHENTICATED ENCRYPTION VERSION OF TOPIC NAME: b1dd2d906ad3d2b0d88d2a5352
819d0dc49bfe042623e0c7caf085defe49a3ba406da0ad03a88c5c42f7d276e6508f92
[2023-05-21 20:57:01,100] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, AUTHENTICATED ENCRYPTION VERSION OF PAYLOAD SEND FOR CHOICE TOKEN: b"\xe1\
xa3`aL\x05\x80SnIje8\xaf\x9aO9\xceY4\x0fhA4\x84\x84\xfa\xc7\xa3j\x91\xfaOs\xe9\xa7\xaf\xae\x01<\xe6\x94s\xc1q\x1bf\x9cY\x12\xfd\x8f\x98\xe7\xba\xc0\x89\xcf\xe2\xc2\t\x
d3\x07\x141\xbe\xb2\x81\x08\xa3_\xc1\xb1\x84&t\x17\x8e6\xc6\xc1\x88\xa3\xeel\xd5p]6/\x8b\xe79\x8fM\x9f\xe4\xe9\xc1\x9e\xe4'\x9f\x86c\xf4\xa0\x82 2\xaf\xfa\x1ek\\\xbc\x
04\xa8\xea5\x80\x0f\xe0\xca\xec\x0c\x046"
[2023-05-21 20:57:01,116] :: INFO :: amqtt_folder.mqtt.protocol.handler :: REQUESTED CHOICE TOKEN IS SENT TO CLIENT (step 4 of choice token schema): 71816975
[2023-05-21 20:57:01,118] :: INFO :: amqtt_folder.broker :: ADD SUBSCRIPTION: ('home/#', 1)
[2023-05-21 20:57:01,119] :: INFO :: amqtt_folder.mqtt.suback :: Signature of the suback packet: b'\xf5\x1a\xfe\xf0\x0e7w~?\x1d\xcfP\x06M\x18"\xf7\x80\xbbM2\xa2R\xec\x
e0\x14\x87\xb8\xb0_\xc5\xaf'
[2023-05-21 20:57:01,121] :: INFO :: amqtt_folder.mqtt.suback :: ###SUBACK WAS SENT###
```

Figure 37

Handling subscription on broker side for wildcard topics

In Figure 37, the broker gets the subscribe packet. The broker receives the subscribed topic name as authenticated and encrypted. It decrypts it by using the session key between the broker and the client sending the subscribe packet. Then, it compares the received MAC and calculated MAC . If the MACs are not the same, an error message is sent to the client in order to inform the client that the subscription failed. If they are the same, the broker checks its 'choice token' table from the database for topic names under the 'home/'. In our example, there are two topics under the 'home/' and these are 'home/room1/temperature' and 'home/room2/temperature'. First their corresponding choice tokens are sent to the client as authenticated and encrypted. Then, broker adds the subscription of 'home/#' to the subscription list and suback is sent to the client. This suback packet includes the MAC of the packet identifier.

```
 20:57:01,079   ----Publish message was received from broker
 20:57:01,081   b'Encrypted payload: \x00\x80\xe1\xa3`aL\x05\x80SnIje8\xaf\x9aO9\xceY
4\x0fhA4\x84\x84\xfa\xc7\xa3j\x91\xfak\x94h\xe3g\x97V4$\xd9\xd2\xe1&\x7f\xceb\xcb(d\x
b4T\xa9\x92\r\xf8\x9aP\x90\xf4(\xb4\xa8\xcd\xfb"\xf0zW\xdaC\xc1;,\x8e\xaa\x188\xe4=C\
xddk\x87\xc6It\xbcD\xed\x03\xebK\xf2%\xdb\x13\xf4\x13\xc2\xb6\xa5\'\xc6.\xfdT\x14\xcc
\rc\xelo\x96 \xce\xce\x9e\xb6*oU\xb4\x7f\x9dV\x05'
 20:57:01,085   Encrypted topic: b1dd2d906ad3d2b0d88d2a5352819d0dc49bfe042623e0c7caf0
85defe49a3ba406da0ad03a88c5c42f7d276e6508f92
 20:57:01,087   CHOICE TOKEN FOR THE WILDCARD TOPIC AFTER DECRYPTION
 20:57:01,089   Topic name: home/room1/temperature
 20:57:01,090   Its corresponding choice token: 15f6f4ac7fecee63c52b3e60f55551c2d1b7d
30720f9852683e4a3f5fc0a415a
 20:57:01,104   ----Publish message was received from broker
 20:57:01,105   b"Encrypted payload: \x00\x80\xe1\xa3`aL\x05\x80SnIje8\xaf\x9aO9\xceY
4\x0fhA4\x84\x84\xfa\xc7\xa3j\x91\xfaOs\xe9\xa7\xaf\xae\x01<\xe6\x94s\xc1q\x1bf\x9cY\
x12\xfd\x8f\x98\xe7\xba\xc0\x89\xcf\xe2\xc2\t\xd3\x07\x141\xbe\xb2\x81\x08\xa3_\xc1\x
b1\x84&t\x17\x8e6\xc6\xc1\x88\xa3\xee1\xd5p]6/\x8b\xe79\x8fM\x9f\xe4\xe9\xc1\x9e\xe4'
\x9f\x86c\xf4\xa0\x82 2\xaf\xfa\x1ek\\\xbc\x04\xa8\xea5\x80\x0f\xe0\xca\xec\x0c\x046"
 20:57:01,109   Encrypted topic: b1dd2d906ad3d2b0d88d2a5352819d0dc49bfe042623e0c7caf0
85defe49a3ba406da0ad03a88c5c42f7d276e6508f92
 20:57:01,111   CHOICE TOKEN FOR THE WILDCARD TOPIC AFTER DECRYPTION
 20:57:01,112   Topic name: home/room2/temperature
 20:57:01,113   Its corresponding choice token: f856cf83fab9f2b3a0937e2a66e4663ebe312
b7cd8dca10238d1f502577eaa6f
 20:57:01,122   Suback received, message id: 10
 20:57:01,123   Signature of SUBACK is verified.
```

Figure 38

Receiving publish on client side for choice tokens related to wildcard topic

In Figure 38, the client receives the publish messages including the choice tokens related to topics ‘home/room1/temperature’ and ‘home/room2/temperature’. These are the topics found in choice token table on the broker side related to ‘home/#’ wildcard topic. These publish messages are received as authenticated and encrypted. First, the client decrypts the messages using the session key and then compares the HMAC received with the HMAC calculated. If they are the same, the client adds these choice tokens to the dictionary. Also at this step, a Suback packet is received from broker as response to subscribe to topic ‘home/#’. This Suback packet includes the HMAC of the packet identifier of the subscribe packet sent. If the received HMAC and calculated HMAC are the same, the signature is verified. Otherwise, the client is informed in case it wants to send the subscribe again.

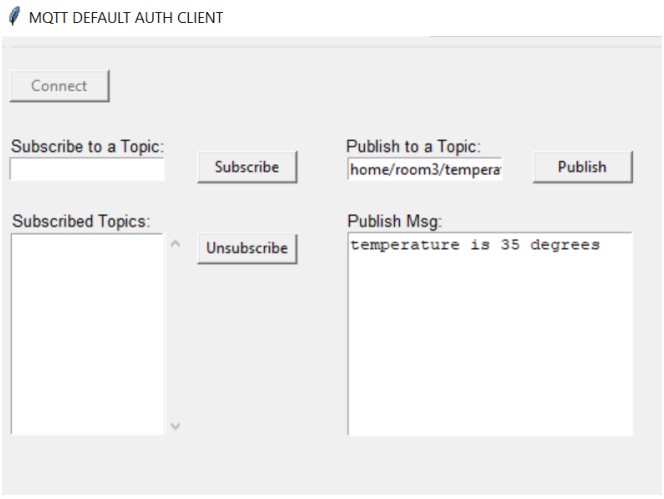

Figure 39

Publish sent to 'home/room3/temperature' from another client

As can be seen in Figure 39, another client sends a publish message for topic 'home/room3/temperature' that is included in the wildcard topic 'home/#'.

```
[2023-05-21 21:02:42,383] :: INFO :: amqtt_folder.broker :: #####PUBLISHED RECEIVED FROM CLIENT: 92194867#####
[2023-05-21 21:02:42,385] :: INFO :: amqtt_folder.broker :: CLIENT: 92194867, AUTHENTICATED ENCRYPTION VERSION OF THE TOPIC NAME IN PUBLISH PACKET: d1f5773f76eb87b1152
28a8741c29bc848b373b55233bae23a141eebd5546eb452faa1a6906114ba0b17f9fb37b1c901eae2de58b2bc50bcf9be2f8a36e35d48
[2023-05-21 21:02:42,386] :: INFO :: amqtt_folder.broker :: CLIENT: 92194867, AUTHENTICATED ENCRYPTION VERSION OF THE PAYLOAD IN PUBLISH PACKET: bytearray(b'A\x91\xb5!
\x8cEB\x91\xc8\xad_\x8f0\x11V\xb4\x16^\x99\x86\xe5)&\xb7\x8b\xe1\x17\x8e\xa1\x0co/0\xd6\x8a\x04\x86\x14\xfe\x18<NS0\x18W\xed\xd7\xea\xc5\xc1\x17\xd8L\x83H\xa3*\x91\xdc
\xbf\xad\xb0\x8e\x0e\x94Sd\xfdv=\x97(\xf6\x93\xac>\xdb\x97\xa0')
[2023-05-21 21:02:42,389] :: INFO :: amqtt_folder.broker :: CLIENT: 92194867, DECRYPTED VERSION OF THE PUBLISHED TOPIC: b'home/room3/temperature'
[2023-05-21 21:02:42,397] :: INFO :: amqtt_folder.broker :: CLIENT: 92194867, RECEIVED MAC OF PUBLISHED TOPIC: b'\xab\xc5,+\x05+\xd4z\xe0\xe6\xfe\xdeJ\x94(\xaa\xfe\xe5
\xd0Y_\xe4\xb7\xa6\x7f4\x02l\x7f%\x9f\x99'
[2023-05-21 21:02:42,401] :: INFO :: amqtt_folder.broker :: CLIENT: 92194867, CALCULATED MAC OF PUBLISHED TOPIC: b'\xab\xc5,+\x05+\xd4z\xe0\xe6\xfe\xdeJ\x94(\xaa\xfe\x
e5\xd0Y_\xe4\xb7\xa6\x7f4\x02l\x7f%\x9f\x99'
[2023-05-21 21:02:42,404] :: INFO :: amqtt_folder.broker :: CLIENT: 92194867, MAC OF TOPIC NAME IS SAME
[2023-05-21 21:02:42,407] :: INFO :: amqtt_folder.broker :: CLIENT: 92194867, RECEIVED MAC OF PAYLOAD: b'x\x9dqo\xc8\xaf*\x07\xa6\xad\xf4\xa3g\xb9\xeb\xb6\xe2:2\xb7\t\
x14\xec\xcfaA]\x9fX\x9b\x8fS'
[2023-05-21 21:02:42,411] :: INFO :: amqtt_folder.broker :: CLIENT: 92194867, CALCULATED MAC OF PAYLOAD: b'x\x9dqo\xc8\xaf*\x07\xa6\xad\xf4\xa3g\xb9\xeb\xb6\xe2:2\xb7\
t\x14\xec\xcfaA]\x9fX\x9b\x8fS'
[2023-05-21 21:02:42,413] :: INFO :: amqtt_folder.broker :: CLIENT: 92194867, MAC OF PAYLOAD IS SAME
[2023-05-21 21:02:42,414] :: INFO :: amqtt_folder.broker :: CLIENT: 92194867, DECRYPTED PAYLOAD (still encrypted with choice token): b"\xb5\xac&%=\xeav\xdc\xca\xb9e\x0
f\xb4\xc0\x18m\xc1u\xaa\x93'8\xa19cs.\x8f\xfe\xfcI\x04" from topic b'home/room3/temperature'
[2023-05-21 21:02:42,415] :: INFO :: amqtt_folder.broker :: CLIENT: 92194867, DECRYPTED VERSION OF THE PUBLISHED TOPIC: b'home/room3/temperature'
[2023-05-21 21:02:42,418] :: INFO :: amqtt_folder.mqtt.protocol.handler :: ----FUNCTION: PREPARATION OF PUBLISH MESSAGE FOR SENDING CHOICE TOKEN TO 71816975 FOR TOPIC
home/room3/temperature ----
[2023-05-21 21:02:42,425] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, TOPIC: home/room3/temperature, AND ITS CORRESPONDING CHOICE TOKEN: 5012306
bfda9d01c05156250117936 6e0d272d604521fdb8f5e3d376797f2b91
[2023-05-21 21:02:42,426] :: INFO :: amqtt_folder.mqtt.protocol.handler :: ----FUNCTION: PREPARATION OF PUBLISH MESSAGE FOR CLIENT 71816975 FOR REQUESTED CHOICE TOKEN
(step 4 of choice token scheme)----
[2023-05-21 21:02:42,428] :: INFO :: amqtt_folder.mqtt.protocol.handler :: Plain version of the message: b"wildcardChoiceToken:::home/room3/temperature::::P\x120k\xfd
\xa9\xd0\x1c\x05\x15bP\x11y6n\r'-E!\xfd\xb8\xf5\xe3\xd3vy\x7f+\x911False8"
[2023-05-21 21:02:42,430] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, ENCRYPTED TOPIC NAME: b1dd2d906ad3d2b0d88d2a5352819d0dc49bfe042623e0c7caf0
85defe49a3ba406da0ad03a88c5c42f7d276e6508f92
[2023-05-21 21:02:42,431] :: INFO :: amqtt_folder.mqtt.protocol.handler :: CLIENT: 71816975, ENCRYPTED PAYLOAD SEND FOR CHOICE TOKEN: b'\xe1\xa3`aL\x05\x80SnIje8\xaf\x
9a09\xceY4\x0fhA4\x84\x84\xfa\xc7\xa3j\x91\xfa\x11\x90\xf8\x1b\xa5M\xf5\x81\xc2e\xa7\x10\xa8M\x82*\x90\xe5\x81L\xdf\x08\xb3<\xa9i\xc8 0\xd7\xfc\x97?\x84\xc9\x00\xe7\xe
9Uv\xc3!\xf4xT\x92#CQ\x84KM\xaa\xc1\x1f3\xd2\x9c\xca\xd4\x1f7J\x94\xa9\x19\x10\x0ep\xf7\xf1k\xcc\x17\xde\xd5\x1d\x02p9u\xba\xe4\x99w\x98\x17\x8b\x9aD*^3\xc8\x86'
[2023-05-21 21:02:42,440] :: INFO :: amqtt_folder.mqtt.protocol.handler :: REQUESTED CHOICE TOKEN IS SENT TO CLIENT (step 4 of choice token schema): 71816975
[2023-05-21 21:02:42,441] :: INFO :: amqtt_folder.broker :: ####PUBLISH MESSAGE IS GOING TO BE RELAYED TO CLIENT: 71816975####
[2023-05-21 21:02:42,441] :: INFO :: amqtt_folder.broker :: CLIENT: 71816975, DECRYPTED VERSION OF THE DATA TO BE SEND (still encrypted with choice token) b"\xb5\xac&%
=\xeav\xdc\xca\xb9e\x0f\xb4\xc0\x18m\xc1u\xaa\x93'8\xa19cs.\x8f\xfe\xfcI\x04" FROM TOPIC: home/room3/temperature
[2023-05-21 21:02:42,442] :: INFO :: amqtt_folder.broker :: CLIENT: 71816975, AUTHENTICATED ENCRYPTION VERSION OF THE TOPIC TO BE SEND 9284c7c4ef76b531054ddd92b1bbbf34
6d1e953fccff9b859d5b07f945c4d926fb6c9fc59b2fea4d4e4ce0ad781aa37d4cfe0e7eb3569e49f2279f91638bb511
[2023-05-21 21:02:42,443] :: INFO :: amqtt_folder.broker :: CLIENT: 71816975, AUTHENTICATED ENCRYPTION VERSION OF THE DATA TO BE SEND b'\xab_\xca\xe4\xedAy"=\xe8\x05T\
xa0c\xabW\xa8\xb6\xa0\xaf}\x87S\xba\xbd\xf1\xde\xdd\x14\x054\x19\xa3`\xe9\xa3\xacG\xa1B\xf1\xce\xf4\x94\xb7{\xc3*>C\x8br_ \xd3\xb7\xe6\x81\xa5azz1N7\x0b\x1bu9\xdf?U8PI
H\xd4\x0eDR'
[2023-05-21 21:02:42,445] :: INFO :: amqtt_folder.broker :: CLIENT: 71816975, PUBLISH MESSAGE IS SENT
```

Figure 40

Broker relaying mechanism for the message 'home/#'

In the Figure 39, another client (ID: 18499112) published the ‘home/room3/temperature’ when our client is already subscribed to the ‘home/#’ topic as can be seen in Figure 36. In this Figure 40, the broker relays the message received for topic ‘home/room3/temperature’ to our client (ID: 2678449). Before relaying the message, it controls whether the choice token of that topic has been sent to the client, who subscribed to the wildcard topic, before. If it was not sent, the broker first sends the choice token to our client for the topic ‘home/room3/temperature’ in an authenticated and encrypted way. Then, the broker relays the publish packet by making the necessary MAC calculations and encryptions.

```
 21:02:42,433   ----Publish message was received from broker
 21:02:42,434   b'Encrypted payload: \x00\x80\xe1\xa3`aL\x05\x80SnIje8\xaf\x9aO9\xceY
4\x0fhA4\x84\x84\xfa\xc7\xa3j\x91\xfa\x11\x90\xf8\x1b\xa5M\xf5\x81\xc2e\xa7\x10\xa8M\
x82*\x90\xe5\x81L\xdf\x08\xb3<\xa9i\xc8 0\xd7\xfc\x97?\x84\xc9\x00\xe7\xe9Uv\xc3!\xf4
xT\x92#CQ\x84KM\xaa\xc1\x1f3\xd2\x9c\xca\xd4\x1f7J\x94\xa9\x19\x10\x0ep\xf7\xf1k\xcc\
x17\xde\xd5\x1d\x02p9Uu\xba\xe4\x99w\x98\x17\x8b\x9aD*`3\xc8\x86'
 21:02:42,436   Encrypted topic: b1dd2d906ad3d2b0d88d2a5352819d0dc49bfe042623e0c7caf0
85defe49a3ba406da0ad03a88c5c42f7d276e6508f92
 21:02:42,437   CHOICE TOKEN FOR THE WILDCARD TOPIC AFTER DECRYPTION
 21:02:42,438   Topic name: home/room3/temperature
 21:02:42,439   Its corresponding choice token: 5012306bfda9d01c051562501179366e0d272
d604521fdb8f5e3d376797f2b91
 21:02:42,446   ----Publish message was received from broker
 21:02:42,446   b'Encrypted payload: \x00P\xab_\xca\xe4\xedAy"=\xe8\x05T\xa0c\xabW\xa
8\xb6\xa0\xaf}\x87S\xba\xbd\xf1\xde\xdd\x14\x054\x19\xa3`\xe9\xa3\xacG\xa1B\xf1\xce\x
f4\x94\xb7{\xc3*>C\x8br_ \xd3\xb7\xe6\x81\xa5azzlN7\x0b\x1bu9\xdf?U8PIH\xd4\x0eDR'
 21:02:42,447   Encrypted topic: 9284c7c4ef76b531054ddd92b1bbbf346d1e953fccff9b859d5b
07f945c4d926fb6c9fc59b2fea4d4e4ce0ad781aa37d4cfe0e7eb3569e49f2279f91638bb511
 21:02:42,449   b'Received MAC of topic name: \x16\xf6<-\xd0b\xd9\x88\xeb3L}H@7\xc9\x
eb\xf5\x89\x03\xc9\x9d\x85]\xd0\xe1\x1c\xcc\x1b\xa7F\xfd'
 21:02:42,450   b'Calculated MAC of topic name: \x16\xf6<-\xd0b\xd9\x88\xeb3L}H@7\xc9
\xeb\xf5\x89\x03\xc9\x9d\x85]\xd0\xe1\x1c\xcc\x1b\xa7F\xfd'
 21:02:42,451   The content of the topic name is not changed. Mac of the topic name i
s correct
 21:02:42,452   Decrypted topic name: home/room3/temperature
 21:02:42,452   Choice token of the topic: 5012306bfda9d01c051562501179366e0d272d6045
21fdb8f5e3d376797f2b91
 21:02:42,453   b"Message after decryption with session key: \xb5\xac&%=\xeav\xdc\xca
\xb9e\x0f\xb4\xc0\x18m\xclu\xaa\x93'8\xa19cs.\x8f\xfe\xfcI\x04"
 21:02:42,455   b'Received MAC of payload: n\xf4\xad[\xe4\x0b\xe8X\xde\xe5\x1dB@\x1e0
9\x9af\x98V\xbc\xe3\xf2V\xbf\t\xe9j\xd3`4C'
 21:02:42,456   b'Calculated MAC of payload: n\xf4\xad[\xe4\x0b\xe8X\xde\xe5\x1dB@\x1
e09\x9af\x98V\xbc\xe3\xf2V\xbf\t\xe9j\xd3`4C'
 21:02:42,457   The content of the payload is not changed. Mac of the payload is corr
ect
 21:02:42,457   b'Message after decryption with choice token: Temperature is 35 degre
es\n'
```

Figure 41

Receiving the message from topic ‘home/room3/temperature’

In Figure 41, the client who subscribed to the topic ‘home/#’ gets the message for the topic ‘home/room3/temperature’. First the client gets the choice token for the

‘home/room3/temperature’ topic from the broker since the broker sends this choice token before relaying the publish message to the client. The client decrypts the packets and learns the corresponding choice token. Then the client gets another publish message. This time, this message is the message which the broker relays at the end of the Figure 40. The client decrypts the topic name and payload of this packet and compares the related MAC values. If everything is correct, the client learns the message by decrypting the message by using the choice token of that topic.

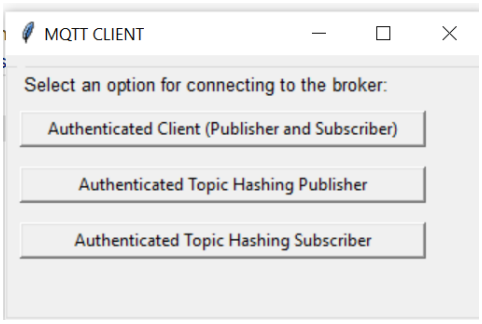

Figure 42

Starting GUI (Topic Hashing)

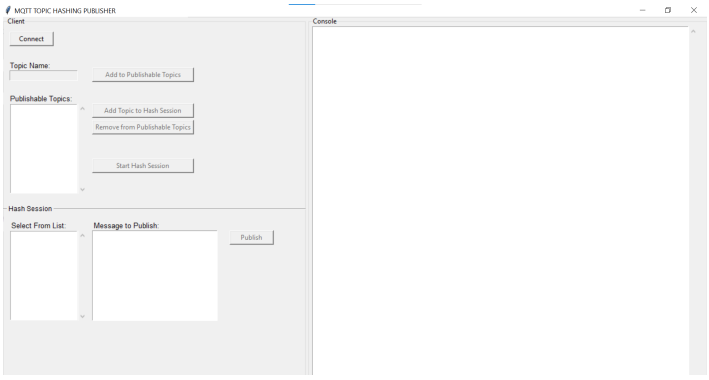

Figure 43

GUI of Topic Hashing Publisher

Figure 42 contains the starting menu of GUI which was mentioned before in Figure 9. This time, the outputs of the topic hashing scheme (Figure 4, Figure 5, Figure 6) will be examined. So, the second button for the publisher and the third button for the subscriber are going to be used. For this scheme, none of the outputs of the broker will be shown since the operations on the broker are the same as above communications. Also, some encryption outputs will be removed from the client. Topic hashing scheme was implemented on the top of the default authenticated client that was implemented in the first option of the starting menu of GUI. So, all the operations shown above also exist here but here the main point is different. So,

outputs mostly will contain the extra features implemented for this topic hashing scheme in order not to repeat previous features one more time.

