Implementation of an authentication mechanism for MQTT

Carlo Leo & Francesco Bocchi

Department of Computer Engineering - Cybersecurity

Introduction: what is MQTT? [1/2]

- MQTT is a publish/subscribe protocol that provides a scalable and reliable way to connect devices over the Internet.

- Its main scenario is related to the IoT landscape due to its minimal resources requirements. Therefore its lightweight structure is suitable for all the IoT devices.

Introduction: what is MQTT? [2/2]

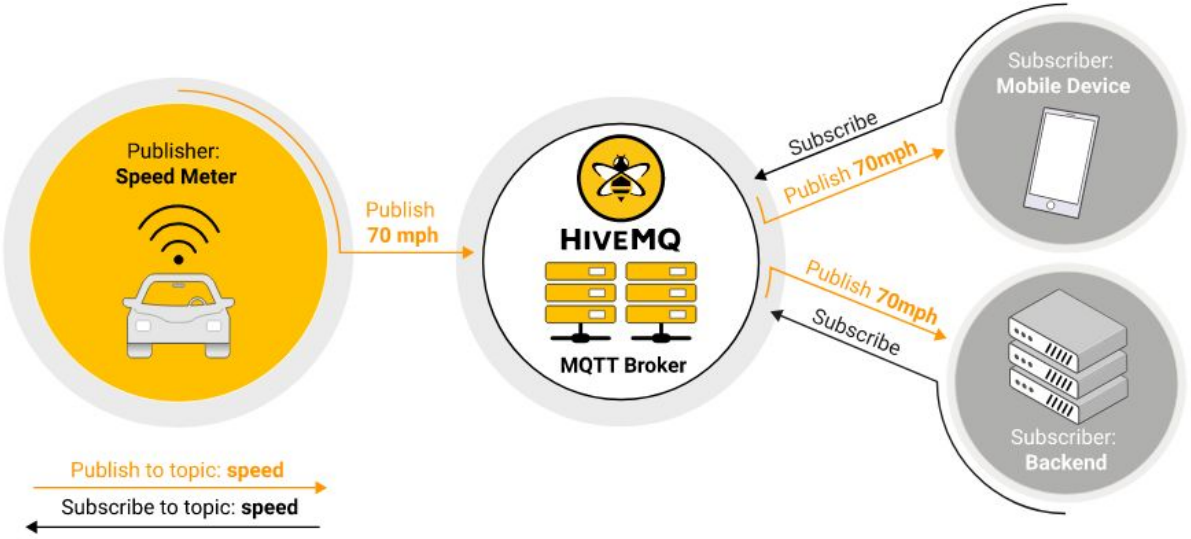
- MQTT Broker
- MQTT Client

An MQTT client can publish messages as soon as it connects to a broker. MQTT utilizes **topic-based** filtering of the messages on the broker

[1] Topic structure

Typical MQTT communication

We used HiveMQ as implementation of a MQTT Broker

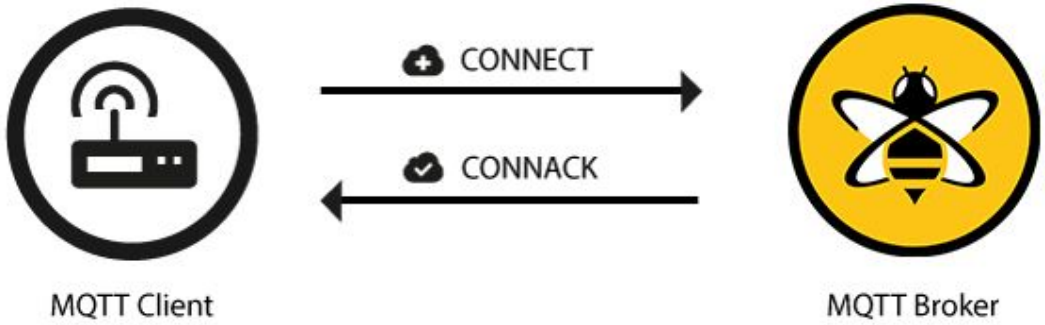


[2] MQTT Publish / Subscribe Architecture

Problem: authentication phase [1/2]

- A critical phase in this protocol is represented by authentication of MQTT Clients to the MQTT Broker.