Figure 43 contains the graphical user interface of the topic hashing publisher. At the beginning only the connect button is enabled, the other buttons and text boxes are disabled. When the client pushes the connect button Diffie Hellman key exchange is started.

```
 21:10:57,171   Get the choice token for topic 'topicHashing'.
 21:10:57,171   ----Function: Prepare publish message for step 2 of the choice token
schema----
 21:10:57,175   b'Payload contains the authenticated encryption version of the topic
name for which a choice token is asked: topicHashing'
 21:10:57,178   ----Publish was sent to 'choiceToken' topic (step 2 of choice token s
chema)----
 21:10:57,181   Suback received, message id: 4
 21:10:57,181   Signature of SUBACK is verified.
 21:10:57,185   ----Puback was received---- (step 3 of choice token schema)
 21:10:57,185   Signature of PUBACK is verified.
 21:10:57,210   ----Publish message was received from broker (step 4 of choice token
schema)---- from topic: 31e9e36feb607ad43a36af2455d3c08fabad584265c076863798eaa4ab7b0
9f1584bd1bf0cabd9c098189d4d43325cce
 21:10:57,212   The content of the message has not been changed. Mac is correct.
 21:10:57,213   b'Topic name: topicHashing'
 21:10:57,214   Its choiceToken: e67bb2d02344c556773dfb6876e14b0331302418b9d0c70a8542
286217f8f6a1
 21:10:57,280   Get the choice token for topic 'newParticipant'.
 21:10:57,281   ----Function: Prepare publish message for step 2 of the choice token
schema----
 21:10:57,282   b'Payload contains the authenticated encryption version of the topic
name for which a choice token is asked: newParticipant'
 21:10:57,284   ----Publish was sent to 'choiceToken' topic (step 2 of choice token s
chema)----
 21:10:57,286   ----Puback was received---- (step 3 of choice token schema)
 21:10:57,287   Signature of PUBACK is verified.
 21:10:57,309   ----Publish message was received from broker (step 4 of choice token
schema)---- from topic: 31e9e36feb607ad43a36af2455d3c08fabad584265c076863798eaa4ab7b0
9f1584bd1bf0cabd9c098189d4d43325cce
 21:10:57,312   The content of the message has not been changed. Mac is correct.
 21:10:57,313   b'Topic name: newParticipant'
 21:10:57,313   Its choiceToken: f5273484aa2d9032c963db34c740c3125725d453fcec269809ed
983lade28850
 21:10:57,388   ----Function to subscribe to topic: newParticipant
 21:10:57,389   b'MAC of the topic: \xb6Tr\xfe\xe7\xf4&\x9aS\xf7z\x08\xcd[\x12\x96x\x
1c\xacEv\xad-\xec\xe8\xdc\xaa\x04\xbc\x1e\xf1b'
 21:10:57,389   Authenticated encryption version of the topic:194d0c4da65b5daa9c6f38b
d0f1308c0d1282c2074ba9a300789caee1724653ddfa1331dfe5ae4912f3502af1e00773377b57e958c08
7d3d8e91e1a6a9e9b0cb
 21:10:57,391   Subscribed to: 194d0c4da65b5daa9c6f38bd0f1308c0d1282c2074ba9a300789ca
ee1724653ddfa1331dfe5ae4912f3502af1e00773377b57e958c087d3d8e91e1a6a9e9b0cb
 21:10:57,391   The publisher was subscribed to newParticipant to learn the   subscri
bers who want to subscribe itself.
 21:10:57,395   Suback received, message id: 7
 21:10:57,395   Signature of SUBACK is verified.
```

Figure 44

Extra Features of the Connection step of Topic Hashing Publisher

When the client pushes the connection button in the topic hashing publisher GUI, first Diffie Hellman key exchange is realized as in the normal authenticated client. All the Diffie Hellman steps are the same with the default-auth client which is in Figure 10. After the broker is authenticated by the client, there are extra steps in this connection step. These extra steps can be

seen in Figure 44. After the session key is established between the topic hashing publisher and the broker, the topic hashing publisher gets the choice tokens of the topics ‘topicHashing’ and ‘newParticiant’ in an encrypted and MACed way similar to the default authenticated client. The choice tokens of these topics are required for the future communications. Also, after getting the choice tokens, the publisher subscribes to the ‘newParticipant’ topic. This topic is used for learning the new participants who will subscribe to the topics of itself.

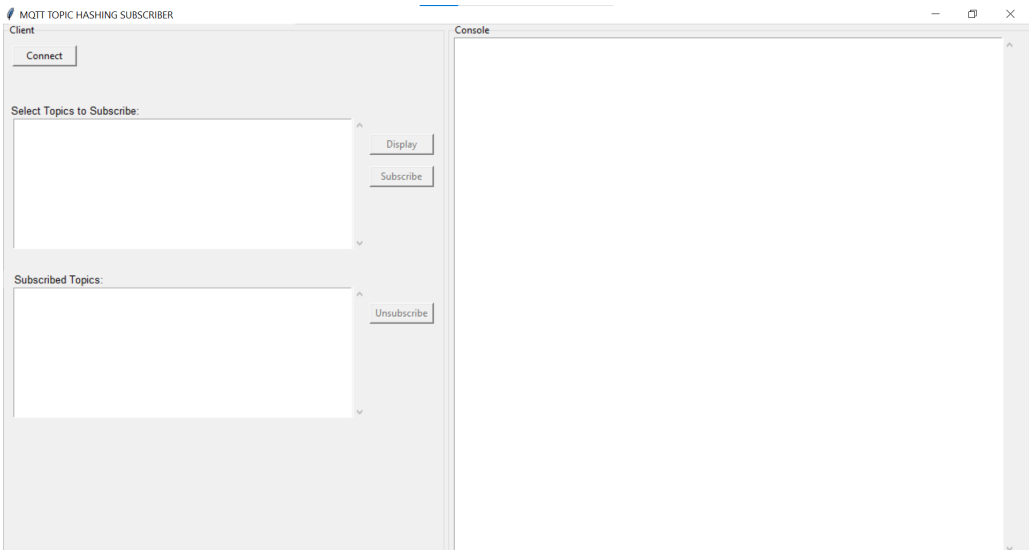

Figure 45

GUI of Topic Hashing Subscriber

Figure 45 contains the graphical user interface of the topic hashing subscriber. At the beginning only the connect button is enabled, the other buttons and text boxes are disabled. When the client pushes the connect button Diffie Hellman key exchange is started.

```
 21:10:54,577   ----Function to publish to topic: newParticipant
 21:10:54,578   Authenticated encryption version of the topic name to be published: d
b8604d7228f9808f1c0c9f53679f796161c80f62bb7e1ff36bc66abd0e1a768cf24dd03b25765020ce24e
244f14aae463dc2dcc7938fa479325b23122315ca0
 21:10:54,580   b'Message after encryption with the choice token: :\xb2h\x17*,\xef\x1
9}~_\x8b}`\x9f\xee'
 21:10:54,581   b'Message after authenticated encyption with the session key: \x02P\x
a47\xb4\x92\xc0A\xf8\x0f\x85\xc5\x0b|J\xb5&$_\x12\x8f\x8e\xfa\xafT\x8eC\x91\x1fJI\x94
\xb7\xf9\x1e\xc2\x88\x8b\xb4\xb6\xa2R\x89\xce\xc1X\xa0F\x02\x12mE\xa3]f\r\xd1\xafmWA\
xda<V'
 21:10:54,582   ----Function to subscribe to topic: topicHashing
 21:10:54,583   b'MAC of the topic: \x9a~=\xdfL\xa3~C\x16\xd6\xb3\n\xa8a\x0b\x02)\xc1
*\x7fo)\x8d\xc7\x0cf\xf6\xaa\xc2\xcfQ\x8d'
 21:10:54,584   Authenticated encryption version of the topic:291de48ee47a6ead9b9578d
920a7f10d99ac0735c407e42e483251e08191799407871a2d71873a456db75b203c7ee6a42dc39c405853
8d25bd6b813e79214513
 21:10:54,584   Puback was received, messageID =7
 21:10:54,585   Subscribed to: 291de48ee47a6ead9b9578d920a7f10d99ac0735c407e42e483251
e08191799407871a2d71873a456db75b203c7ee6a42dc39c4058538d25bd6b813e79214513
 21:10:54,585   Signature of PUBACK is verified.
 21:10:54,602   Suback received, message id: 8
 21:10:54,603   Signature of SUBACK is verified.
```

Figure 46

Extra Features of the Connection step of Topic Hashing Subscriber

When the client pushes the connection button in the topic hashing publisher GUI, first Diffie Hellman key exchange is realized as in the normal authenticated client. All the Diffie Hellman steps are the same with the default-auth client which is in Figure 10. After the broker is authenticated by the client, the subscriber gets the choice tokens of the topics ‘topicHashing’ and ‘newParticiant’ similar to the topic hashing publisher shown in Figure 44. The choice tokens of these topics are required for the future communications. Also, after getting the choice tokens, the subscriber publishes to the ‘newParticipant’ topic to inform the publisher. Then, it subscribes to ‘topicHashing’ topic. It will learn the initial seed mappings for the real topic names and the random polynomial from that topic when the topic hashing publisher publishes.

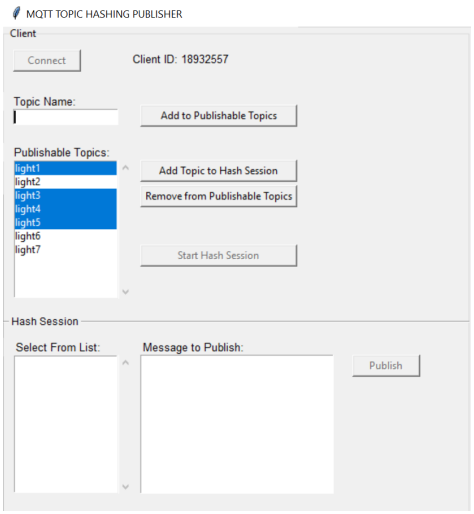

Figure 47

GUI of Topic Hashing Publisher after connection

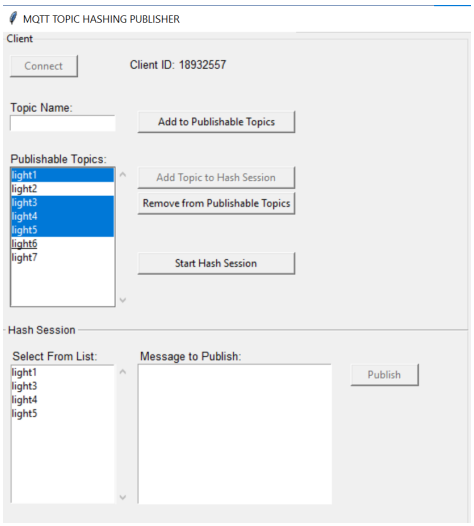

Figure 48

GUI of Topic Hashing Publisher after it publishes the seeds

After the connection process, the publisher writes topic names and pushes the 'Add to Publishable Topic' button to add these topics to the 'Publishable Topics' list as can be seen in Figure 47. The topic hashing publisher will select topics from that list and the selected topics will be used during the hash session. The publisher can add new topics to the 'Publishable Topics' list or remove topics from that list anytime it wants.

After the publisher selects topics from 'Publishable Topics' list and pushes the 'Add Topic to Hash Session' button, the topics will be added to hash session list called 'Select From List' as can be seen in Figure 48 and no more topics can be added to that list before hash session ends. When the publisher pushes the 'Add Topic to Hash Session' button, the initial seeds of the topics that are selected and random polynomial coefficients will be sent to the subscribers who subscribed to 'topicHashing' topic.

```
 21:13:36,638   Topic Name: light1 and it corresponding seed: 7598414945
 21:13:36,640   Topic Name: light3 and it corresponding seed: 7192853369
 21:13:36,640   Topic Name: light4 and it corresponding seed: 7528930364
 21:13:36,643   Topic Name: light5 and it corresponding seed: 4647678835
 21:13:36,645   Polynomial Coefficient: 3087
 21:13:36,647   Authenticated encryption version of the topic name to be published: 3
b2ea24f0a3feb946eed1979f8577bc60a39d681e73675d04572340136873d93319d8acd6dac93d2897228
04f2b097d8b2dcb36759b254f2800bea85c6229a5e
 21:13:36,651   b"Seeds and polynomial after encryption with the choice token: z\xd1\
xd8\xe6\x14\x027\xe9\xcf\xe0\xd3x(\x95\x01]p\x8b\\:\xa9\x17\x0fv\xc2\xe8/\xf6\xff\xa4
\xc3\x84\xf1.\xd3&\x89\xdf\x98Z\x98\xf7\xdb#\xf2GCxqm\xa3\xfe7\xf7\r\x18\x960\xa4\xfb
\xf7\xf5j:^\x03/j\x1a\x8ele.\x91_^\xda\x7fI\xf5\xb1dA=\xef\x89\xf5\xe4\xd6VA\x9b\x18\
xbc\xa3\xb2\xbb\x14\x10\xb2\x92\x91\x03'\xb0\xd61)\xd5\xb7\xa7\x97"
 21:13:36,655   b'Message after authenticated encyption with the session key: \x0fF\x
b7\xa5\xa28\xebH\xd1\xb5\x8bbJUy\x9e"I\xcb\xc7\x06B\x8a\xd1\t\xa7L\x07MQFv[\xdb`)R\xe
a\xe0\xb8X\xff\xe5;\xacv\xf1\xd9\xd2"\xeb\xebx\x06\'\xe1\xfc\x92\xfe\xe4\xc9\xd8\x9d\
x94\xdet\xb7\x9aZ\x12\x86H\x87\x8d\xafE\xfd\xdaZ\xcd\xbcx\xea\xa7\x94\x87\x81\x82\xa8
\x02g\x18I\xf1s\xbf\x91d\x93\x1d:-XYc\xc4\xd1H\xd7\xcdj\x8d\xa7PT\xdb\x11.\xaa\x896\x
a1fxF(\xb45T\x0ezdH\xbf\xf5\xbc\x1a;\xae\x92\xea!\xe1\x05\xb5\xfa|\xbf\xac_\x17\x1eFX
%3\x81\xd9\xe1('
 21:13:36,661   Salt polynomial: 1018 and its count: 1
 21:13:36,666   Puback was received, messageID =12
 21:13:36,667   Signature of PUBACK is verified.
 21:13:36,816   Topic names: light1, and its first hash value: 43492eb90abf20c45c021f
4acbf6efle979c0b1178408c6728d9c4f76498e33a89af7e04b5d4bac64e376f464ef5a3752c69d63f619
8c8e0a6a6bdaf42c80d98
 21:13:37,017   Topic names: light3, and its first hash value: ae7f7989540c127406e299
af4d55c6ea14e12ddee4c548919af0bac5f0c3652bd2c94d70b21de8382ceb706872647e8c47a23e087cc
37cd41bb012d4d619dc3b
 21:13:37,194   Topic names: light4, and its first hash value: 3d7190b98baab188121lc2
4160f616c683426fe8242f274b8011698fba65f0ee37c3ef9c9f6b66c17024607658f654f5edeebb29e15
18e552cec090203c79d37
 21:13:37,291   Topic names: light5, and its first hash value: 9adbca9d769a345e233950
9e48f90bcb4f4d53fe631bf772d829e8c417d1acabdacc40fd5b183c29e99223cb6954195d2035720c04c
134ab4c5b80a194522545
```

Figure 49

Topic hashing publisher after pushing the 'Add to Hash Session' button

When the topic hashing publisher pushes the 'Add Topic to Hash Session' button, first the choice token for the selected topics will be received from the broker in an authenticated and encrypted way. Output of this process is not shown in Figure 49 since it is the known choice token taking process. Then the initial seed values for the topic names and random polynomial coefficients are formed. These are first encrypted with the choice token of 'topicHashing' topic. Then the HMAC is calculated and added at the end of the message. After that, the whole message is encrypted with the session key between the broker and publisher and sent to the broker. After publishing the seed mappings from the 'topicHashing' topic, the publisher calculates the polynomial that will be used for the salting purposes. At the end of the Figure 49,

the first hash values of selected topics is calculated by using the initial seed values and the polynomial.

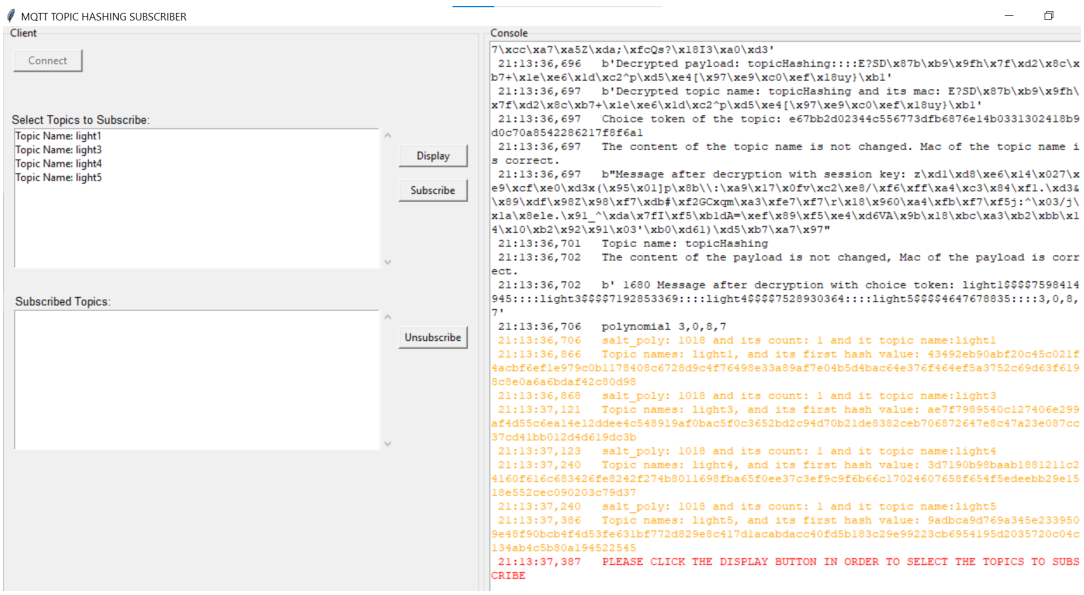

Figure 50

Topic hashing subscriber after getting the initial seeds of topics

In Figure 50, the topic hashing subscriber gets the initial seed values from the 'topicHashing' topic in an authenticated and encrypted way. Then, it calculates the polynomial and the first hash values of received topics as can be seen in Figure 50. After seeds mapping comes, the subscriber should push the 'Display' button to add the received topic names to the 'Select Topics to Subscribe' list. Now, the subscriber can select topics from that list in order to subscribe during the hash session.

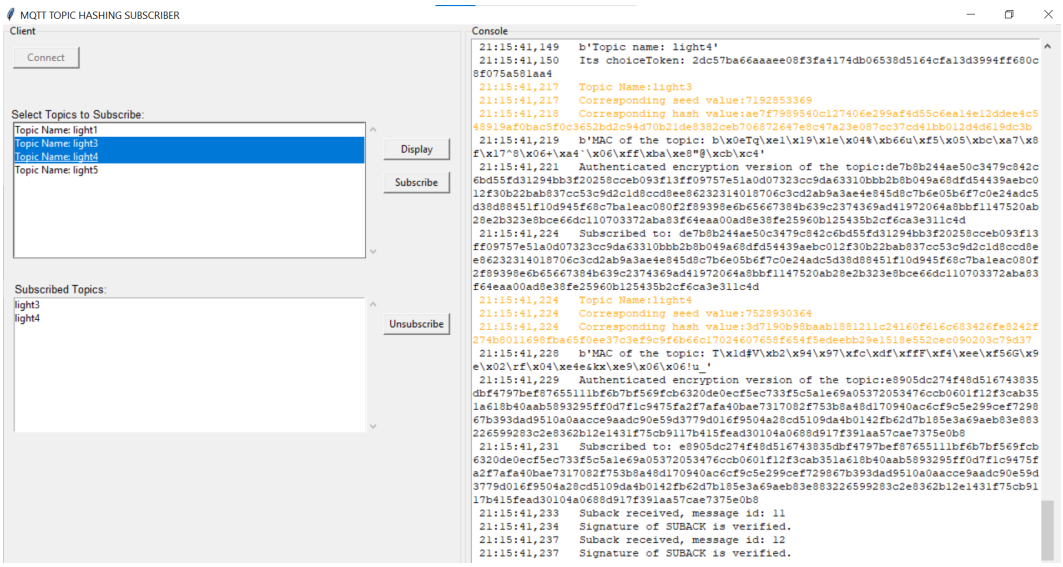

Figure 51

Topic hashing subscriber subscribes to topics 'light3' and 'light4'

In Figure 51, the topic hashing subscriber selects the topics 'light3' and 'light4' from the list and pushes the 'Subscribe' button in order to subscribe to these topics. First, it gets the choice tokens of 'light3' and 'light4' from the broker. Output of this process is not shown in Figure 51 since it is the known choice token taking process. Then the client subscribes to the authenticated encryption version of the first hash values of the topics 'light3' and 'light4'.

```
 21:16:39,941   A publish message send to the 'topicHashing' topic to start the hash
session.
 21:16:39,942   Message:tick
 21:16:39,942   Authenticated encryption version of the topic name to be published: 3
b2ea24f0a3feb946eed1979f8577bc60a39d681e73675d045723401368 73d93319d8acd6dac93d2897228
04f2b097d8b2dcb36759b254f2800bea85c6229a5e
 21:16:39,947   b'Message after encryption with the choice token: \xe2=\xle\x06\x18\x
81\x17\xfc\xd6\xff\x96\xceV\x18\xfcI'
 21:16:39,947   b'Message after authenticated encyption with the session key: \xef\xa
00,\xe2yi\x95kKq\xbfV\x94\x90\x9e\xd5f\xb0\x85\xfdo\xce\x12\t$G\xb2\x9e\xa6\xf1\xb5\x
c5\xf5\r\xelQ\xbf.l"\x89+\x06\xff\xfd\xebn\x8c\xcf\xab\xa8a5\xe7C\x90\x18h\x9cgV(\xla
'
 21:16:39,959   Puback was received, messageID =13
 21:16:39,960   Signature of PUBACK is verified.
```

Figure 52

Topic hashing publisher after pushing the 'Start Hash Session' button

In order to start the hash session, the topic hash publisher should push the ‘Start Hash Session’ button which can be seen in Figure 48. When the publisher pushes that button, a ‘tick’ message will be sent to the subscribers who subscribed to ‘topicHashing’ topic to inform the subscribers that the topic hashing session started. The publisher after pushing the ‘Start Hash Session’ button is shown in Figure 52.

```
21:16:39,991   ---Publish message was received from broker
21:16:39,992   Encrypted topic: 291de48ee47a6ead9b9578d920a7f10d2652c6e09a0beeba51e9
adeb96abld00c1d460b649e4726a3f6fc2869f182f5a2dc39c4058538d25bd6b813e79214513
21:16:39,993   b'Encrypted payload: g\xdfE\xc3E.9\xf6\xeb\x01\xb8\xde\xeb\x06\x93[\x
98\t\xecV\x95%\xdfUX\x9b\xa6\x88[s\xe2\x80\x969\xeeSwx<\xfc=\x9d\x17\xac\xbe\x17\xb4\
xf4u\x9f,\xbf+\xb9\xdf\x87t\x87sR\xdc\xd3-K'
21:16:39,999   b'Decrypted topic name: topicHashing and its mac: E?SD\x87b\xb9\x9fh\
x7f\xd2\x8c\xb7+\x1e\xe6\x1d\xc2^p\xd5\xe4[\x97\xe9\xc0\xef\x18uy}\xb1'
21:16:40,001   Choice token of the topic: e67bb2d02344c556773dfb6876e14b0331302418b9
d0c70a8542286217f8f6a1
21:16:40,004   The content of the topic name is not changed. Mac of the topic name i
s correct.
21:16:40,006   Topic name: topicHashing
21:16:40,007   b'Message after decryption with session key: \xe2=\x1e\x06\x18\x81\x1
7\xfc\xd6\xff\x96\xceV\x18\xfcI'
21:16:40,008   The content of the payload is not changed, Mac of the payload is corr
ect.
21:16:40,008   b'Message after decryption with choice token: tick'
```

Figure 53

Topic hashing subscriber when getting the ‘tick’ message

Figure 53 shows a subscriber who gets this ‘tick’ message. After the topic hashing publisher pushes the ‘Start Hash Session’ button, subscribers cannot subscribe to new topics during the hash session.

```
21:16:40,008   b'Message after decryption with choice token: tick'
21:17:38,051   Hash Session start already, you have to wait the next session to subs
cribe
```

Figure 54

Topic hashing subscriber when tries to subscribe after getting the ‘tick’ message

. In Figure 54, the subscriber tries to subscribe to a topic after it receives the ‘tick’ message from the ‘topicHashing’ topic. The subscription trials of the subscriber are blocked and subscribers are informed about the situation which can be seen in Figure 54.

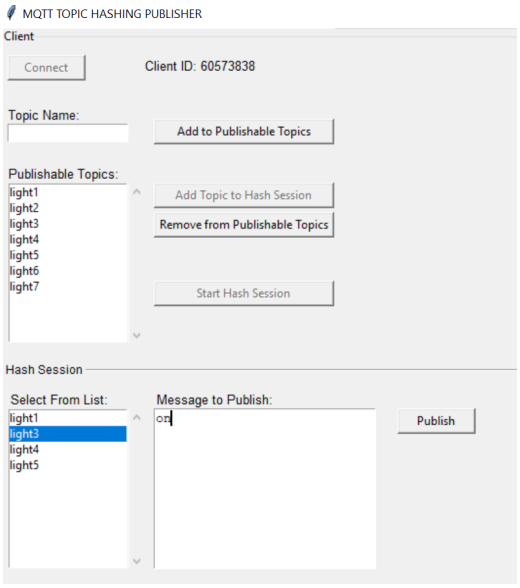

Figure 55

Topic hashing Publisher GUI when publishing to a topic

In Figure 55, topic hashing publisher wants to publish on the topic ‘light3’. When it pushes the ‘Publish’ button, random messages for the ‘light1’, ‘light4’ and ‘light5’ and the real message for the ‘light3’ will be encrypted with the choice token of the topic ‘light3’. Then, the related HMAC values are calculated for each packet and added at the end of the messages. After that, the messages will be encrypted with the session key between the broker and the publisher. The first hash values of the topics will be used as the corresponding topic names. Hash values

and its HMAC will be encrypted using the session key between the broker and the publisher. At the end of the Figure, these publish messages will be sent to the broker.

Figure 56

Topic hashing Publisher when it publishes to a topic

Figure 56 shows the topic hashing publisher during the publishing process. As can be seen in the Figure 56, the first hash value of the topic ‘light3’ is used as the topic name and its message is encrypted using the session key of the ‘light3’. Also, Figure 56 shows the encryption process of the ‘light1’. As can be seen, the choice token that is used to encrypt the messages for the ‘light3’ and the ‘light1’ is the same. ‘light1’ uses the choice token of ‘light3’ since ‘light3’ is the real topic in this iteration. Since the other topics ‘light4’ and ‘light5’ are also prepared similar to ‘light1’, their output is not added.

```
21:18:25,298   Salt Polynomial: 1018, and counter value: 1
21:18:25,300   Topic names: light1, and its old hash value: 43492eb90abf20c45c021f4a
cbf6efle979c0b1178408c6728d9c4f76498e33a89af7e04b5d4bac64e376f464ef5a3752c69d63f6198c
8e0a6a6bdaf42c80d98
21:18:25,320   Puback was received, messageID =16
21:18:25,322   Signature of PUBACK is verified.
21:18:25,370   Puback was received, messageID =17
21:18:25,371   Signature of PUBACK is verified.
21:18:25,578   Topic names: light1, and its new hash value: 0e2a16cae0f611f18dc0b703
7db8ea5a0d1f1080253387cdb353d92871043ddfeeab131fdf7929037a2c8c57e157268d3a12c67ee2713
67ce3c4cb4481f60391
21:18:25,578   Topic names: light3, and its old hash value: ae7f7989540c127406e299af
4d55c6ea14e12ddee4c548919af0bac5f0c3652bd2c94d70b21de8382ceb706872647e8c47a23e087cc37
cd41bb012d4d619dc3b
21:18:25,718   Topic names: light3, and its new hash value: a7d124fa42acele89abb64f8
e2042ad127164d97f5185d716f6b81bdbc1bb40e9374a007dd01b3a1a51e81eaela877d98d899979e7695
b7cea598a5e878b4bd3
21:18:25,724   Topic names: light4, and its old hash value: 3d7190b98baab1881211c241
60f616c683426fe8242f274b8011698fba65f0ee37c3ef9c9f6b66c17024607658f654f5edeebb29e1518
e552cec090203c79d37
21:18:25,937   Topic names: light4, and its new hash value: 37abddc572976e322892afd6
2dc36bed9876dd825daf64a0fea916a49efb3cace95aaf0d0f995c8adb2293ec34085ae951879bfa0da5e
14b2b4fe0a0b1b7ecce
21:18:25,937   Topic names: light5, and its old hash value: 9adbca9d769a345e2339509e
48f90bcb4f4d53fe631bf772d829e8c417d1acabdacc40fd5b183c29e99223cb6954195d2035720c04c13
4ab4c5b80a194522545
21:18:26,112   Topic names: light5, and its new hash value: 2d384030eb85a04e1f000e52
d158ad000a592461af244f22ff29e4ca2ce5066f4a83a915e5bc4d257518b4e555702777f3446f1d400ef
08e6c8535257c368cab
```

Figure 57

Topic hashing Publisher when calculating the new hash values

In figure 57, topic hashing publisher calculates the next hash values for the topic names.

```
21:18:25,279   ---Publish Message was received from Broker
21:18:25,279   Encrypted topic: de7b8b244ae50c3479c842c6bd55fd31294bb3f20258cceb093f
13ff09757e51a0d07323cc9da63310bbb2b8b049a68dfd54439aebc012f30b22bab837cc53c9d2c1d8ccd
8ee86232314018706c3cd2ab9a3ae4e845d8c7b6e05b6f7c0e24adc5d38d88451f10d945f68c7baleac08
0f2f89398e6b65667384b639c2374369ad9ff5d58f9af211e9e75ebb6dbced7a4a7befd72a830161aa839
0308760867393d03c9566fa2352394bf49ff4f62b429e
21:18:25,289   b"Encrypted payload: \xd1\x1a)\xb9\xec\x0b\x9c\x0f\x8e0\xa3s\x04\x80&
\x98\x97\x05\x8aG\xe2\x15XU\xd7\xd0b=\xc8\xeab'\xfd\n\xaf1\xfbQR-\xd3\xa6\xcc-\xc7\xc
cS\r\xf0]\x96/[\xad\xe8\r\x81\x04M\x80J;(S"
21:18:25,297   Real topic name: light3
21:18:25,299   Its corresponding hash value: ae7f7989540c127406e299af4d55c6ea14e12dd
ee4c548919af0bac5f0c3652bd2c94d70b21de8382ceb706872647e8c47a23e087cc37cd41bb012d4d619
dc3b
21:18:25,301   The content of the topic name is not changed. Mac of the topic name i
s correct.
21:18:25,302   Salt polynomial: 1018, and counter value: 1 and its topic name:light3
21:18:25,620   Real topic name: light3 and its next hash value: a7d124fa42acele89abb
64f8e2042ad127164d97f5185d716f6b81bdbc1bb40e9374a007dd01b3a1a51e81eaela877d98d899979e
7695b7cea598a5e878b4bd3
21:18:25,620   Authenticated encryption version of the next hash topic: 32f489361bbc
735a71456ef13d4b0499f404c3bb9e1932fbf30c0d8619d0e00e5ac8b1a67fdd5a9206c8fc4d4458fff2c
f3b2647869131cdb1492b41d5f9884f0ec8914bbdc7cfef274b9c9131adc130da8ad8611db6b866c8c616
bba4843f25d6203f4cb45734c033ece5d0862590275f5517d3c2a69a26a003bf53638577523226d81fa52
887ba69f2cb6c04d859a226ac61f1736082f101f57377d22a6aff2b8bc3d0369981b485efb716b3b20ea8
21:18:25,620   Subscribed to authenticated encryption version of the next hash value
: 32f489361bbc735a71456ef13d4b0499f404c3bb9e1932fbf30c0d8619d0e00e5ac8b1a67fdd5a9206c
8fc4d4458fff2cf3b2647869131cdb1492b41d5f9884f0ec8914bbdc7cfef274b9c9131adc130da8ad861
1db6b866c8c616bba4843f25d6203f4cb45734c033ece5d0862590275f5517d3c2a69a26a003bf5363857
7523226d81fa52887ba69f2cb6c04d859a226ac61f1736082f101f57377d22a6aff2b8bc3d0369981b485
efb716b3b20ea8
21:18:25,620   Unsubscribed to authenticated encryption version of the current hash
value: de7b8b244ae50c3479c842c6bd55fd31294bb3f20258cceb093f13ff09757e51a0d07323cc9da6
3310bbb2b8b049a68dfd54439aebc012f30b22bab837cc53c9d2c1d8ccd8ee86232314018706c3cd2ab9a
3ae4e845d8c7b6e05b6f7c0e24adc5d38d88451f10d945f68c7baleac080f2f89398e6b65667384b639c2
374369adaeccd9b5bfd70547adcc59f213ce4dbab74406b58a91d9093d458bbd01e36ed364484e9229fd7
d499596affb2d4b5bbb
21:18:25,628   MESSAGE IS DECRYPTED WITH THE CHOICE TOKEN
21:18:25,629   The content of the payload is not changed, Mac of the payload is corr
ect.
21:18:25,629   b'Message after decryption with choice token: on\n'
```

Figure 58

Topic hashing Subscriber when gets the Publish messages

Figure 58 shows the topic hashing subscriber when gets the publish messages. This subscriber gets messages from ‘light3’ and ‘light4’ since it subscribed to these topics in Figure 51. As can be seen in the Figure 58, when the subscriber gets the publish packet first decrypts the topic name. Then it looks at the dictionary to find out the name of the topic name, which is the equivalent to the received hash value. After that, it will calculate the next polynomial value and the next hash value that will be used for the ‘light3’ and subscribes to the authenticated encryption version of that hash value. After the subscription to the new hash value, the unsubscribe packet is sent to the previous hash value of that topic. At the end of Figure 58, the client can decrypt the message using the choice token of the ‘light3’ since it is the real topic the publisher publishes in this iteration.

Figure 59

Topic hashing Subscriber when gets the Publish messages

Figure 59 shows the time when publish message is received for the ‘light4’. The same operations as Figure 58 are done for the ‘light4’ in Figure 59. The only difference is the last part of Figure 59. The client tries to decrypt the message using the choice token of the ‘light4’ but it cannot decrypt it since it is encrypted with the choice token of the ‘light3’.

Figure 56 and Figure 57 contain the first iterations from the publisher side. Figure 58 and Figure 59 contain the first iterations from the subscriber side. The second iteration starts when the publisher publishes on a new topic. These iterations continue until the counter reaches a predefined maximum value on the publisher side. When this counter ends, everything will reset. If the publisher wants to start another hash session, it can do that by selecting topics and pushing the ‘Add to Hash Session’ button again and everything will start from the beginning with new seeds and polynomial values.

4.3. Modules Used for the Implementation Process

AMQTT broker is used as the broker which is implemented with Python programming language. Eclipse Paho MQTT client library is used as the client library which is implemented with Python programming language. For generating the X.509 public key certificates the “X509” from the cryptography library was used (“X.509”, n.d.) for the both broker and client sides. For generating a common shared key using Diffie Hellman key exchange protocol “DiffieHellman” from the py-diffie-hellman library was used (“*Py-diffie-hellman*”, n.d.). For RSAsign of the Diffie Hellman public key “rsa” from the cryptography library was used (“RSA”, n.d.). For generating the 32 byte (256 bit) session from the Diffie Hellman common shared key ‘base64’

library of Python was used (“Base-64”, n.d.). For the encryption purposes AES256 function of the cryptography library was used (“Symmetric encryption”, n.d.). For nonce generation ‘token_urlsafe()’ function of the ‘secrets’ library of Python was used (“Secrets”, n.d.). For providing the integrity of messages, MAC generation was achieved by using ‘HMAC’ function of the cryptography library (“Hash-based message authentication codes (HMAC) ”, n.d.). For topic based choice token generation, token_hex()’ function of the ‘secrets’ library of Python was used (“Secrets”, n.d.). For the topic hashing scheme, pbkdf2_hmac() function of the ‘hashlib’ was used (“hashlib”, n.d.).

4.4. Performance Testing

During the performance testing, connect, subscribe, unsubscribe, publish, receive publish and publish seed operations have been focused on by comparing the default client module, the authenticated client module and the topic hashing client module (subscriber and publisher client), the default broker and the authenticated broker, for different number of topics were applicable. Main focus was on connect, subscribe, publish, unsubscribe and receive publish from a subscribed topic because these operations are milestones of the implementation of the project. Throughout the connection, these operations are frequently used. Also for the topic hashing clients, publish seed operation is considered. Unfortunately, there exists no data to be compared for the publish seed operation since the default version or the normal authenticated version do not consist of such operation.

A default client (non-updated version of Paho client without any encryption, hashing, etc.) has been used to measure the initial performance and an authenticated client (a client who uses all of the schemes introduced except the topic hashing scheme) has been used later on to

measure the current performance after the changes, for the operations connect, publish, subscribe, unsubscribe, publish and receive publish. A default broker (non updated version of the aMQTT broker) has been used to measure the initial performance and an authenticated broker (modified version of the aMQTT broker) has been used later on to measure the current performance of the operations as connect, subscribe, unsubscribe and publish. An authenticated topic hashing subscriber client has been used to measure connect, subscribe, unsubscribe operations and an authenticated topic hashing publisher client has been used to measure connect, publish and seed publish operations.

For subscribe and unsubscribe operation, different numbers of topics have been tried out to gain insight according to the changes in the performance in busier communications since a client can subscribe to multiple topics using only one option in the GUI and unsubscribe from multiple topics at the same time, using only one option in the GUI. Also for the topic hashing publisher client, publish and seed publish operations have been tried out with different numbers of topics because the client can publish to different numbers of topics and publish seed for different number of topics, depending on the number of topics selected for that hash session, making such comparisons needed.

Histograms, box plots (if needed) and scatter plots for comparison have been used along with standard summary metrics (mean, median, standard deviation, min and max) to present the results. Also, increases in mean of the observed durations of the operations have been compared with the initial mean values in order to interpret the performance of each operation separately. For the operations tried out with different numbers of topics, observed values of different numbers of topics have also been compared.

4.4.1. Performance of Clients

The information regarding the properties of the machines used during performance testing of the clients are provided below.

Default Client, Normal Authenticated Client and Authenticated Topic Hashing Subscriber Client:

- Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz 2.50 GHz
- 16 GB RAM
- 256 GB SSD
- Windows 10 Home Single Language 22H2

Authenticated Topic Hashing Publisher Client:

- Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz
- 16 GB RAM
- 1 TB SSD
- Windows 10 Home Single Language 22H2

4.4.1.1. Performance of Connect Operation

Since the connect operation is necessary before performing any other operation, more samples have been collected to achieve more accurate results. 100 default client instances and 100 authenticated client instances have been observed and the time spent during the connect operation have been measured by calculating the difference of start time and the end time of the operation. In terms of the default client, connect operation consists of preparing and sending a CONNECT package. But the process is more time consuming and complex for the authenticated

client module. As explained earlier, “connect” operation of the authenticated client consists of Ephemeral DH key exchange protocol to establish a unique session key between the client and the broker to be used in the latter stages of the connection. Figure 1 shows the details of this operation in detail. The time spent is measured by calculating the difference between start time (just before sending the CONNECT packet) and receiving the last publish message of the scheme. Also, time spent on connect operation has been measured for the topic hashing clients, subscriber and publisher separately, with a total of 100 instances. Since those clients join hash sessions that use topic hashing for all of the topics subscribed/published to, those clients have extra steps to perform the connect operation, compared to a non topic hashing authenticated client.

Dataset	no of observations	mean	std	median	min	max	auth/default ratio	ratio (percent)
Normal Auth Connect Testing	100	0.7166	0.3199	0.6226	0.4959	1.9540	3.96	396%
Default Connect Testing	100	0.1809	0.2106	0.1381	0.1118	1.2625		
Topic Hashing Auth Subscriber Connect Testing	50	1.5439	0.6372	1.2739	1.0035	4.0182	8.5326	853%
Topic Hashing Auth Publisher Connect Testing	50	1.4900	0.0828	1.4765	1.3757	1.8006	8.2344	823%

Figure 60

Summary Metrics of Connect Tests with Default and Authenticated Client (Topic Hashing and Not Topic Hashing) Instances

Figure 60 shows the summary metrics of the connect operation for client instances as default client and authenticated client, also the topic hashing clients as subscriber and publisher. Standard deviation has been observed higher in authenticated client modules compared to default client modules. Figure 61 shows histograms of authenticated client, default client, topic hashing subscriber client and topic hashing publisher client modules and it can be said that distributions are not close to standard normal distribution and there exists skewness in all results.

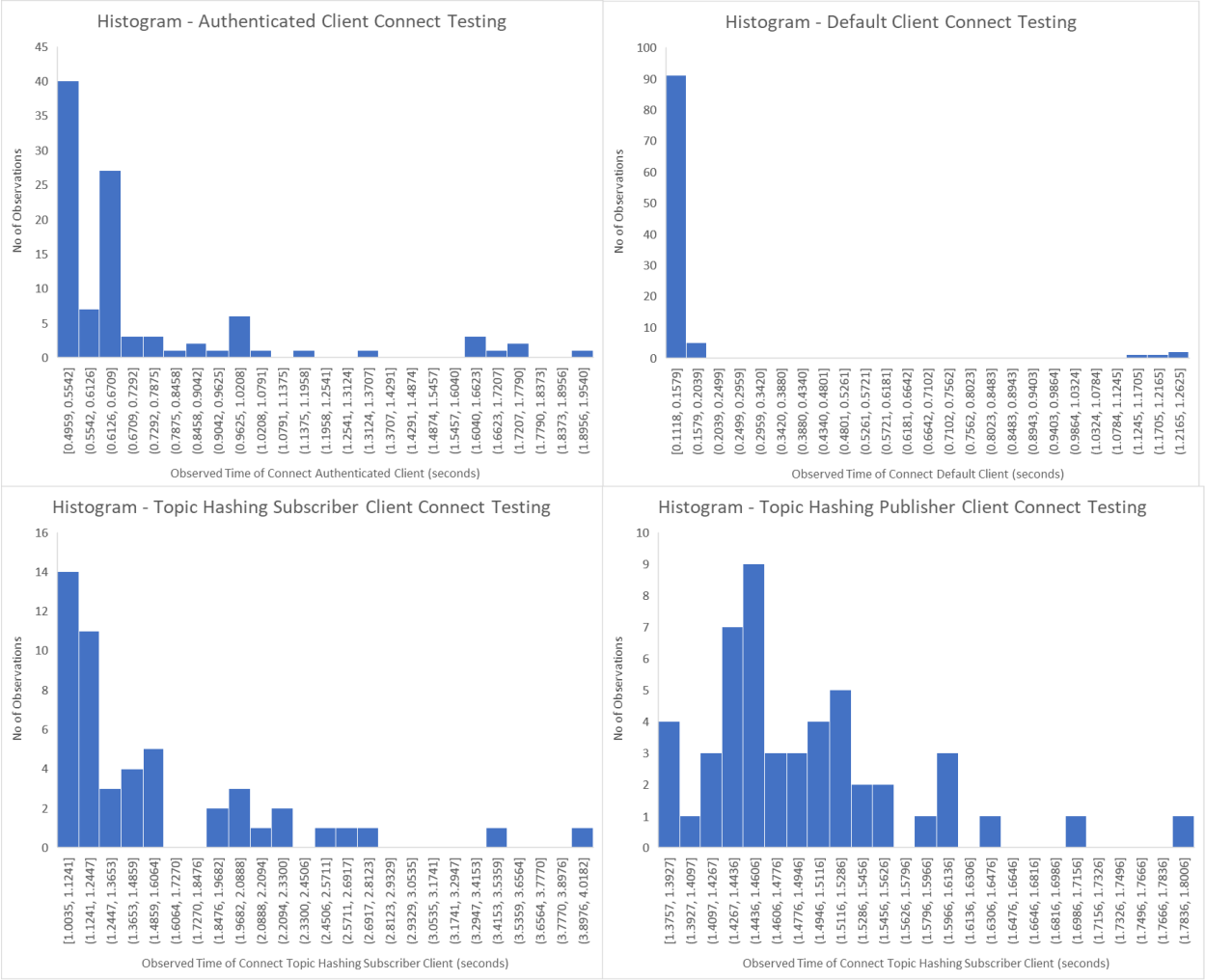

Figure 61

Histograms of Authenticated Client Connect Testing, Default Client Connect Testing, Topic Hashing Subscriber Client and Topic Hashing Publisher Client, All Having Number of Bins Equal to 25

Since a high difference has been observed between minimum and maximum values, boxplots have been drawn to determine the outliers, shown in Figure 62. The extreme values can be observed in the box plots.