- The client sends a CONNECT message to the broker. The broker responds with a CONNACK message and a status code



[3] Connection phase in HiveMQ

Problem: authentication phase [2/2]

- In order to deal with this problem, we implemented a challenge-response authentication mechanism to avoid direct exchanging of passwords during connection.
- HiveMQ offers an open-source SDK that allows creation of custom broker-extension used to add ***fine-grained security*** layer.
- As a consequence, we also developed a custom client that supports the protocol extension.

Solution design

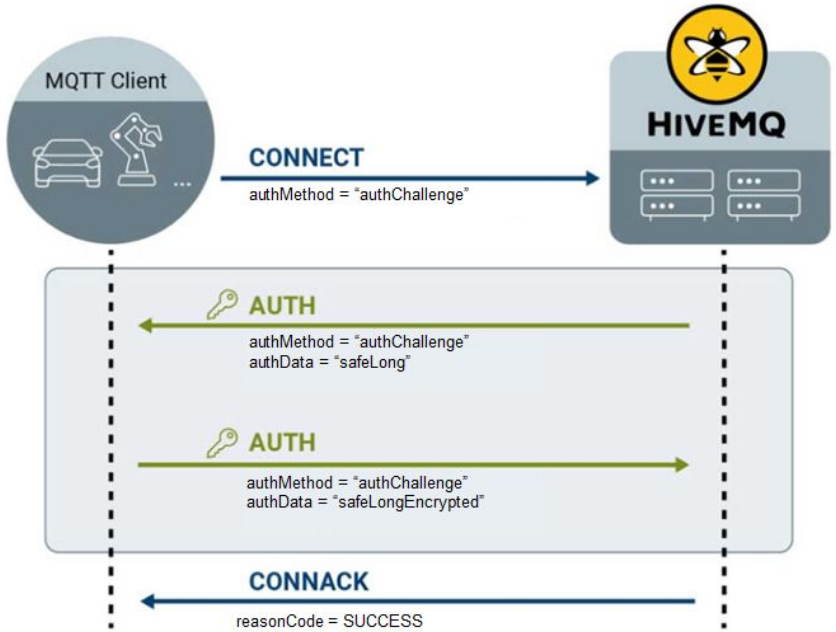
It has been added a further field ***"authMethod"*** to the authentication packet that specifies the type of authentication.

So a client authentication phase can be described as follow:

1. If a client is not registered, takes place a **Simple Authentication** where the broker saves the client credentials locally (hashed password)
2. If the client is already registered, takes place a **Challenge-Response Authentication** where the broker will send a challenge based on the client password