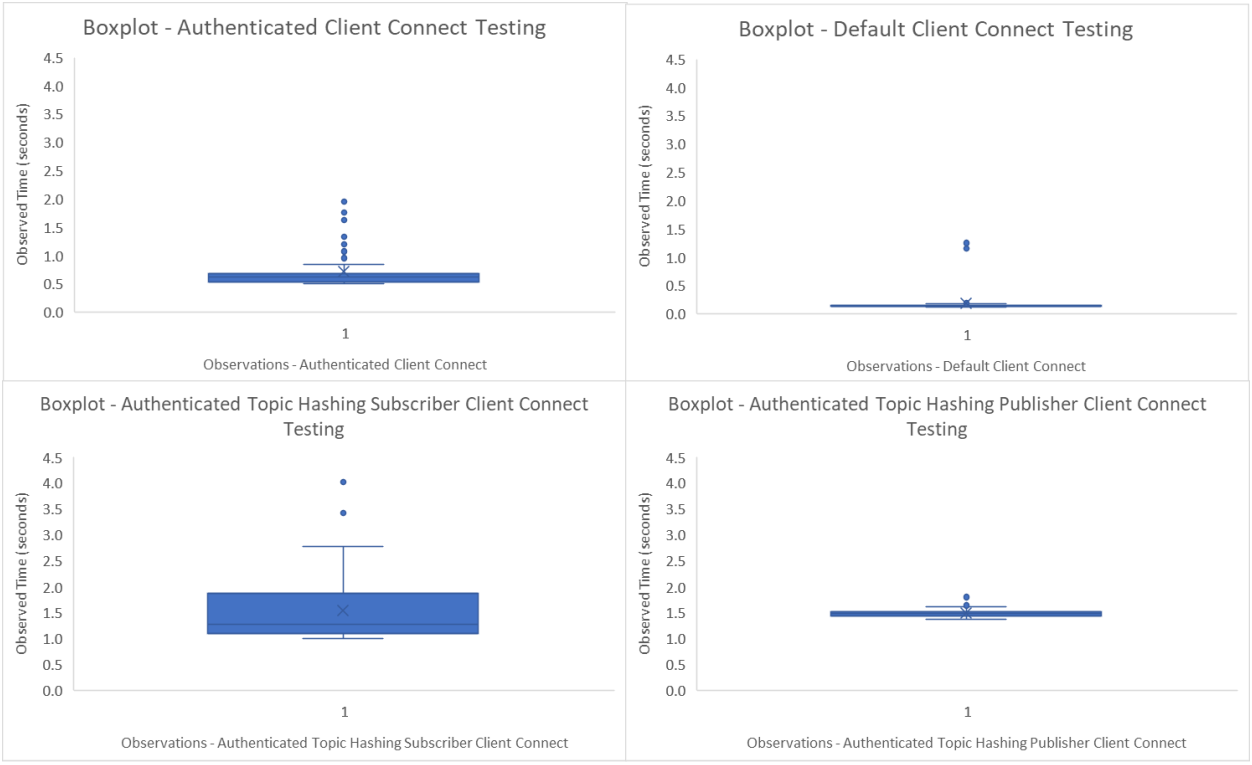

Figure 62

Box Plots of Connect Testing with Different Clients, Measurements Ranging from 0 to 4.5 Seconds

In terms of average performance, the default client has been observed as 0.1802 seconds and the authenticated client has been observed as 0.7116 seconds, causing nearly 3 times increase in the duration of the connection. For the topic hashing subscriber client, measurement is found as 1.5439 seconds on average and for the topic hashing publisher, the measurement is found as 1.49. These results lead to ratios with the default as 8.5326 and 8.2344. Figure 63 consists of a comparison between the observations of the 4 different client modules. There is a clear difference between the default and the authenticated client module. This is an expected result considering the number of total packets exchanged during the Ephemeral DH exchange protocol (10 packets, shown in Figure 1) vs the number of packets exchanged during the default

connection. As a conclusion, these results for the connection operation (comparing the default and the authenticated client) can be accepted excluding the extreme outliers.

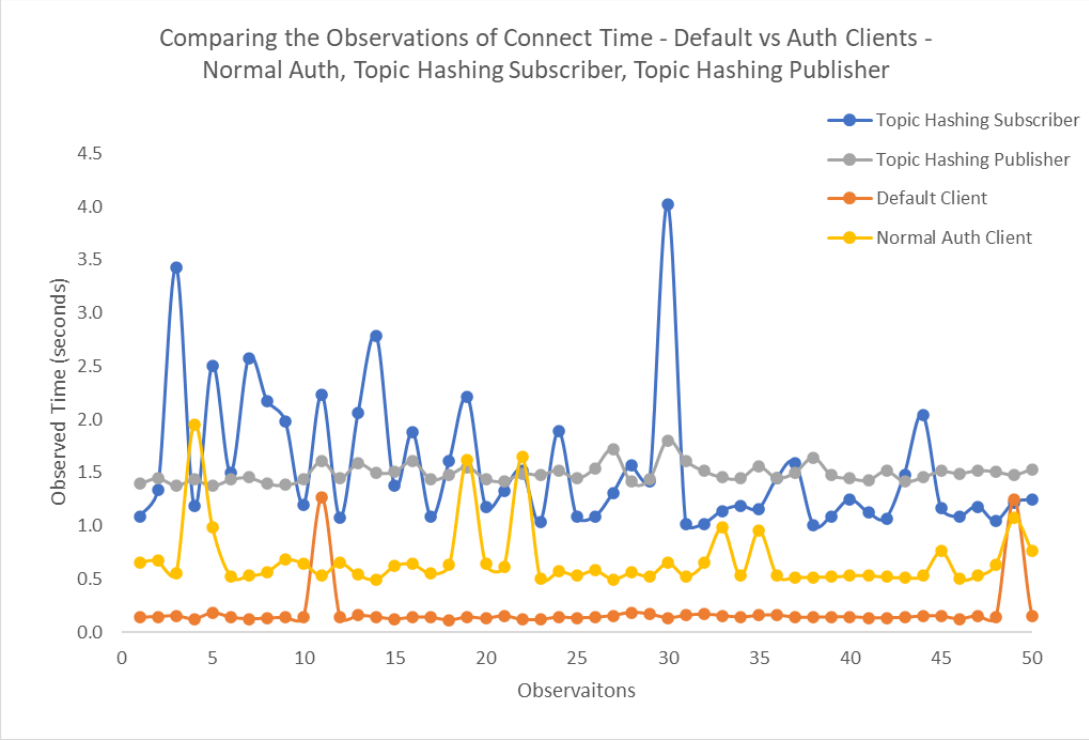

Figure 63

Plot Consisting of Four Series for Authenticated, Default, Topic Hashing Subscriber and Publisher Client

For the topic hashing case, it is clear that topic hashing reduces the performance of the connect operation and the measurements are approximately two times of the normal authenticated client's measurements in seconds. Topic hashing client's connection operations have extra steps compared to authenticated clients so a difference in terms of duration was expected. As a conclusion, using the normal authenticated client is much better and if not needed or requested specifically, topic hashing subscriber and publisher should not be selected.

4.4.1.2. Performance of Subscribe Operation

For comparing the performances of default and authenticated client modules regarding the subscribe operation, 50 instances for each number of topics (1, 5, 10, 20) for each client type have been created, and a total number of 400 instances have been used. For the default client, time is measured as total time spent for preparing and sending n subscribe packets for n topics. For the authenticated client, there are extra operations to be handled and included in the subscription operation, shown in Figure 2 and Figure 3. All the steps in Figure 2 should be followed and after that, the first package in Figure 3 should be sent by the client in order to subscribe to a topic. These operations should be followed n times for n topics. Figure 64 below shows the histograms of default client instances subscribing to 1, 5, 10 and 20 topics.

An extra measurement has been made for the subscribe operation of topic hashing subscriber client with 1 topic. This measurement is not categorized as the same as the normal authenticated client because the process is different compared to a normal authenticated client's subscription, even though both processes consist of common steps like publishing for choice tokens, etc. For the subscription of topic hashing subscriber, 50 client instances have been used to measure the duration.

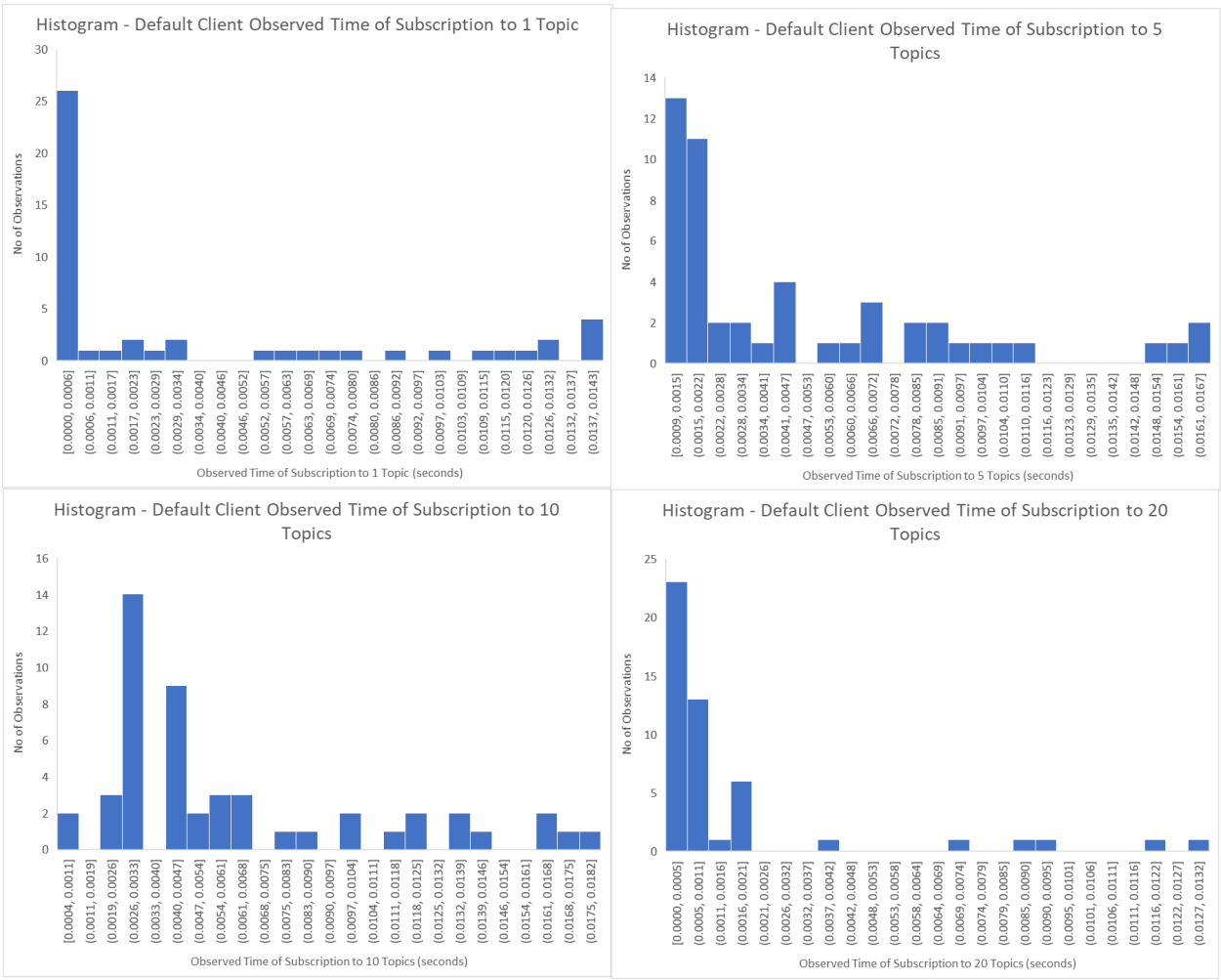

Figure 64

Histograms for Default Client Instances' Subscribe Operations Regrading 1, 5, 10, 20 Topics

Figure 65 shows the histograms of authenticated client instances subscribing to 1, 5, 10 and 20 topics.

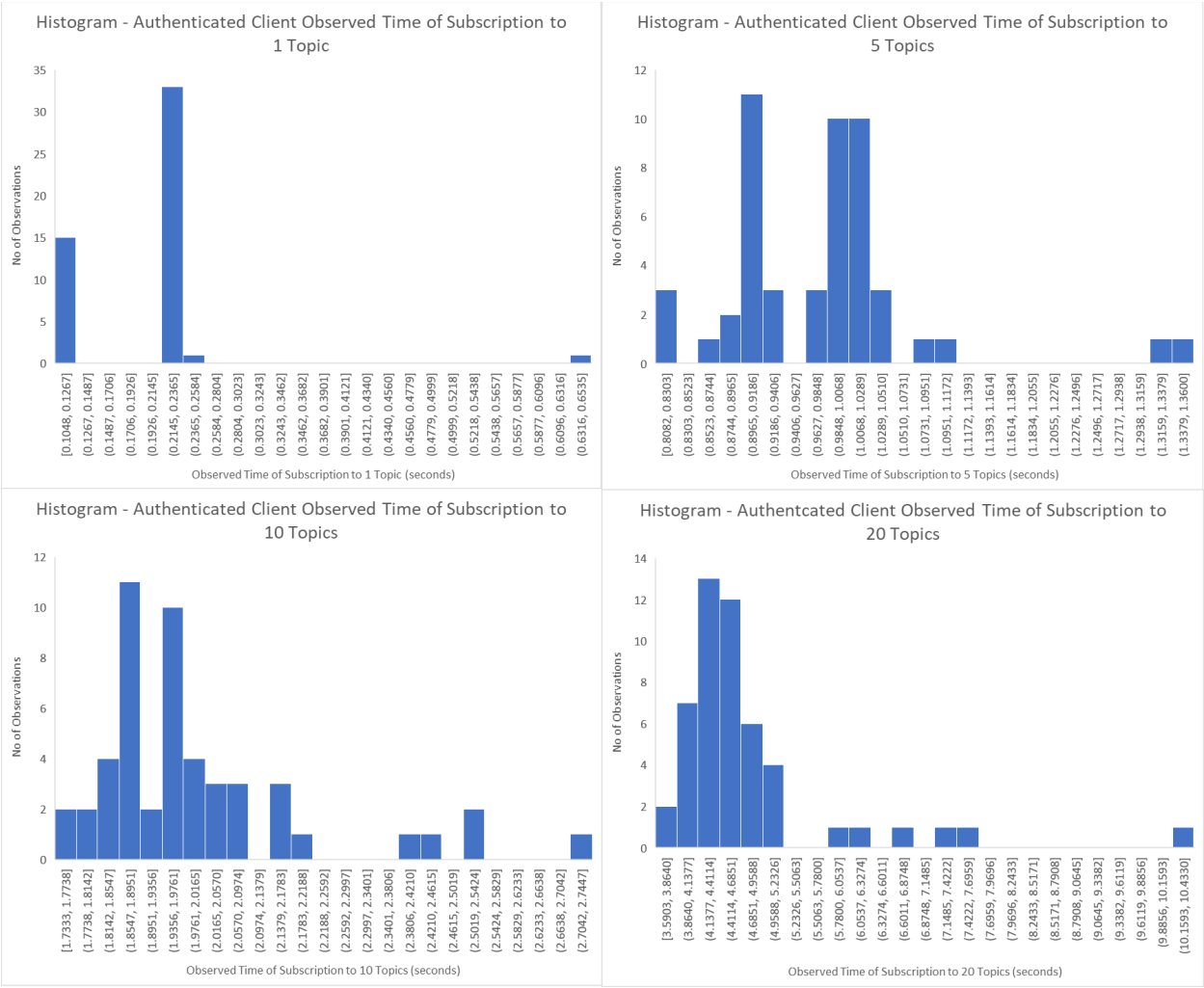

Figure 65

Histograms for Authenticated Client Instances' Subscribe Operations Regrading 1, 5, 10, 20 Topics

Again, it has been observed that there are no distributions close to standard normal distribution and mostly, there exists skewness (positive). Figure 66 below shows the histogram of topic hashing subscriber client instances subscribing to 1 topic. Again there exists extreme outliers in the distribution as well.

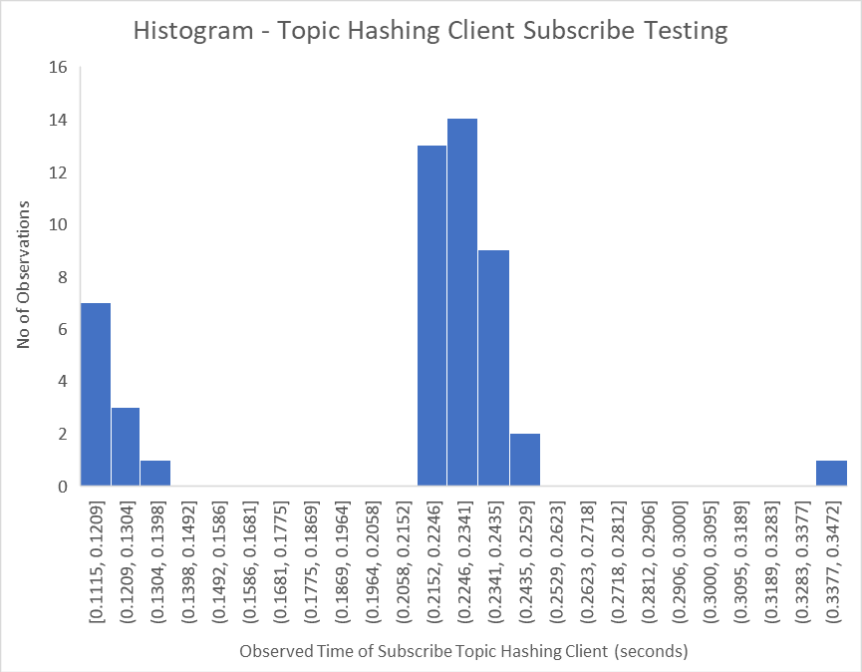

Figure 66

Histogram of Topic Hashing Client Instances' Subscribe Operation Regrading 1 Topic

Summary metrics for all 9 different datasets displayed in Figure 64, Figure 65 and Figure 66 can be seen in Figure 67.

Dataset	no of observations	mean	std	median	min	max	auth/default ratio	ratio (percent)
Default Subscription (1 topic)	50	0.0039	0.0052	0.0000	0.0000	0.0143	50.71	5071%
Authenticated Subscription (1 topic)	50	0.1995	0.0820	0.2198	0.1048	0.6535		
Default Subscription (5 topic)	50	0.0048	0.0045	0.0024	0.0009	0.0167	203.6892	20369%
Authenticated Subscription (5 topic)	50	0.9779	0.0997	0.9890	0.8082	1.3600		
Default Subscription (10 topic)	50	0.0063	0.0047	0.0040	0.0004	0.0182	318.7347	31873%
Authenticated Subscription (10 topic)	50	2.0003	0.2101	1.9571	1.7333	2.7447		
Default Subscription (20 topic)	50	0.0086	0.0037	0.0090	0.0000	0.0213	555.3973	55540%
Authenticated Subscription (20 topic)	50	4.7762	1.1391	4.4925	3.5903	10.4330		
Topic Hashing Subscription (1 topic)	50	0.2073	0.0502	0.2266	0.1115	0.3472	52.6859	5269%

Figure 67

Summary Metrics for Subscription Durations of Authenticated, Default and Topic Hashing Client Instances, Regarding 1, 5, 10, 20 Topics

In order to better interpret the results, performances of client instances for subscribing to 1, 5, 10 and 20 topics have been compared for both client types. Figure 68 shows two plots comparing the results for default, authenticated and topic hashing client instances. As it can be seen from the first plot in Figure 68, no consistent increase in durations for subscribing to more topics has been observed for the default client. But in authenticated client instances, there exists clear distinction between the durations of subscribing to 1, 5, 10 and 20 topics among each other.

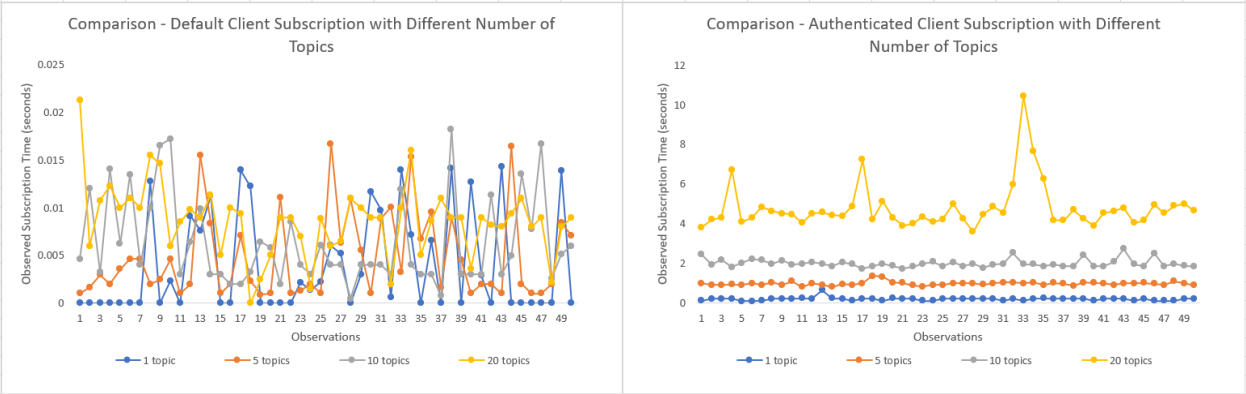

Figure 68

Two Plots Comparing Subscription to Different Number from Topics for Default and Authenticated Client Instances

In the observations of authenticated client instances, increase in duration is very distinct and clear. mean values for subscribing to 1, 5, 10, 20 topics are measured (shown in Figure 67) as follows: 0.1995, 0.9779, 2.0, 4.7762 (seconds). Those increases in durations are consistent since mean value of 5 topics is roughly 5 times of mean value of 1 topic, mean value of 10 topics is roughly 2 times of mean value of 5 topics, mean value of 10 topic is roughly 2 times of mean value of 5 topics and mean value of 20 topics is roughly 2 times of mean value of 10 topics. But when compared with the results of the default instances, these increases in durations can be

perceived as extreme. But considering the added steps for subscribing to a topic, these results can be partly tolerated. Figure 69 below shows the comparison of mean values of default client instances for different numbers of topics with mean values of authenticated client instances for different numbers of topics. By looking at Figure 69, the increase seems to be linear for the authenticated subscription time for increasing number of topics while the increase for default subscription time for different numbers of topics is almost zero.

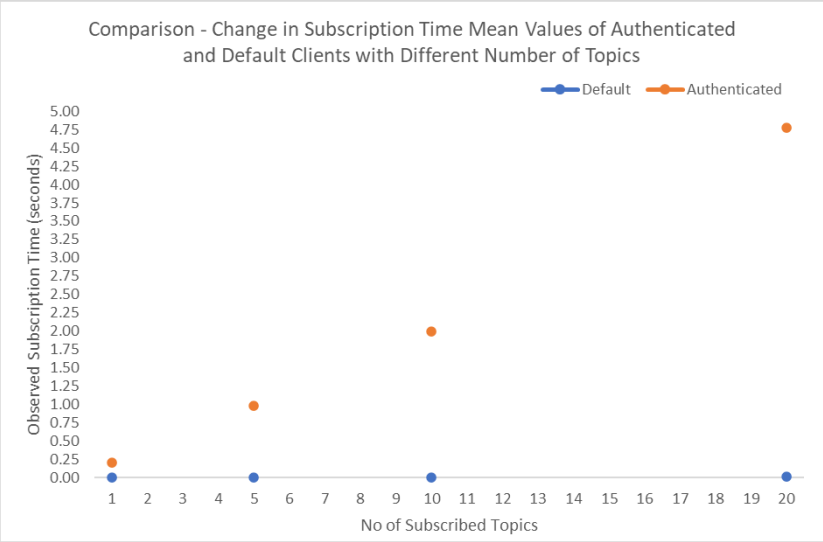

Figure 69

Comparison of Mean Values for Subscribing to Different Numbers of Topics for Authenticated and Default Client Instances

As a comparison of subscription to 1 topic among default, authenticated and topic hashing subscriber client modules, Figure 70 is provided below. As it can be seen in the figure, there is no clear difference between topic hashing subscription and authenticated subscription, A similar interpretation can be done by **Figure 67,** observed mean of authenticated client's

subscription is 0.1995 while observed mean of topic hashing subscriber client's subscription is 0.2073. This is a desired solution for the subscription.

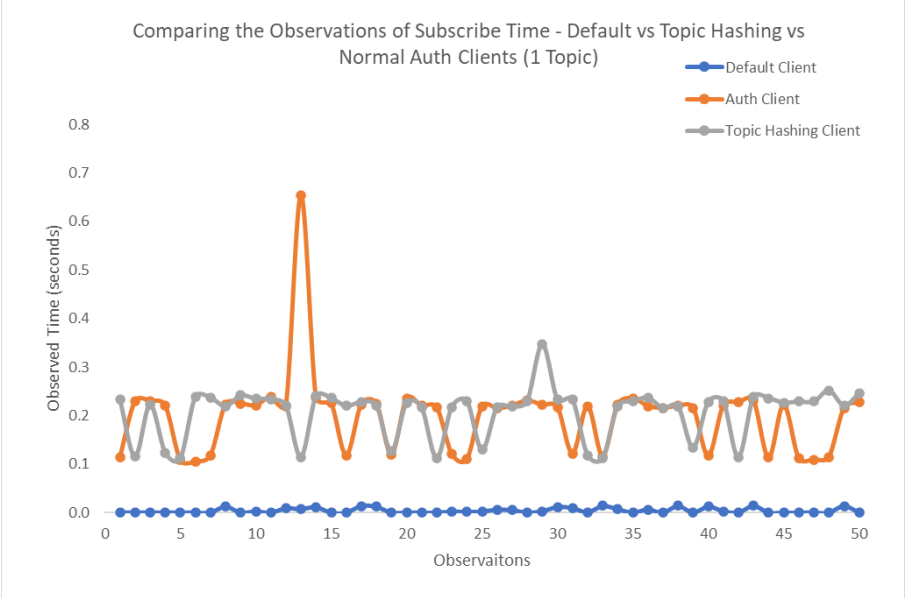

Figure 70

Comparison of Subscription to 1 Topic of Default, Authenticated and Topic Hashing Subscriber Client Modules

4.4.1.3. Performance of Unsubscribe Operation

Similar to subscribe operation, different numbers of topics (1, 5, 10, 20) have been tried out for default and client instances, for each case, 50 observations have been recorded, making a total of 400 instances of unsubscriptions. Since MQTT protocol can handle unsubscribing from more than one topic with only one unsubscribe packet, no distinct increase in duration of unsubscribe operation is expected for the authenticated client instances. For both authenticated and default clients, the time is measured as preparing an unsubscribe packet and sending it to the

broker. Of course, the preparation of an unsubscribe packet in an authenticated client module is different and consists of more operations, so it is logical to measure the difference either way.

As an extra measurement, duration of unsubscribe operation for the topic hashing subscriber with 1 topic has been observed with 50 client instances. Results received from this measurement have been used to compare 3 different client modules' durations of subscribe operations when the subscribed topic number is 1.

Figure 71 shows histograms for the default client instances unsubscribing from different numbers of topics (1, 5, 10 and 20) and Figure 72 shows histograms for the authenticated client instances unsubscribing from different numbers of topics (1, 5, 10 and 20). All histograms have a number of bins equal to 25 for consistency. Again, there exists outliers in some histograms and they are not very close to standard normal distribution.

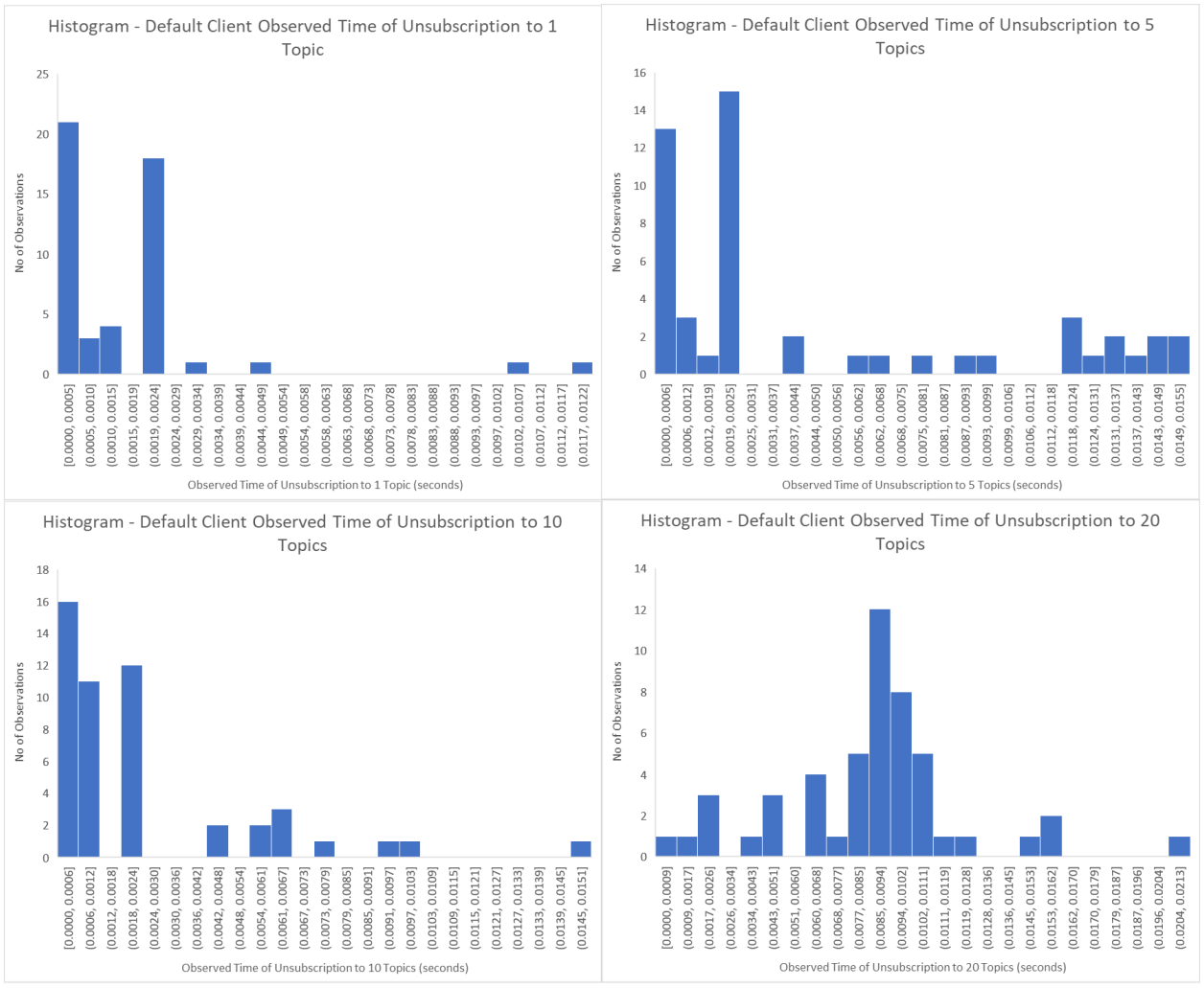

Figure 71

Histograms for Default Client Instances' Unsubscribe Operations Regrading 1, 5, 10, 20 Topics

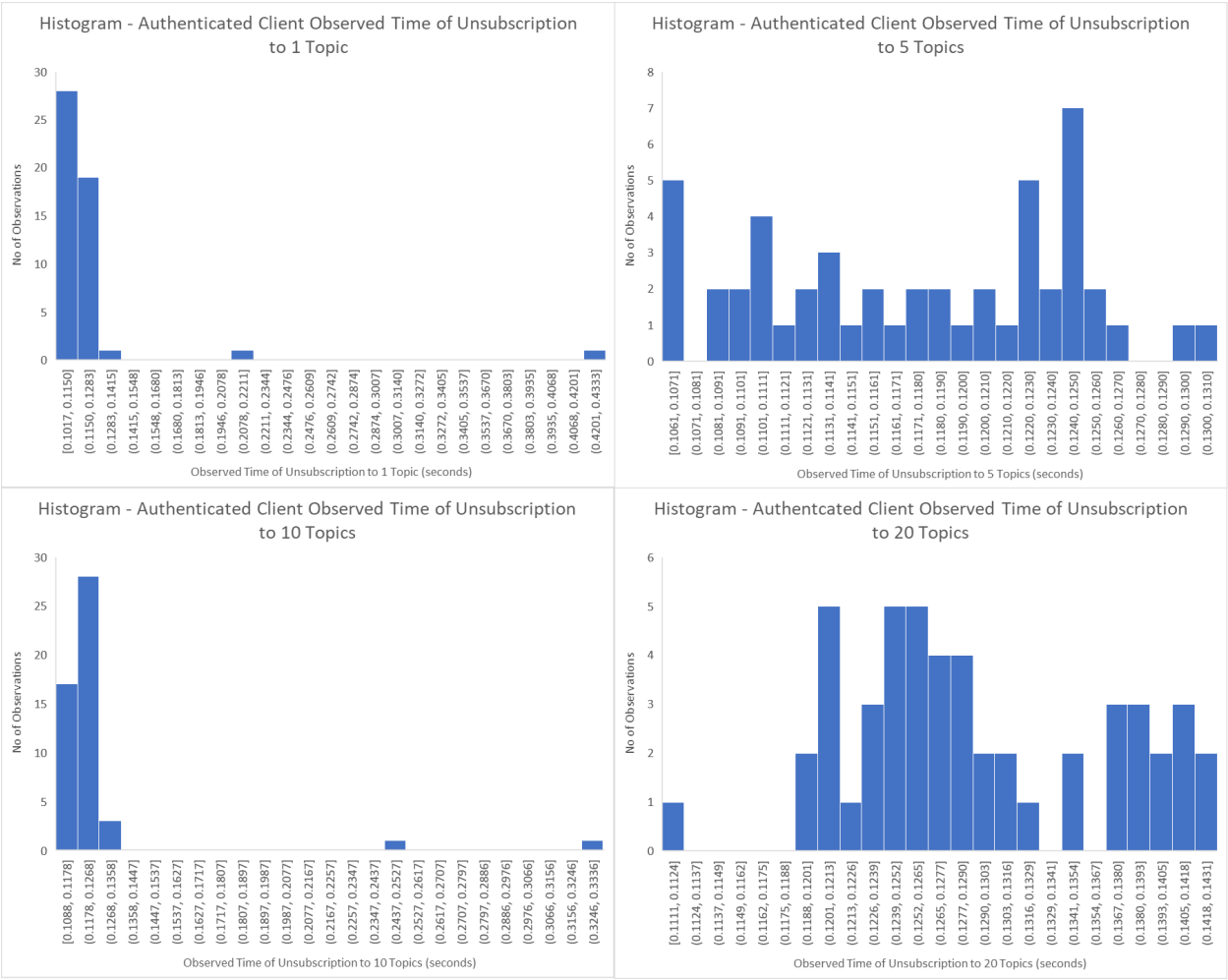

Figure 72

Histograms for Authenticated Client Instances' Unsubscribe Operations Regrading 1, 5, 10, 20 Topics

Figure 73 shows the histogram of unsubscribing from 1 topic for the topic hashing subscriber client. Most of the observations are gathered around the mean but there exists a small number of outliers which is expected for the testing.

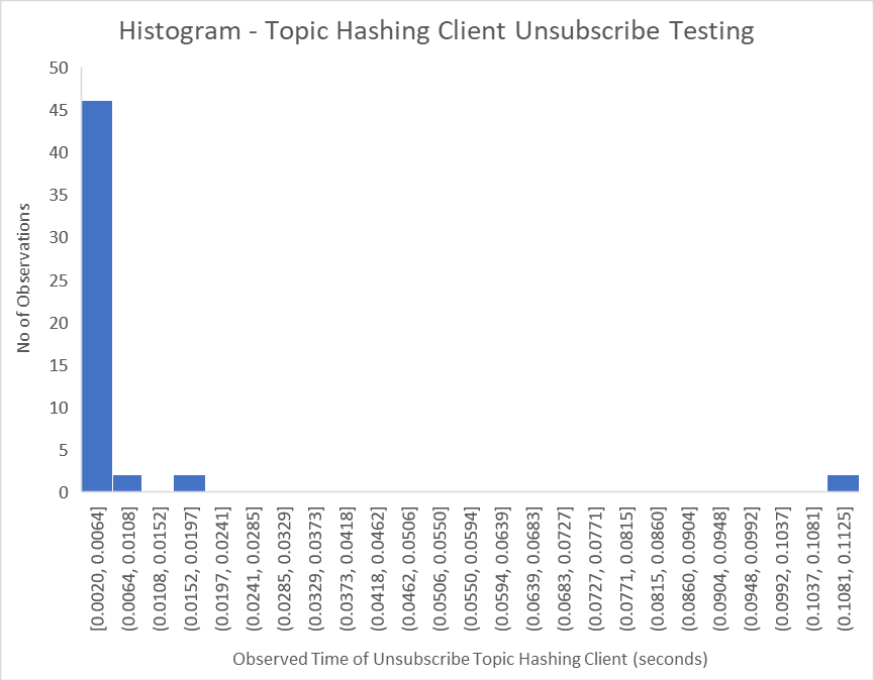

Figure 73

Histogram of Topic Hashing Client Instances' Unsubscribe Operation Regrading 1 Topic

Figure 74 shows the summary metrics of different cases for unsubscribe operation. Default client's mean values are not very consistent, this might be due to sending only one packet for any number of topics to be unsubscribed from. When authenticated clients' mean values are examined, it can be seen that values are close to each other.

Dataset	no of observations	mean	std	median	min	max	auth/default ratio	ratio (percent)
Default Unsubscription (1 topic)	50	0.0015	0.0023	0.0010	0.0000	0.0122	83.6182	8362%
Authenticated Unsubscription (1 topic)	50	0.1227	0.0471	0.1129	0.1017	0.4333		
Default Unsubscription (5 topic)	50	0.0046	0.0053	0.0020	0.0000	0.0155	25.3000	2530%
Authenticated Unsubscription (5 topic)	50	0.1175	0.0068	0.1183	0.1061	0.1310		
Default Unsubscription (10 topic)	50	0.0023	0.0031	0.0010	0.0000	0.0151	53.6411	5364%
Authenticated Unsubscription (10 topic)	50	0.1259	0.0354	0.1204	0.1088	0.3336		
Default Unsubscription (20 topic)	50	0.0019	0.0034	0.0007	0.0000	0.0132	69.1637	6916%
Authenticated Unsubscription (20 topic)	50	0.1291	0.0075	0.1272	0.1111	0.1431		
Topic Hashing Unsubscription (1 topic)	52	0.0078	0.0209	0.0030	0.0020	0.1125	5.3217	532%

Figure 74

Summary Metrics for Unsubscription Durations of Authenticated and Default Client Instances, Regarding 1, 5, 10, 20 Topics

Figure 75 below consists of comparison of unsubscribing from different numbers of topics for authenticated and default clients instances separately. As mentioned before, when compared for different types of client within themselves by only looking at topic number to be unsubscribed from, there is no clear increase in durations, both plots show the observations are not consistent enough to observe an increase. Especially for the default client, there are many outliers for unsubscribing from 1, 5, 10 and 20 topics. It is difficult to make an interpretation just by looking at the comparison of observations for the same client type.

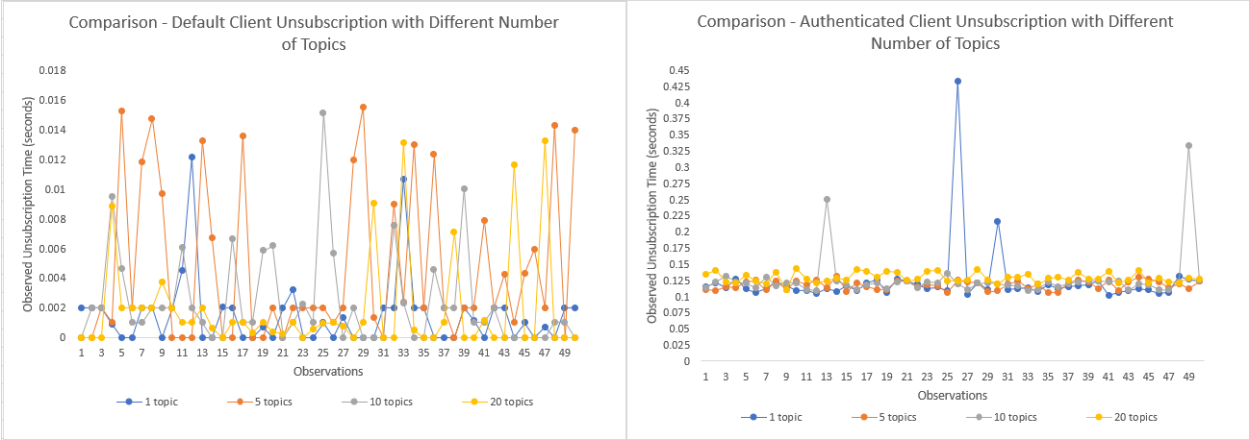

Figure 75

Two Plots Comparing Unsubscription from Different Number of Topics for Default and Authenticated Client Instances

When compared among the client types, there exists differences in durations which can also be understood from Figure 74 by the mean values of different client types. Figure 76 below consists of a comparison between two client types' (default and authenticated clients) measured durations of unsubscribing from different numbers of topics.

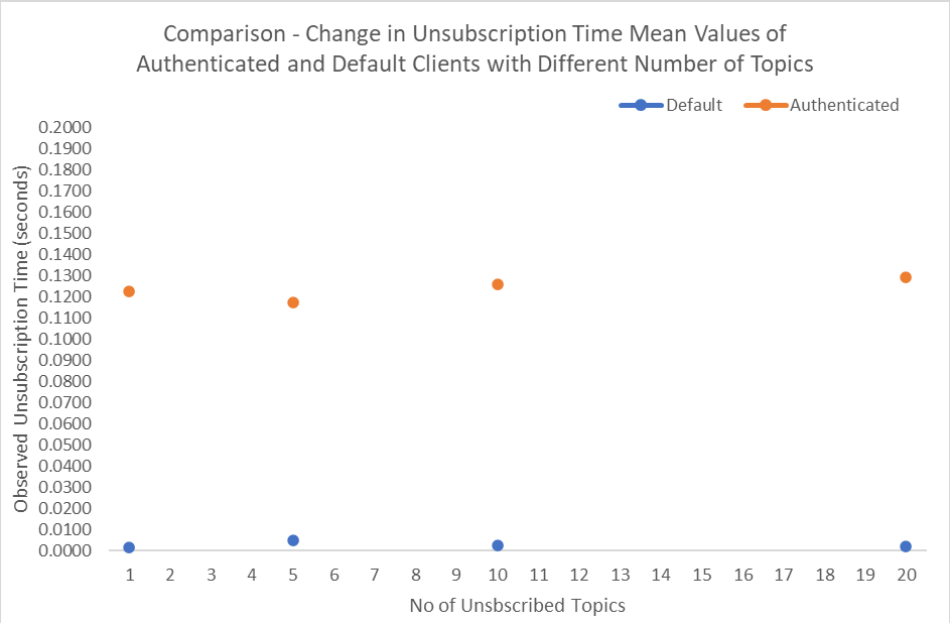

Figure 76

Comparison of Mean Values for Unsubscribing from Different Numbers of Topics for Authenticated and Default Client Instances

By looking at Figure 76, there is a clear increase in duration of unsubscribing in authenticated clients compared to the default client. Durations for authenticated client unsubscribing seem rather small on its own (approximately 0.13 second) but compared to default client's observations, there is an undeniable performance decrease in terms of durations. For 1 topic, the authenticated version is 84 times slower, for 5 topics, it is 25 times slower, for 10 topics, 54 times slower and for 20 topics, it is 69 times slower than the default version of that corresponding topic. There is a high number of increases but those increases are not consistent among themselves. As a conclusion, unsubscription can be tried to be improved if desired, but for getting consistent testing results, more instances should be tried as a precaution.

Figure 77 below shows the comparison of unsubscription operation for 3 different client types, the number of topics is equal to 1. This is an unexpected result since topic hashing client's unsubscription duration is observed less than the authenticated client's duration and it is almost equal to chat is observed for the default client. This might be due to business of the network used during this operation since the expectancy for the topic hashing client was having a similar mean value as the authenticated client. As a conclusion, performance of topic hashing client's unsubscribe operation is considered to be good enough.