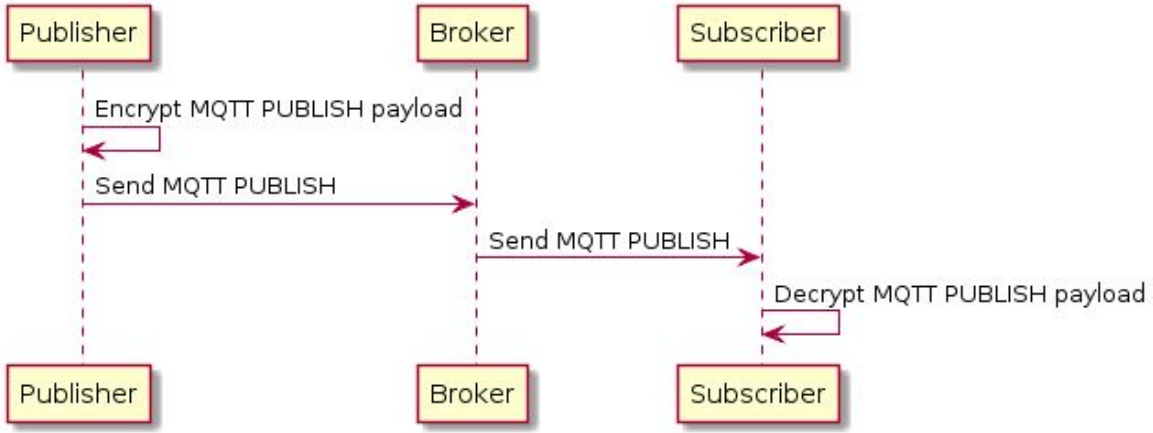
Challenge-Response Authentication

- Broker sends a long number powered by secure random numbers generator
- Client encrypt it with **AES** by means of **SHA-256** of its password as key and sends cipher text obtained to the broker
- Broker compares the cipher text received from the client and that which has been computed from itself
    - If they are equal then **AUTHENTICATION SUCCESSFUL**
    - else **AUTHENTICATION DENIED**



Secure end-to-end communication [1/4]

- MQTT allows to encrypt data at application level
- Only trusted clients have access to the decryption key of the data.



[4] E2E Encryption Mechanism

Secure end-to-end communication [2/4]

In order to achieve this it has been used:

- A key exchange mechanism based on Elliptic Curve Diffie-Hellman (ECDH) to establish the symmetric key used by two clients
- AES 256-bit to encrypt the E2E communication

Secure end-to-end communication [3/4]

Let’s take for example two clients A and B.

Both, after authenticating, publish their public key on username/pubKey topic

When A wants to send a message to B:

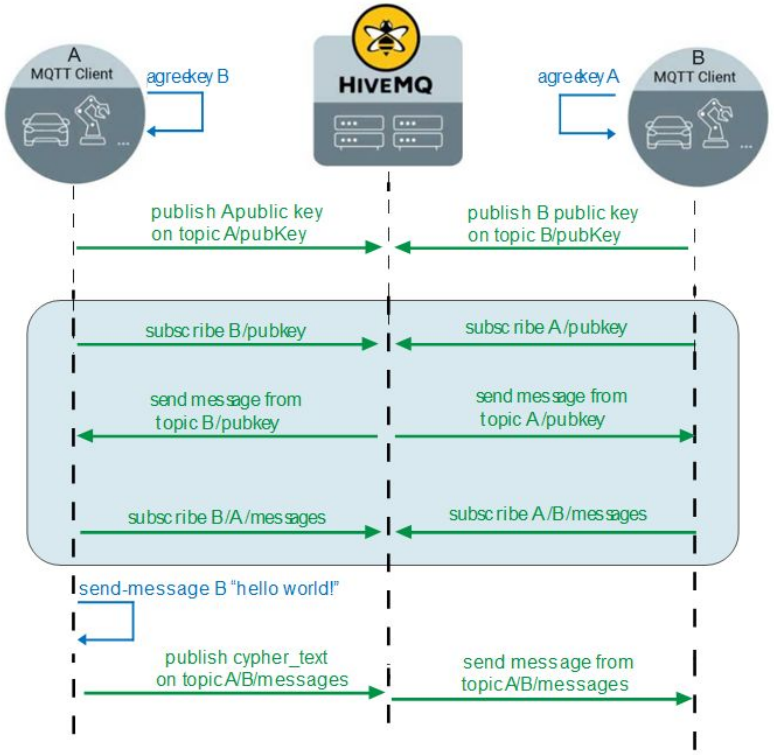
1. A must **agree the key** with B
    a. A subscribes itself to ***topic*** *B/pubKey* and to ***topic*** *B/A/messages*
2. A **publish** on *topic A/B/messages* the **encrypted message**

In this way B will receive A’s messages only when it will perform key agreement, then it will be able to decode them by means of agreed key.

Secure end-to-end communication [4/4]

Legend:

- In BLUE CLI command
- In GREEN MQTT operations



Code available at: https://github.com/francesboc/DSS-Project

Thanks for your attention

Questions?

References:

[1] https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/

[2] https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/

[3] https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/

[4] https://www.hivemq.com/blog/mqtt-security-fundamentals-payload-encryption/