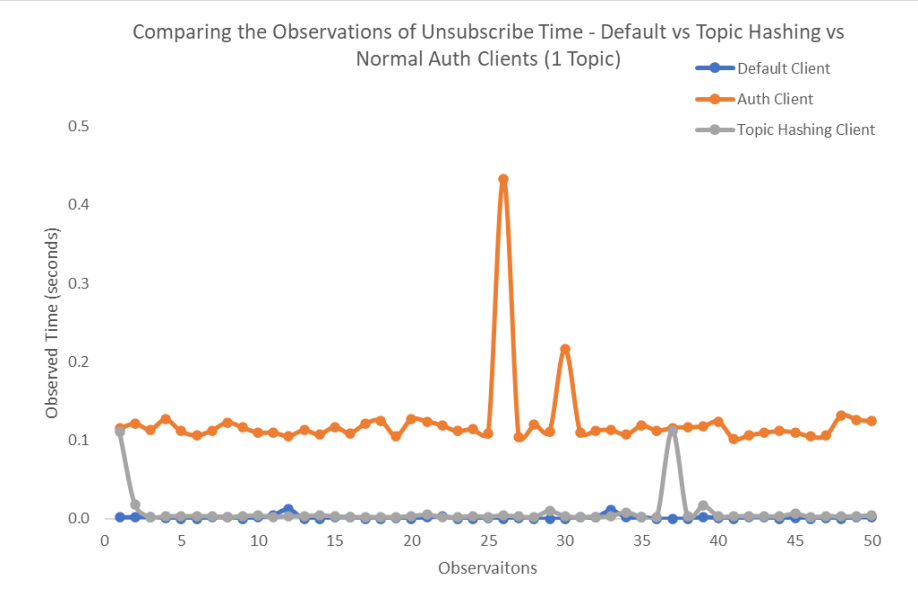

Figure 77

Comparison of Unsubscription to 1 Topic of Default, Authenticated and Topic Hashing Subscriber Client Modules

4.4.1.4. Performance of Publish Operation - Authenticated Client vs Default Client

Since clients can send a publish message to only one topic at a time, no topic number distinction has been made for this test suite, 50 instances of default client publish to 1 topic and 50 instances of authenticated client publish have been used. In the authenticated case, there exists extra operations to perform the publish operation compared to default. Similar to the subscribe operation, the client needs to know the related choice token in order to publish a message to a topic. Choice token retrieval and preparation and sending of the publish packet are included in the measurement of the publish operation for the authenticated client.

Figure 78 shows the histograms for the default client and the authenticated client. For the default case, there exists outliers but most of the observations are around the mean. But the same comment cannot be done for the authenticated client. Authenticated client has a wider distribution compared to the default one. But the determination of the performance relies on the comparison of mean values more than the distribution of the histograms.

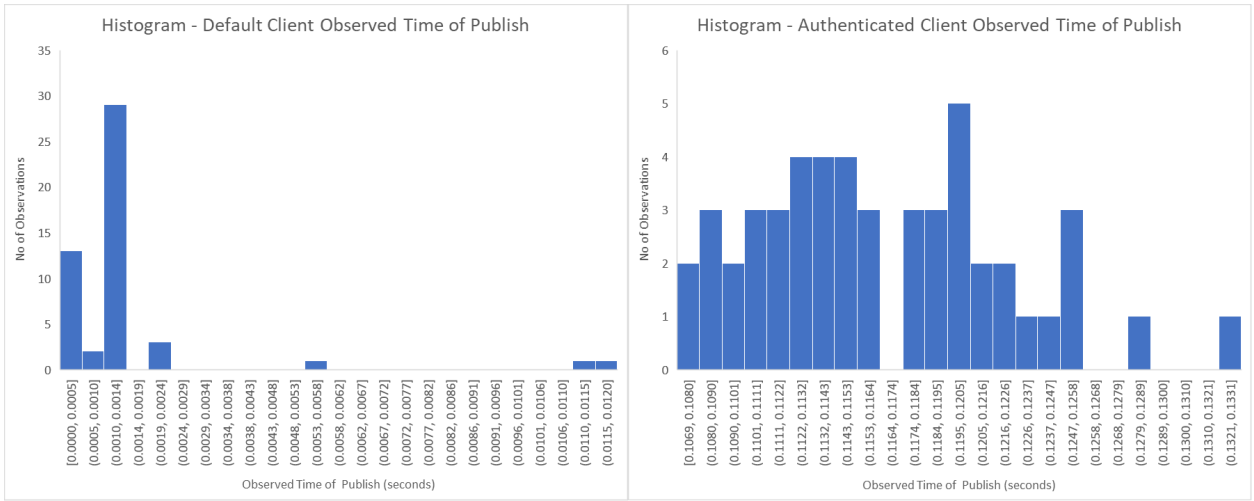

Figure 78

Histograms of Default and Authenticated Client Instances for Publish Operation

Figure 79 below shows the summary metrics for the publish operation bot for authenticated and the default client modules. By looking at the mean values, it is easy to see that the authenticated client module has a decrease in the performance of the publish operation. This can be related to the fact that an authenticated client needs to receive the choice token first in order to publish any message and should perform the choice token check (if the client has it already in its dictionary or not) each time the client desires to publish a message.

Dataset	no of observations	mean	std	median	min	max	auth/default ratio	ratio (percent)
Default Publish	50	0.0013	0.0023	0.0010	0.0000	0.0120	89.1841	8918%
Authenticated Publish	50	0.1164	0.0058	0.1156	0.1069	0.1331		

Figure 79

Summary Metrics for Publish Durations of Authenticated and Default Client Instances

Figure 80 below consists of a comparison of the observations of the default and the authenticated client module. There is a clear difference between the client types, and the observations seem consistent within themselves.

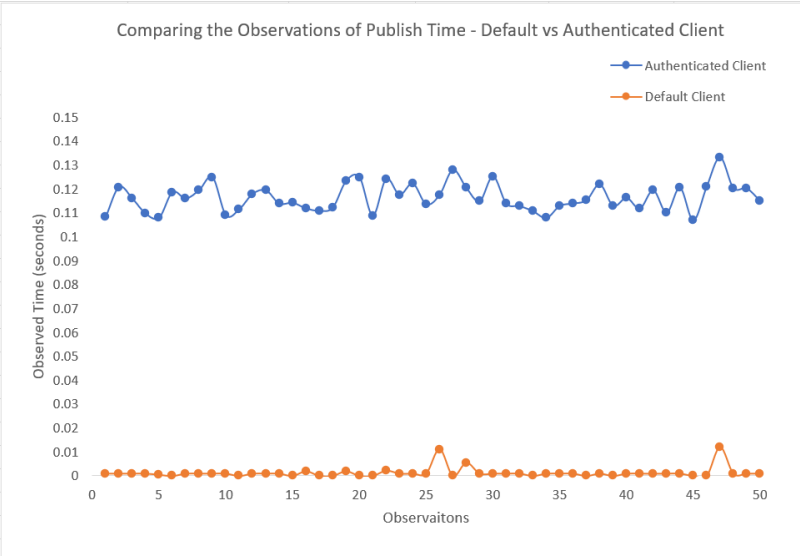

Figure 80

Comparison of Mean Values from Publish Operation by Authenticated and Default Client Instances

Since it is clear that performance is negatively affected for the publish in the authenticated client, mean values can be compared to see the difference in a single numerical value. By looking at Figure 80, in terms of publish operation, the authenticated client module's duration is 89 times of the default client's duration. As a conclusion, improvement might be needed in terms of the publish operation.

4.4.1.5. Performance of Publish Operation - Topic Hashing Publisher Client

The publish operation of topic hashing client modules (publisher client) is examined separately. Since in the hash session, the publisher client publishes to multiple topics with only one button click due to the nature of the topic hashing scheme, in order to ensure privacy, publishing to different numbers of topics have been measured and compared within each other.

The number of topics to be subscribed depends on the number of topics the publisher client adds to the current hash session. During this measurement, different numbers of topics are selected as 5, 10 and 15 to see if there is a linear or exponential change in terms of duration. For each measurement type, 100 instances have been used, making up to a total of 300 client instances.

Figure 81 below shows the histograms of publishing to different numbers of topics for the topic hashing publisher clients.

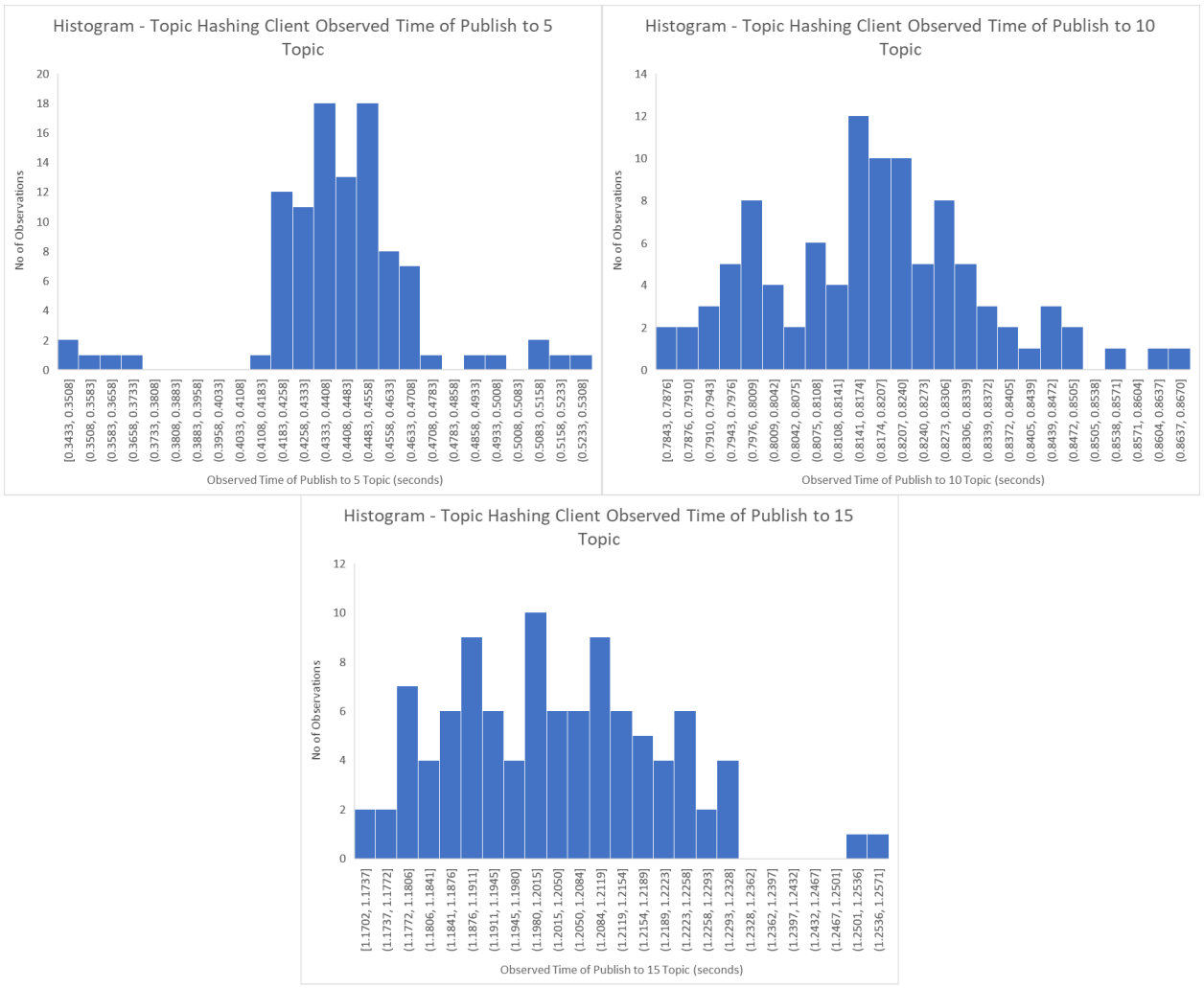

Figure 81

Histograms of Topic Hashing Publisher Client Instances for Publish Operation

Figure 82 below shows the summary metrics of the publish operation of topic hashing publisher clients for 5, 10 and 20 topics. By interpreting the mean values, the increase is linear, 0.4427 seconds have been observed for 5 topics, 0.8179 seconds have been observed for 10 topics and the duration approximately doubles when the number of topics doubles. For 15 topics, the duration is measured as 1.2026 seconds which is approximately 3 times of the 5 topics' duration.

Dataset	no of observations	mean	std	median	min	max
Topic Hashing Publish (5 Topic)	100	0.4427	0.0290	0.4447	0.3433	0.5308
Topic Hashing Publish (10 Topic)	100	0.8179	0.0165	0.8179	0.7843	0.8670
Topic Hashing Publish (15 Topic)	100	1.2026	0.0172	1.2014	1.1702	1.2571

Figure 82

Summary Metrics of Publishing to 5, 10, 15 Topics for Topic Hashing Publisher Client Instances

In order to see the difference clearly, a comparison graph is displayed in Figure 83. The increase between 3 different measurement types seems consistent and acceptable since the increase is not exponential.

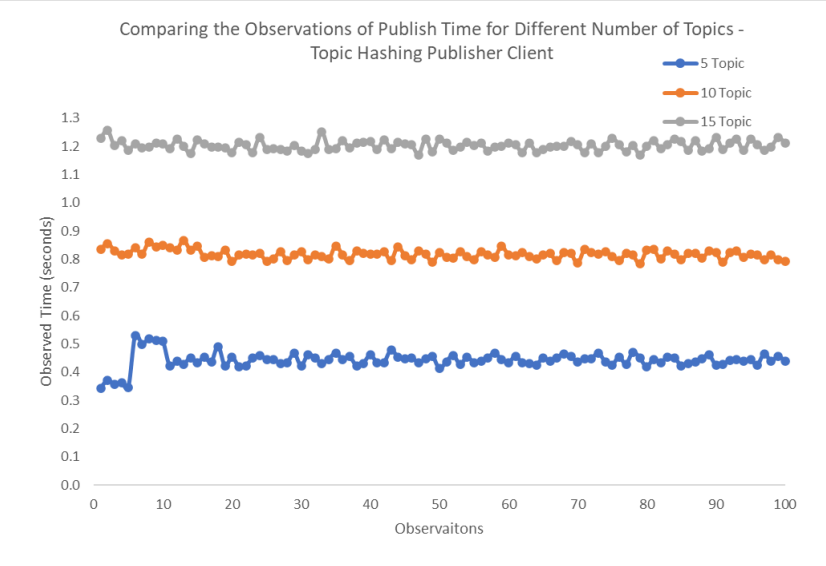

Figure 83

Comparison of Publish Durations to 5, 10, 15 Topics of Topic Hashing Publisher Client Instances

4.4.1.6. Performance of Publish Seed Operation

For publish seed operation, 5, 10 and 15 topics have been used as parameters and for each number of topics, 50 client instances have been used, making a total number of 150 client instances. Unfortunately, there is nothing to compare this operation with other than itself since publishing seeds is added for the topic hashing scheme only, default version and the authenticated version does not have it. Thus, the goal to reach here is to see if the increase is linear or not, linear increase is desired over exponential increase.

Figure 84 below shows the histograms of each measurement for each topic number used. The distributions have some outliers and very different results have been observed.

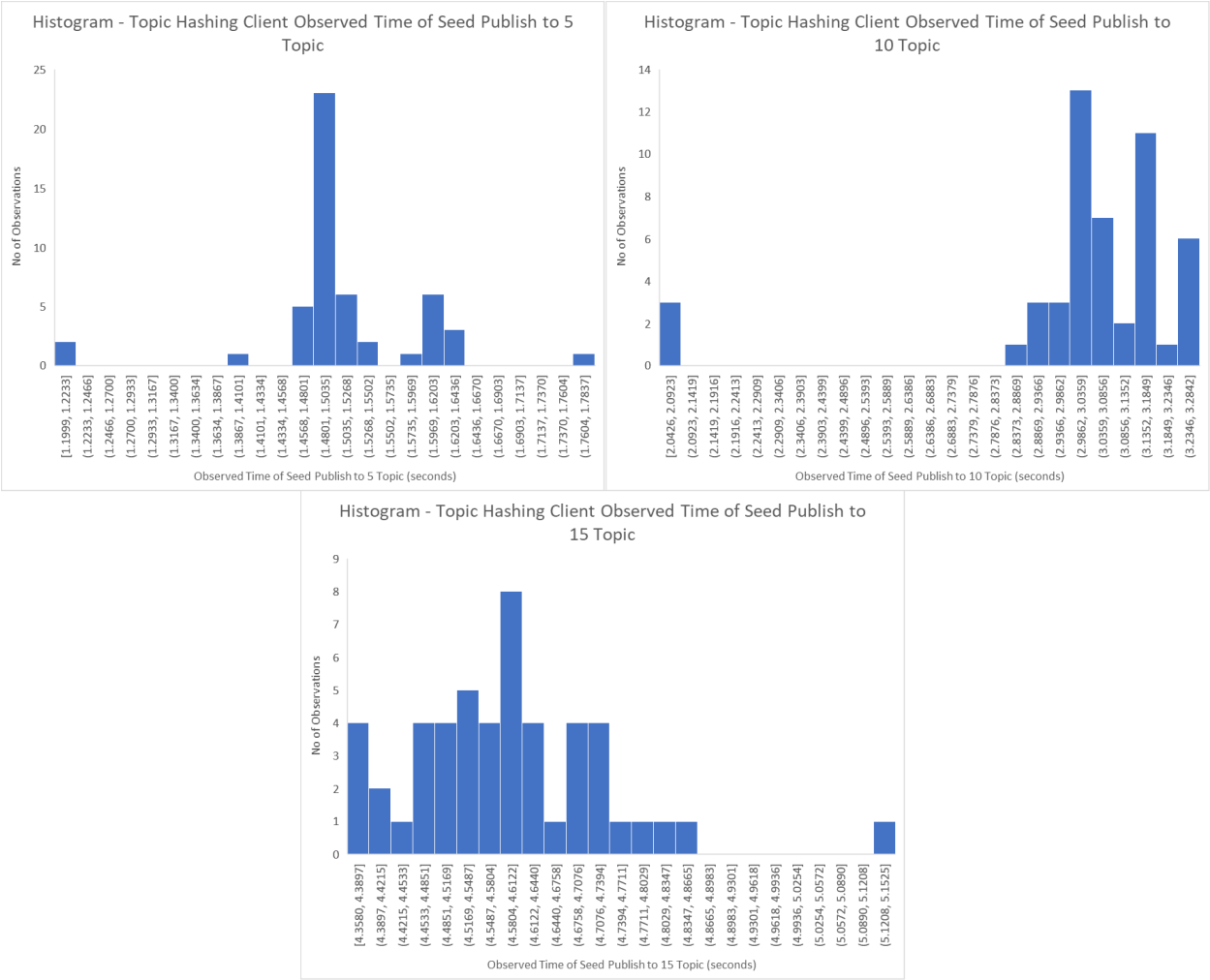

Figure 84

Histograms of Topic Hashing Publisher Client Instances for Publish Seed Operation Regarding 5, 10, 15 Topics

Figure 85 below shows the summary metrics of publish seed operation for different numbers of topics. The difference between 5, 10 and 15 topics are linear and this is a desired result. While duration for 5 topics is 1.5103 seconds, duration for 10 topics is approximately 3

seconds which is double of 5 topics' duration and duration for 15 topics is approximately 4.6 seconds which is 3 times of 5 topics' duration.

Dataset	no of observations	mean	std	median	min	max
Topic Hashing Seed Publish (5 Topic)	50	1.5103	0.0893	1.4968	1.1999	1.7837
Topic Hashing Seed Publish (10 Topic)	50	3.0254	0.2653	3.0442	2.0426	3.2842
Topic Hashing Seed Publish (15 Topic)	50	4.5894	0.1430	4.5908	4.3580	5.1525

Figure 85

Summary Metrics of Topic Hashing Publisher Instances' Seed Publish for 5, 10, 15 Topics

In order to make a better comparison between different topics, **Figure 86** below is added to show the difference. The amount of increase is consistent and this can be accepted since the increase is not exponential.

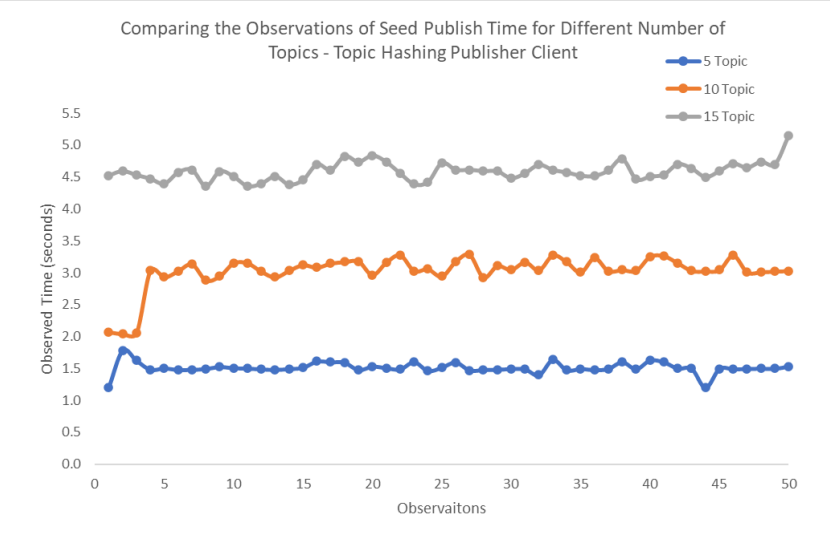

Figure 86

Comparison of Seed Publish for Different Number of Topics

4.4.1.7. Performance of Receiving Publish Operation (Receiving Message from a Subscribed Topic)

For the measurement of the publish receiving operation, 50 instances of authenticated client and 50 instances of authenticated client module and 50 instances for default client module have been used. Authenticated client again has more operations to perform compared to the default client. I should perform decryption and signature calculation and comparison operations in order to count the received message valid and display it to the console.

Figure 87 below shows histograms for the observations of publish receiving of the default client and authenticated client modules. Both histograms display more even distribution, not just around the mean like in some previously shown histograms. This shows there is no high consistency in terms of measured time, the duration can vary between observations in almost the same conditions.

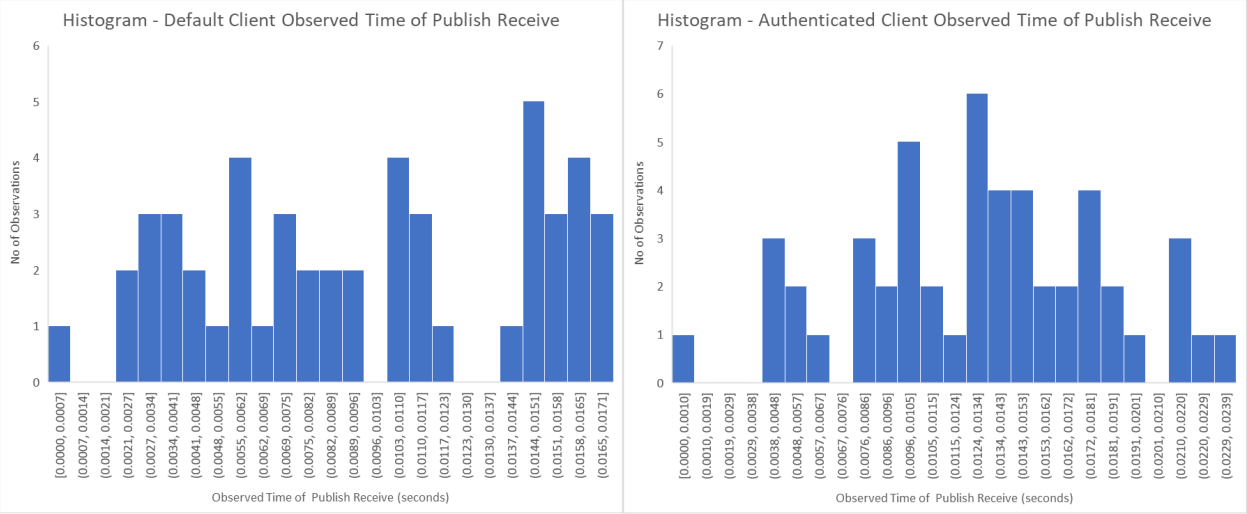

Figure 87

Histograms of Default and Authenticated Client Instances for Publish Receiving Operation

Figure 88 displays summary metrics of the observations for two client types, and mean values can be observed from it. Means value for authenticated client and default client are really close, authenticated client's mean of measured duration of receiving publish messages is only 1.35 times of the default client's mean of measured duration of receiving publish messages. This is a desired result, this means there is no improvement needed for publish receiving in the authenticated client module.

Dataset	no of observations	mean	std	median	min	max	auth/default ratio	ratio (percent)
Default Publish Receive	50	0.0096	0.0049	0.0094	0.0000	0.0171	1.3589	136%
Authenticated Publish Receive	50	0.0130	0.0053	0.0130	0.0000	0.0239		

Figure 88

Summary Metrics for Publish Receiving Durations of Authenticated and Default Client Instances

To support the conclusion that there is no clear distinction between the measurement of two client types and the difference between them can be accepted as a desired result, Figure 89 below can be examined.

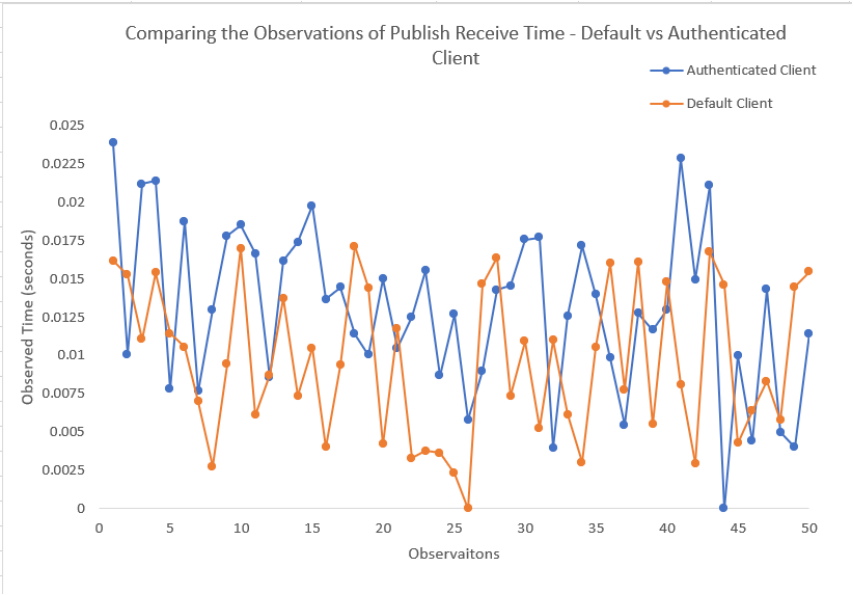

Figure 89

Comparison of Mean Values from Publish Receiving Operation by Authenticated and Default Client Instances

As mentioned above, there exists ups and downs in observations of the both client types. Authenticated client's observations are slightly higher in terms of duration (seconds) and this is also observed by comparing two mean values. As a conclusion, publish receiving operation's performance is as desired and currently an improvement is not expected.

4.4.2. Performance of the Broker

The information regarding the properties of the machines used during performance testing of the broker modules are provided below.

Default Broker and Authenticated Broker:

- Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz
- 32 GB RAM

- 512 GB SSD
- Windows 10 Pro 22H2

4.4.2.1. Performance of Connect Operation

Both for the authenticated version and the default version, 100 instances have been used, making a total number of 200 broker instances used. Below, Figure 90 is added in order to show the distribution of the observations. In both results, skewness has been observed. Figure 91 below shows the box plots of the observations of the broker instances, both for the default and the authenticated. As a prediction, a noticeable difference is expected between the authenticated version and the default version of the broker since authenticated version have many more steps compared to the default one, shown in Ephemeral Diffie-Hellman key exchange scheme in this report.

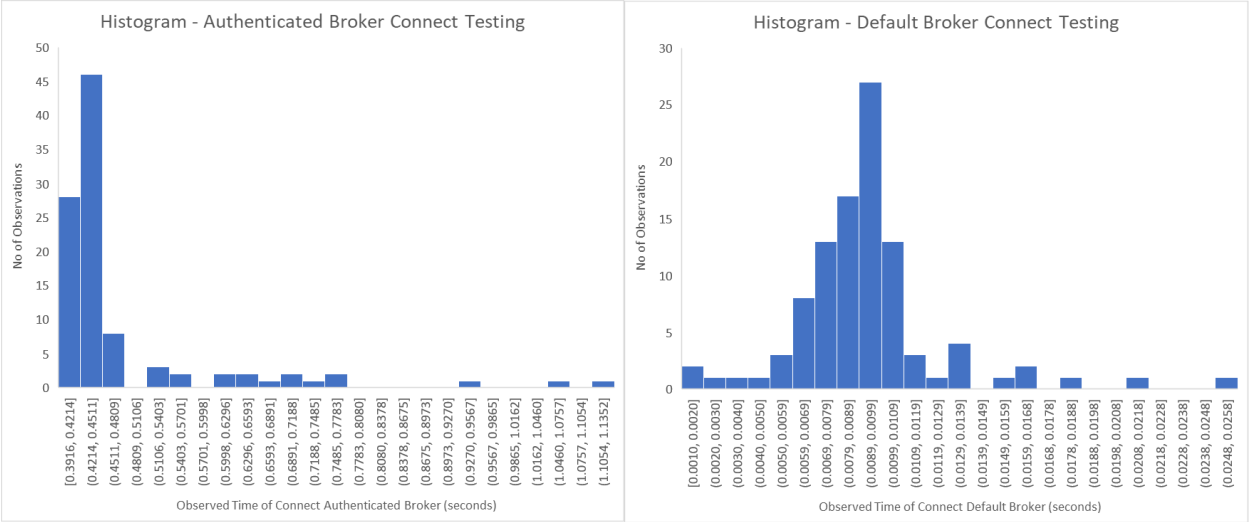

Figure 90

Histograms of Default and Authenticated Broker Instances of Connect Operation

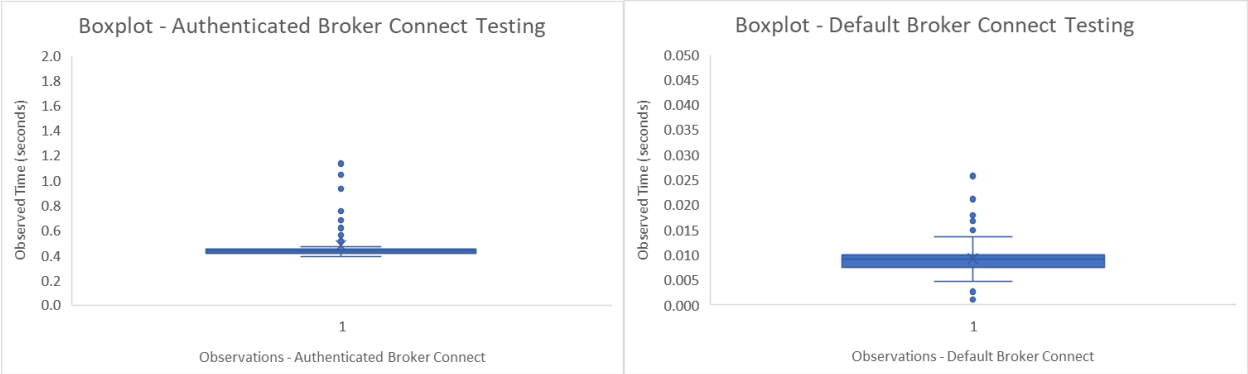

Figure 91

Box Plots of Default and Authenticated Broker Instances of Connect Operation

Figure 92 below shows the summary metrics for the default broker and authenticated broker instances. For the authenticated broker connection operation, approximately 52 times increase have been observed. This number might be due to outliers of the authenticated broker's connect sessions.

Dataset	no of observations	mean	std	median	min	max	auth/default ratio	ratio (percent)
Default Connect Broker	100	0.0092	0.0034	0.0092	0.0010	0.0258	52.0493	5205%
Authenticated Connect Broker	100	0.4777	0.1292	0.4354	0.3916	1.1352		

Figure 92

Summary Metrics for Connection Durations of Authenticated and Default Broker Instances

In order to see the difference better, a graph is added in order to make a comparison between the authenticated observations and the default observations, as Figure 93 below. As it can be interpreted by the graph, there exists extreme outliers in the observations of the authenticated broker instances. This clearly increases the mean value thus causing the ratio to be high. The outlier might be observed due to business of the network used, etc.

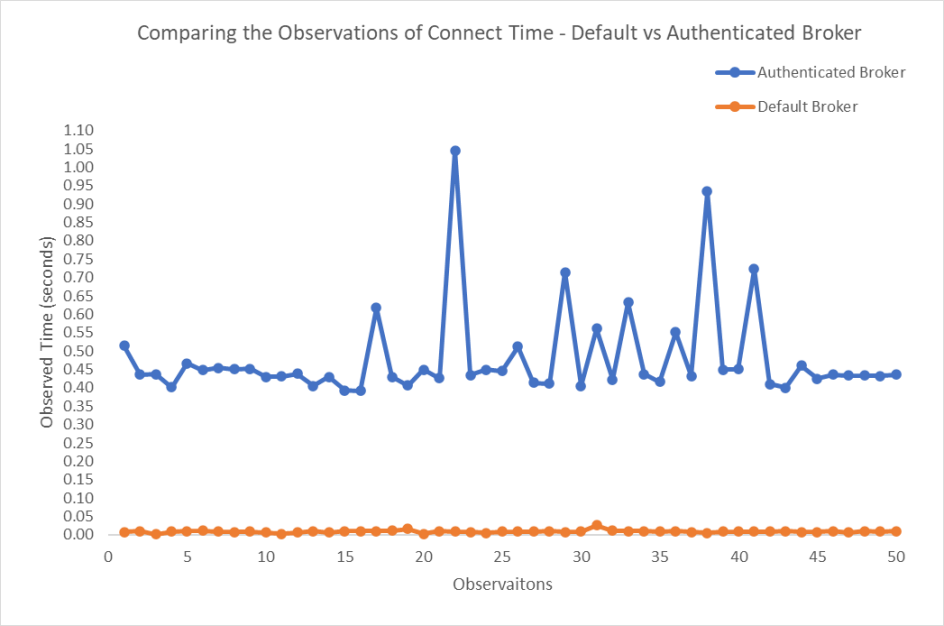

Figure 93

Comparison of Connect Operation of Authenticated and Default Broker Instances

As a conclusion, connect can be improved for the broker side in order to achieve better results later on but this can be tolerable due to having extreme outliers.

4.4.2.2. Performance of Subscribe Operation

For measuring the performance of the subscribe operation on the broker side, 100 default and 100 authenticated broker instances have been used, making a total of 200 broker instances used. Subscribing to only one topic is measured since the broker receives the subscribe requests one by one.

Below, Figure 94 shows the histograms of subscribe operation for the authenticated and the default broker instances. Default broker's observations are gathered around the mean mostly and make the observations consistent enough. Outliers have been observed for the authenticated broker instances and this clearly affects the mean value negatively.

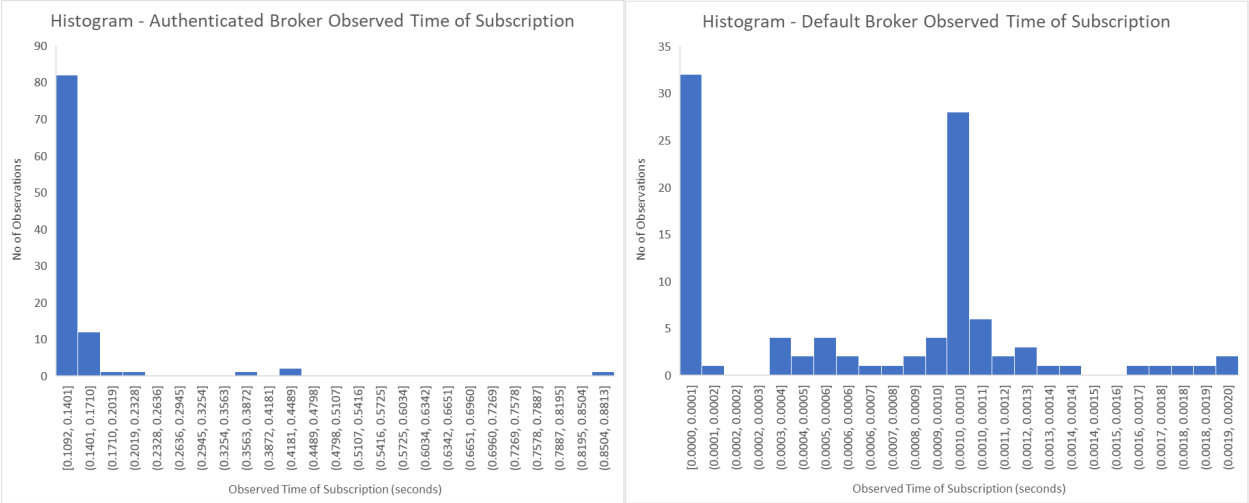

Figure 94

Histograms of Default and Authenticated Broker Instances of Subscribe Operation

Figure 95 below shows the summary metrics for the subscribe operation of the authenticated and default broker instances, and the resulting ratio is 223.8767. This result might be due to observed outliers of the authenticated broker instances. Even though the mean value of authenticated broker's subscription is not very high on its own, since the default version is very fast, the ratio is observed to be too high in the end.

Dataset	no of observations	mean	std	median	min	max	auth/default ratio	ratio (percent)
Default Subscribe Broker	100	0.0007	0.0005	0.0009	0.0000	0.0020	223.8767	22388%
Authenticated Subscribe Broker	100	0.1494	0.0893	0.1327	0.1092	0.8813		

Figure 95

Summary Metrics for Subscription Durations of Authenticated and Default Broker Instances

In order to see the difference better, a comparison graph is added below, as Figure 96. One can clearly see the extreme outliers that increase the mean value of authenticated broker instances' measurements. With more instances, the mean value might be improved or the structure of subscribe in the broker side can be altered to improve the results as a conclusion.

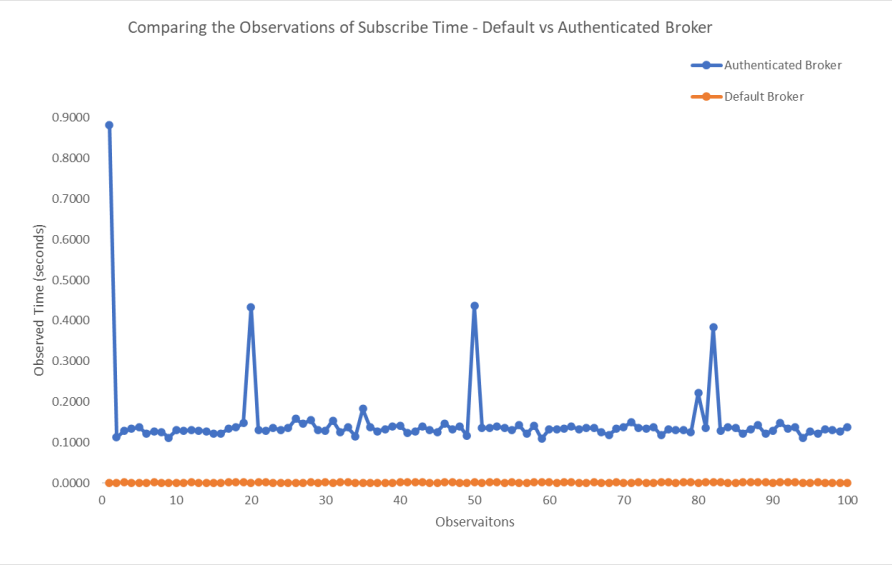

Figure 96

Comparison of Subscribe Operation of Authenticated and Default Broker Instances

4.4.2.3. Performance of Unsubscribe Operation

For measuring the performance of the unsubscribe operation on the broker side, 100 default and 100 authenticated broker instances have been used, making a total of 200 broker instances used. Unsubscribing from only one topic is measured since the broker receives the unsubscribe requests one by one.

Below, Figure 97 shows the histograms of unsubscribe operation for the authenticated and the default broker instances. Default broker's observations are gathered around the mean but some outliers have been observed too. Some outliers have been observed for the authenticated broker instances but most of the observations are close to the mean.

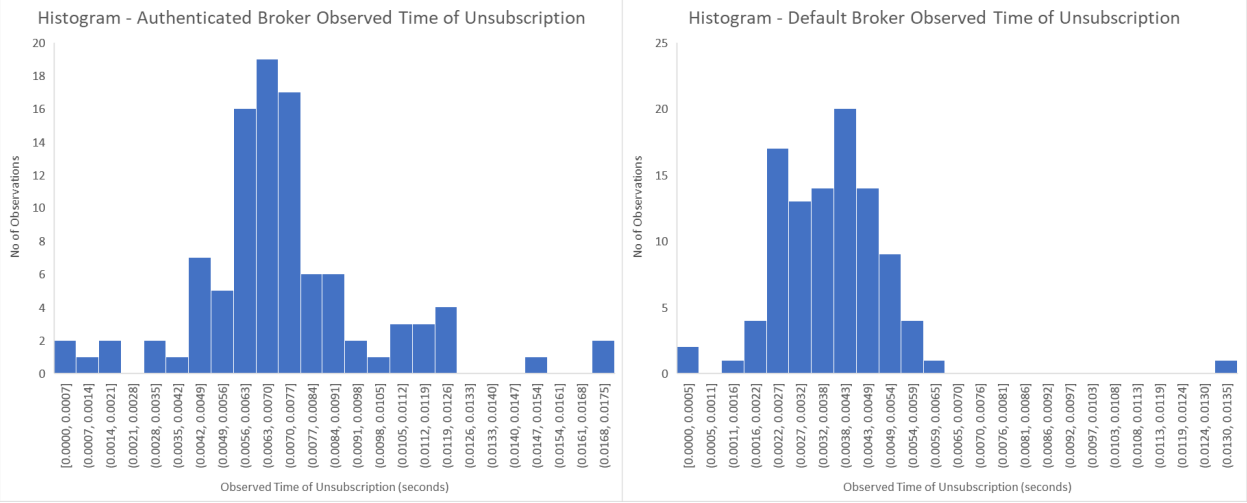

Figure 97

Histograms of Default and Authenticated Broker Instances of Unsubscribe Operation

Figure 98 below shows the summary metrics for the unsubscribe operation of the authenticated and default broker instances, and the resulting ratio is very much acceptable since it is only 1.9241 and this is an expected result.

Dataset	no of observations	mean	std	median	min	max	auth/default ratio	ratio (percent)
Default Unsubscribe Broker	100	0.0037	0.0015	0.0037	0.0000	0.0135	1.9241	192%
Authenticated Unsubscribe Broker	100	0.0072	0.0029	0.0068	0.0000	0.0135		

Figure 98

Summary Metrics for Unsubscription Durations of Authenticated and Default Broker Instances

In order to see the difference better, a comparison graph is added below, as Figure 99. One can clearly see that there is no clear difference between the two series that can be realized easily. Still, there exists some outliers for the authenticated broker instances' measurements but this is acceptable. As a conclusion, the ratio of 1.9241 is acceptable for the broker unsubscribe operation and there is no improvement needed currently.

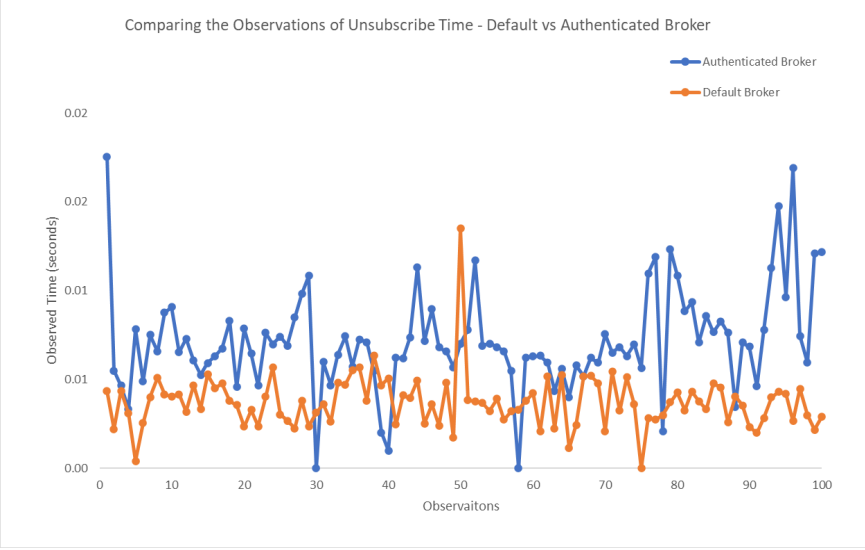

Figure 99

Comparison of Unsubscribe Operation of Authenticated and Default Broker Instances

4.4.2.4. Performance of Publish Operation

For measuring the performance of the publish operation on the broker side, 50 default and 50 authenticated broker instances have been used, making a total of 100 broker instances used. There is no distinction as to the different number of topics in the observations since a client can only publish a message to one topic at a time and the broker receives the publish packets one by one.

Figure 100 below shows the histograms of the publish operation of the authenticated and default broker instances separately. Authenticated broker instances' observations have some extreme outliers which might increase the mean value. There exists some outliers for the default instances too but most of the observations are gathered around the mean value.

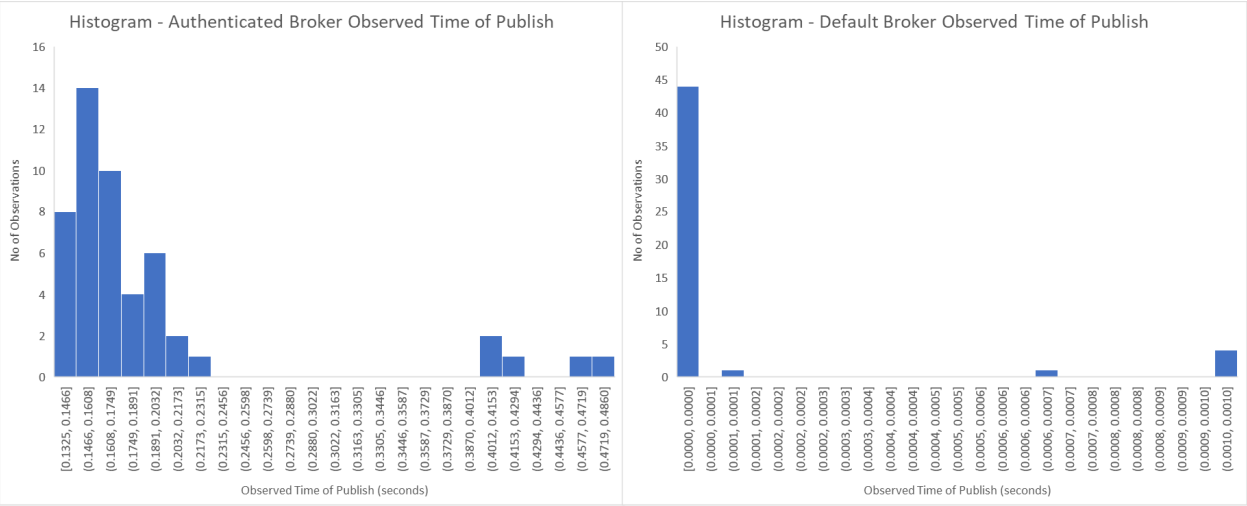

Figure 100

Histograms of Default and Authenticated Broker Instances of Publish Operation

Figure 101 below shows the summary metrics of publish operation for the authenticated broker and the default broker instances. The ratio is extremely high compared to other results that have been shown in this report. In order to better examine the results, a comparison graph is added below.

Dataset	no of observations	mean	std	median	min	max	auth/default ratio	ratio (percent)
Default Publish Broker	50	0.0001	0.0003	0.0000	0.0000	0.0010	2032.8160	203282%
Authenticated Publish Broker	50	0.1931	0.0844	0.1657	0.1325	0.4860		

Figure 101

Summary Metrics for Publish Durations of Authenticated and Default Broker Instances

Figure 102 below shows the comparison between default and authenticated broker instances of the publish operation. As it can be interpreted by the graph, authenticated observations have extremely high observations nearly equal to 0.5 seconds. This might be due to business of the network used during testing since many outliers have been observed while the other results are approximately seen closer to each other near to 0.2 seconds.

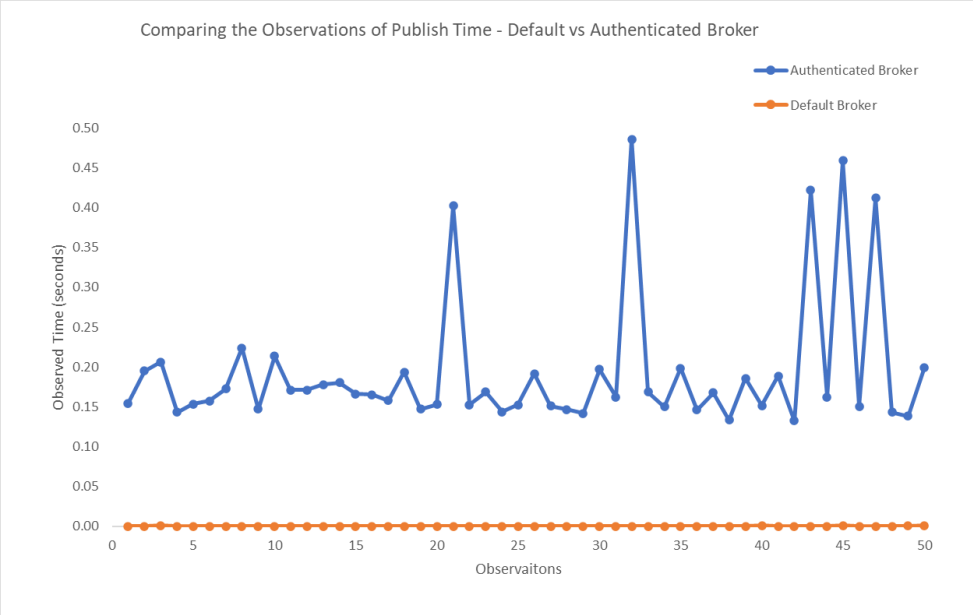

Figure 102

Comparison of Publish Operation for the Default and Authenticated Broker Instances

As a conclusion, publishing can be improved on the broker side and a different network should be tried out during the tests. After these improvements, mean values can be compared again to achieve healthier results.

4.5. Initial Objectives and Results

In terms of what was planned to be done, nearly all of the initial objectives were realized since the project was implemented successfully. All of the crucial parts of the detailed design were implemented. There are a few minor changes in the light of new knowledge learned in the implementation process. For example, in the acknowledgements of the MQTT there is no payload part or there is but it allows a few bits. So, in the default packet format we cannot add MAC of the packet identifiers to the acknowledgement packets. When we were in the design process, we thought of encapsulating the acknowledgements to add MAC values. But in the

implementation process, we realized that we can add payloads to the acknowledgements and we added the MAC values to those payloads.

Apart from these, additional things that we could not notice in the designing process were implemented. For example, handling the wildcard topics in a secure way was not thought of in the design process. Also, there was no plan for the security of unsubscribe packets. The actions to be taken in case of mismatch of MACs are another issue that we did not think of in the design process.

In terms of the outputs that we expected to see, the topic hashing scheme was a disappointment for us since it looks that there is no possibility of using it in real systems. As far as we could do, there is just one publisher in the whole communication of topic hashing scheme. Also, the clients need to run synchronously in order to get the same hash value of topic names for each iteration. This constraint is one of the reasons why the topic hashing scheme is not applicable in real systems, synchronization of the clients may not be achieved due to many reasons such as the behaviors of the clients, they might not select to subscribe to desired topics at the same time. After the hash session starts, none of the subscribers can subscribe to the topics during that hash session. Besides the previous problem, if one publish message sent by the publisher does not reach a subscriber due to some network problems, that subscriber won't get any message from that topic along that hash session again. Because the next hash value of that topic is calculated on the publisher side but the subscriber does not calculate the next hash value since it does not know to do that. Besides that it turned out that there was no need for the bloom filter to be used and subscribing to random topics, both were removed from the design.

This project has contributions in terms of providing security features not available in the default MQTT protocol. Since the default protocol is not enough to ensure required security

principles, this project provides confidentiality, integrity, authentication, and authorization on top of default MQTT.

5. IMPACT

As a socio-economic impact, since all of the materials we have used are open source, our solution is open source and free as well. Users can achieve authentication, confidentiality, integrity and authorization, etc. Anyone can use it without a fee, this project does not have commercialization potential. As mentioned earlier, mainly used resources in this project (Eclipse Paho MQTT Client, aMQTT Broker) are open source resources thus there exists no Freedom-To-Use (FTU) issues.

Although there exists other examples in the literature, by actualizing this project, it has been verified again that achieving security principles like authentication, confidentiality, integrity and authorization are possible in IoT technologies. Of course, there is a tradeoff regarding the lightweight nature of IoT and the security aspect but it is possible to achieve mentioned security principles by compromising to some degree from the lightweight characteristic of IoT.

6. ETHICAL ISSUES

There exist no ethical issues as a consequence of our solution. In fact, our solution aims to eliminate a significantly large ethical issue as an objective, which is the issue of providing necessary security principles during MQTT messaging. The absence of these principles would put both parties in danger directly since various attacks might happen during the MQTT messaging and by providing those principles to the clients, a large ethical issue as privacy of data can be handled.

7. PROJECT MANAGEMENT

In terms of what was planned to be done, nearly all of the initial objectives were realized since the project was implemented successfully. But we could not follow the time table properly which was planned at the design process for the implementation process. The biggest reason of that we changed the broker software that we planned to use at the design process. At the design stage, we planned to use the Eclipse Mosquitto as the broker. But the source code of the Mosquitto broker was not manageable to make required changes to implement the features we had introduced in our design charts in the limited amount of time that we had. In order to prevent losing more time, we have decided to change the MQTT broker software from Mosquitto to aMQTT in the third week of the semester. This change made us waste some time since we decided to change in the third week and we did not know anything about the aMQTT broker at that time. The learning process of the aMQTT broker caused delays in the time table. But at the later stages of the project, that time delay has been closed by spending more time on the project.

In terms of managing the project, we realized that details unpredictable in the design process may emerge during the implementation process. These details may cause changes and additions in the design schemes.

8. CONCLUSION AND FUTURE WORK

In terms of important findings of this project, one issue is the awareness of the importance of security features required to prevents private data to be reached by unauthorized third parties and to be changed during the communication of the parties.

In terms of limitations of this project, one issue was encountered during the performance testing. Since we do not have enough computers and people for testing, we could not increase the connection load on the broker side properly. We just compared the performance of the default client and our authenticated version of the client for observing the overhead created by added security features. Another limitation is the time. This project contains a lot of details which makes it difficult to realize all the cases in the limited amount of time that we had.

In terms of logical next steps of this project, one issue is to examine the topic hashing scheme in detail, determine possible improvements/changes/removals and to evaluate the new results regarding the applicability of the scheme, especially in the design part. As mentioned earlier, due to multiple constraints and edge cases, the topic hashing scheme is not applicable for real systems at this point. Another possible logical next step is to make changes in the implemented code to try to improve the performance and make the updated protocol more lightweight. As an example, different encryption/decryption and hashing libraries and methods can be tried out and after the testing phase, new results can be compared with the current results. Last suggestion as a next logical step is to achieve improvements in current GUIs (especially in

topic hashing subscriber and publisher GUIs). A new library with more options can be tried out to replace the current library, Tkinter. Design of the topic hashing GUIs can be attempted to change as well.

9. APPENDIX

MQTT Control Packet Format

An MQTT control packet includes three parts. These are fixed header, variable header and payload. The fixed header is a 2 byte header which is present in all of the MQTT packets (Banks et. al, 2014). In the first byte of the fixed header, the first most significant four bits specify the MQTT control packet type. The last four bits of the first byte of the fixed header is reserved for flags specific to packet type. Flag parts of some packets are marked as ‘reserved’ for future uses. The second byte of the fixed header contains the remaining length of the packet. The variable header is a header that is located between the fixed header and payload (Banks et. al, 2014). Variable header is not present in all of the MQTT packets and its content varies regarding the packet type. The payload is the final part which is present in some types of the MQTT control packets.

Some MQTT control packet types are the followings: CONNECT, CONNACT, PUBLISH, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK, DISCONNECT.

The CONNECT packet contains fixed header, variable header and payload. In the fixed header as shown in Figure 103, MQTT control packet type for CONNECT packet is 0001, and the flag is 0000 which means ‘reserved’ (Banks et. al, 2014).

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (1)				Reserved			
	0	0	0	1	0	0	0	0
byte 2…	Remaining Length							

Figure 103

CONNECT Packet Fixed Header

The variable header of the CONNECT packet includes protocol name, protocol level, connect flags and keep alive as can be seen in Figure 104. Protocol name is MQTT. Protocol level indicates the version of the MQTT to be used. For MQTT 3.1.1, it is 00000100 (Banks et. al, 2014). Connect flags in the variable header are username flag and password flag, will retain, will QoS, will Flag, Clean Session and reserved. If the username and the password will be used, then their flags are set to 1, otherwise set to 0. Will flag indicates whether the will message will be sent in case the client disconnects ungracefully. If the will flag is set to 1, the will topic and will message fields are present in the payload of the CONNECT packet (Banks et. al, 2014). Will QoS is a two bit information to specify the Quality of Service level that will be used when the will message will be sent by the broker (Banks et. al, 2014). It can take values of 00, 01 and 10. 00 means the packet will be sent at most once and it does not provide any guarantee. 01 means the packet will be sent at least once. 00 means the packet will be sent exactly once. The clean session flag specifies whether the client wants to use a persistent session (HiveMQ, 2020).

	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Protocol Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (4)	0	0	0	0	0	1	0	0
byte 3	'M'	0	1	0	0	1	1	0	1
byte 4	'Q'	0	1	0	1	0	0	0	1
byte 5	'T'	0	1	0	1	0	1	0	0
byte 6	'T'	0	1	0	1	0	1	0	0
Protocol Level									
	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 7	Level (4)	0	0	0	0	0	1	0	0
Connect Flags									
byte 8	User Name Flag (1) Password Flag (1) Will Retain (0) Will QoS (01) Will Flag (1) Clean Session (1) *Reserved* (0)	1	1	0	0	1	1	1	0
Keep Alive									
byte 9	Keep Alive MSB (0)	0	0	0	0	0	0	0	0
byte 10	Keep Alive LSB (10)	0	0	0	0	1	0	1	0

Figure 104

CONNECT Packet Variable Header

The last part of the variable header is kept alive which indicates the longest time period that the client and the broker can stand in the connection without sending messages. If there are no messages to be sent over the connection, then the client will send a ping request to keep the connection open.

The payload of the CONNECT packet contains client identifier, will topic, will message, username and password. The client identifier is a unique clientId. If the will flag is set to 1, the will topic and will message fields specify the topic name and the message that will be sent in the case clients disconnect ungracefully (HiveMQ, 2020). If the username flag and password flag are set to be 1, then the username and password information will be sent in the related fields (Banks et. al, 2014).

CONNACK packet contains fixed header and variable header. In the fixed header, MQTT control packet type for CONNACK packet is 0010, and the flag is 0000 which means 'reserved' (Banks et. al, 2014). The variable header of the CONNACK packet includes the session present flag, and connect return code. The session present flag is a one bit information that specifies whether the broker has previous session information stored for this client. Return code is one byte information that specifies whether the connection attempt was successful or not.

PUBLISH packet contains fixed header, variable header and payload. In the fixed header as shown in Figure 105, MQTT control packet type for PUBLISH packet is 0011. The first bit of the flag part is the DUP flag which is going to be set to 1, if this publish message has been sent at least one time before (Banks et. al, 2014). The second and third bit of the flag part gives the QoS level which can be 00, 01, or 10. The last bit of the flag part is used as a retain flag that specifies if this message will be saved by the broker as a last message in order to be sent to new subscribers that subscribe to the specified topic.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (3)				DUP flag	QoS level		RETAIN
	0	0	1	1	X	X	X	X
byte 2	Remaining Length							

Figure 105

PUBLISH Packet Fixed Header

The variable header of the PUBLISH packet includes topic name and packet identifier fields. An example variable header is shown in Figure 106.

	Description	7	6	5	4	3	2	1	0
Topic Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	'a' (0x61)	0	1	1	0	0	0	0	1
byte 4	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 5	'b' (0x62)	0	1	1	0	0	0	1	0
Packet Identifier									
byte 6	Packet Identifier MSB (0)	0	0	0	0	0	0	0	0
byte 7	Packet Identifier LSB (10)	0	0	0	0	1	0	1	0

Figure 106

Example for Publish Packet Variable Header

The last part of the PUBLISH packet is the payload which includes the actual content of the message. The broker gives different responses depending on the QoS level of the publish message (Banks et. al, 2014). If QoS level is 0, the broker will not send any acknowledgements. If QoS level is 1, the PUBACK packet will be sent by the broker. If QoS level is 2, PUBREC packet will be sent by the broker.

PUBACK packet contains fixed header and variable header. In the fixed header, MQTT control packet type for PUBACK packet is 0100 (Banks et. al, 2014). The variable header

contains a 2 byte packet identifier field which contains the identifier of the PUBLISH packet it acknowledges.

PUBREC packet is the response sent by the broker for the PUBLISH packet, if the QoS level is 2. It contains a fixed header and variable header. In the fixed header, MQTT control packet type for PUBREC packet is 0101 (Banks et. al, 2014). The variable header contains a 2 byte packet identifier field which contains the identifier of the PUBLISH packet it acknowledges.

PUBREL packet is the response sent by the publisher for the PUBREC, if the QoS level is 2. It contains a fixed header and a variable header. In the fixed header, MQTT control packet type for PUBREL packet is 0110. The variable header contains a 2 byte packet identifier field which contains the identifier of the PUBREC packet it acknowledges (Banks et. al, 2014).

PUBCOMP packet is the response sent by the broker for the PUBREL , if the QoS level is 2. It contains a fixed header and a variable header. In the fixed header, MQTT control packet type for PUBCOMP packet is 0111 (Banks et. al, 2014). The variable header contains a 2 byte packet identifier field which contains the identifier of the PUBREL packet it acknowledges.

SUBSCRIBE packet contains fixed header, variable header and payload. In the fixed header as shown in Figure 107, MQTT control packet type for SUBSCRIBE packet is 1000.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type (8)				Reserved			
	1	0	0	0	0	0	1	0
byte 2	Remaining Length							

Figure 107

SUBSCRIBE Packet Fixed Header

The variable header of the SUBSCRIBE packet includes a packet identifier as shown in Figure 108.

	Description	7	6	5	4	3	2	1	0
Packet Identifier									
byte 1	Packet Identifier MSB (0)	0	0	0	0	0	0	0	0
byte 2	Packet Identifier LSB (10)	0	0	0	0	1	0	1	0

Figure 108

SUBSCRIBE Packet Variable Header

The last part of the SUBSCRIBE packet is the payload which includes one or more Topic Filter / QoS pairs. It is shown in Figure 109. The topic filter may specify a single topic name or multiple topic names by using wildcards characters. Each topic filter has a requested QoS level.

Description	7	6	5	4	3	2	1	0
Topic Filter								
byte 1	Length MSB							
byte 2	Length LSB							
bytes 3..N	Topic Filter							
Requested QoS								
	Reserved						QoS	
byte N+1	0	0	0	0	0	0	X	X

Figure 109

SUBSCRIBE Packet Payload Format

SUBACK packet is the response sent by the broker for the SUBSCRIBE packet. It contains fixed header, variable header and payload. In the fixed header, MQTT control packet type for SUBACK packet is 1001 (Banks et. al, 2014). The variable header contains the packet identifier of the SUBSCRIBE packet it acknowledges. The payload includes a list of return codes

for all topic filters in the SUBSCRIBE packet. The format of the payload is shown in Figure 110. If the broker accepts the subscription for that topic filter, values of the return code can be 0x00, 0x01 or 0x10 depending on the QoS level. If the broker refuses a subscription, the return code value is 0x80.

Bit	7	6	5	4	3	2	1	0
	Return Code							
byte 1	X	0	0	0	0	0	X	X

Figure 110

SUBACK Packet Payload Format

UNSUBSCRIBE packet contains fixed header, variable header and payload. In the fixed header, MQTT control packet type for UNSUBSCRIBE packet is 1010 (Banks et. al, 2014) . The variable header of the UNSUBSCRIBE packet includes a packet identifier. The payload includes one or more Topic Filters which the clients want to unsubscribe.

UNSUBACK packet is the response sent by the broker for the UNSUBSCRIBE packet. It contains fixed header, and variable header. In the fixed header, MQTT control packet type for UNSUBACK packet is 1011 (Banks et. al, 2014). The variable header contains the packet identifier of the UNSUBSCRIBE packet it acknowledges.

DISCONNECT packet is sent by the client to the broker. It contains a fixed header. The MQTT control packet type for DISCONNECT packet is 1110 (Banks et. al, 2014)

10. **REFERENCES**

aMQTT. (n.d.). Retrieved April 16, 2023, from https://amqtt.readthedocs.io/en/latest/

Anthraper, J., & Kotak, J. (2019). Security, Privacy and Forensic Concern of MQTT Protocol. *Political Economy - Development: Domestic Development Strategies eJournal.* https://doi.org/10.2139/ssrn.3355193

Base64 - Base16, Base32, base64, Base85 Data encodings. Python documentation. (n.d.). Retrieved April 16, 2023, from https://docs.python.org/3/library/base64.html

Calabretta, M., Pecori, R., Vecchio M. & Veltri, L. (2018). MQTT-Auth: a Token-based Solution to Endow MQTT with Authentication and Authorization Capabilities. *Journal of Communications Software and Systems*, 14(4), 320-331. Retrieved from https://doi.org/10.24138/jcomss.v14i4.604

Hash-based message authentication codes (HMAC). Hash-based message authentication codes (HMAC) - Cryptography 41.0.0.dev1 documentation. (n.d.). Retrieved April 16, 2023, from https://cryptography.io/en/latest/hazmat/primitives/mac/hmac/

Hashlib - secure hashes and message digests. Python documentation. (n.d.). *https://docs.python.org/3/library/hashlib.html*

Hue, A., Sharma, G., & Dricot, J.-M. (2021). Privacy-Enhanced MQTT Protocol for Massive IoT. *Electronics*, 11(1), 70. MDPI AG. Retrieved from http://dx.doi.org/10.3390/electronics11010070

IBM. (2022, February 14). *MQTT standards and requirements*. Retrieved from https://www.ibm.com/docs/en/maximo-monitor/8.4.0?topic=messaging-standards-requirements.

ISO/IEC. (2016). *ISO/IEC 20922:2016(en) information technology — message queuing telemetry transport (MQTT) v3.1.1*. Retrieved from https://www.iso.org/obp/ui/#iso:std:iso-iec:20922:ed-1:v1:en

McGrew, D. (2008). *An Interface and Algorithms for Authenticated Encryption*. (IETF RFC 5116). IETF. https://www.ietf.org/

MQTT & MQTT 5 Essentials. Retrieved from https://www.hivemq.com/mqtt-essentials

MQTT Version 3.1.1. (2014, 29 October). Edited by Andrew Banks and Rahul Gupta. OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

Niruntasukrat, A., Issariyapat, C., Pongpaibool, P., Meesublak, K., Aiumsupucgul, P., & Panya, A. (2016). Authorization mechanism for MQTT-based internet of things. *2016 IEEE International Conference on Communications Workshops (ICC)*. https://doi.org/10.1109/iccw.2016.7503802

Py-diffie-hellman. PyPI. (n.d.). Retrieved April 16, 2023, from https://pypi.org/project/py-diffie-hellman/

Ramos, Santiago Hernández , Villalba, M. & Lacuesta, Raquel (2018). MQTT Security: A Novel Fuzzing Approach. Wireless Communications and Mobile Computing Volume 2018, Article ID 8261746, Retrieved from https://doi.org/10.1155/2018/8261746

Rescorla, E. (1999). *Diffie-Hellman Key Agreement Method*. (IETF RFC 2631). IETF. https://www.ietf.org/

RSA. RSA - Cryptography 40.0.0.dev1 documentation. (n.d.). Retrieved January 8, 2023, from https://cryptography.io/en/latest/hazmat/primitives/asymmetric/rsa/

Singh, M., Rajan, M. A., Shivraj, V. L., & Balamuralidhar, P. (2015). Secure mqtt for internet of things (IOT). *2015 Fifth International Conference on Communication Systems and Network Technologies*. https://doi.org/10.1109/csnt.2015.16

Secrets - generate secure random numbers for managing secrets. Python documentation. (n.d.). Retrieved April 16, 2023, from https://docs.python.org/3/library/secrets.html

Symmetric Encryption. Symmetric encryption - Cryptography 41.0.0.dev1 documentation. (n.d.). Retrieved April 16, 2023, from https://cryptography.io/en/latest/hazmat/primitives/symmetric-encryption/

Upadhyay, Y., Borole, A., & Dileepan, D. (2016). MQTT based secured home automation system. *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*. https://doi.org/10.1109/cdan.2016.7570945

X.509. X.509 - Cryptography 40.0.0.dev1 documentation. (n.d.). Retrieved January 8, 2023, from https://cryptography.io/en/latest/x509/