MQTT-Auth: a Token-based Solution to Endow MQTT with Authentication and Authorization Capabilities

Marco Calabretta, Riccardo Pecori, Massimo Vecchio, and Luca Veltri

Original scientific paper

***Abstract*—Security in the Internet of Things is a current hot topic and it may comprise different aspects such as confidentiality and integrity of personal data, as well as the authentication and the authorization to access smart objects that are spreading more and more in our every-day lives. In this work we focus on MQTT (Message Queue Telemetry Transport), a message-based communication protocol explicitly designed for low-power machine-to-machine communications and based on the publish-subscribe paradigm. First of all, we provide an accurate analysis of some of the most recent security solutions and improvements of MQTT found in the literature. Secondly, we describe in detail a novel secure solution, called MQTT-Auth, to protect specific topics in MQTT. This solution is based on the AugPAKE security algorithm for guaranteeing confidentiality, and onto two tokens which permit to authenticate the usage of a topic and to guarantee authorization in accessing a topic respectively. MQTT-Auth can also be easily extended to a hierarchical structure of topics and entities. Finally, we compare MQTT-Auth with some solutions for securing MQTT being present in the relevant literature, and we provide some details on how MQTT-Auth has been implemented and successfully tested.**

***Index Terms*—Internet of Things, Security, MQTT, Publish-subscribe, token-based authentication, token-based authorization**

I. Introduction

THE envisioned worldwide Internet of Everywhere (IoE) paradigm is becoming a reality day after day, mainly thanks to the spreading of Smart Objects (SOs), such as sensors, smartphones, wearables, tablets, and the like, in almost every aspect of everyday life. According to various estimates, by 2020 billions of SOs are expected to be deployed within urban, home, industrial and rural scenarios, in order to collect relevant data, which may be used, in turn, to build new applications and innovative services for citizens, industries, institutions and many other stakeholders. This huge amount of information, coming from various and heterogeneous sources, can be considered, for sure, within the Big Data paradigm, as it is characheterized in terms of large volume, velocity, variety, veracity and value [1].

As this deluge of data continuously increases, concerns about their privacy, integrity and confidentiality, as well as authentication and authorization issues have to be carefully considered, together with possible countermeasures against well-known attacks SOs are particularly vulnerable to. Examples of these may encompass attacks aiming to exhaust the energy of nodes, such as unsolicited communications from spamming nodes [2], or to manipulate the network such as the intrusion of malicious nodes [3]. However, some of the most dangerous are the large Distributed Denial of Service (DDoS) attacks that have taken place in the last years, with unprecedented volumes of data used to knock-down various Internet services across the world, and that could interfere with important services provided by the Internet of Things, such as alerting in crisis scenarios [4]. The attack performed by the Mirai malware in 2016 [5], specifically designed to attack and hijack IoT devices and to transform them into bots, exploitable to carry out subsequent coordinated attacks, was only the first sign of a still active threat [6]. As a matter of fact, researchers of cyber-security have recently spotted new variants of Mirai which can easily dwarf it: they are *Okiru*[1], specifically designed to target devices running on ARC embedded processors, and *Satori*[2], which aims at zombifying Huawei routers by exploiting a zero-day vulnerability. In the light of what we have explained above, this research paper focuses on MQ Telemetry Transport (MQTT), a communication protocol originally developed by IBM in 1999, featuring low power consumption and minimal bandwidth requirements, that became an OASIS standard in 2014 [7]. MQTT is a binary protocol mainly designed for Machine-to-Machine (M2M) communications, with the aim to be lightweight and message-oriented, in order to transmit data using very few computational resources. It is a publish-subscribe protocol and it works well also in scenarios with small reliability, where the bandwidth is limited and latency may be high, since it guarantees the delivery of messages to all subscribers. MQTT is very suitable for an Internet of Things (IoT) environment also because of its support for various levels of Quality of Service (QoS), its inherent

Manuscript received September 24, 2018; revised October 17, 2018. Date of publication October 29, 2018. The associate editor Prof. Nikola Rožić has been coordinating the review of this manuscript and approved it for publication.

Marco Calabretta is with IQVIA, Milan, Italy. E-mail: marco.calabretta86@gmail.com.

Riccardo Pecori is with the SMARTEST Research Centre, eCampus University, Novedrate,CO, Italy. E-mail: riccardo.pecori@uniecampus.it.

Massimo Vecchio is with CREATE-NET FBK, OpenIoT Research Unit, 38123 Trento, TN, Italy. E-mail: vecchiomassimo@gmail.com.

Luca Veltri is with the Department of Engineering and Architecture, University of Parma, Parma, PR, Italy. E-mail: luca.veltri@unipr.it.

Digital Object Identifier (DOI): 10.24138/jcomss.v14i4.604

[1]http://www.csoonline.com/article/3247794/security/mirai-okiru-new-ddos-botnet-targets-arc-based-iot-devices.html

[2]http://thehackernews.com/2017/12/satori-mirai-iot-botnet.html

simplicity and lack of complex management, flexibility in the payload format, last-will-and-testament mechanism, possibility of broker federation, etc.

The official OASIS specifications for MQTT do not include mandatory security charachteristics, delegating them to the specific implementation. As a consequence, the contribution of this work, whose preliminary version was presented in [8], is to propose a novel secure communication scheme for MQTT, based on the augmented PAKE (AugPAKE) algorithm [9], as well as on an authentication token, for creating and publishing on a certain topic, and on an authorization token, used to grant access to a specific topic. The authorization token, which is required by the subscribers to access data coming from the publishers on that particular topic, is transmitted to the subscribers through a different secure side channel. Both tokens represent further authorization and authentication means, completely transparent for the broker, which is unaware, in advance, of the legitimate topics and subscribers.
Unlike the work in [8], both tokens have been also applied to a hierarchy of topics with the usage of wildcards; moreover, we also compare the proposed solution, called MQTT-Auth, with other solutions for securing MQTT presented in the relevant literature.
MQTT-Auth has been completely implemented in Java language by means of ActiveMQ [10] as Message-oriented Middleware (MoM), and in the end of the paper we also provide some information about the main modules used in the practical implementation.

The rest of the article has the following structure: in Sec. II we review the main contributions in the literature regarding the security mechanisms proposed for MQTT, in Sec. III we detail the exploited technologies (MQTT protocol and AugPAKE algorithm), while in Sec. IV we describe MQTT-Auth, its basic characteristics and its extension to a hierarchy of topics and of entities in detail. In Sec. V we describe some modules exploited in the practical implementation of MQTT-Auth, we evaluate its versions (basic and extended) and we compare it with other solutions presented in the literature. Finally, in Sec. VI we sum up the paper by means of some conclusions while highlighting some future research directions.

II. Related Works

As stated in Section I, the original standard specifications of MQTT do not provide mandatory security solutions. Only a sort of authentication is optionally possible by sending username and password in the initial connection phase between a node and the broker. Therefore, cybersecurity researchers started investigating some mechanisms for securing MQTT, especially in the last few years, when MQTT returned to the limelight as one of the protocols usable in an Internet of Things scenario, e.g., for machine-to-machine communications.

In [11] the authors try to create a secure end-to-end IoT environment. This is done by using also MQTT, together with Universal Asynchronous ReceiverTransmitter (UART) and RIME, as protocols to transmit medical data of patients. Confidentiality is reached through AES - GaloisCounter Mode with 256-bit key, however heavy key management is required and performed through Elliptic Curve Diffie-Hellman (ECDH) or HKDF.

The contribution of Singh in [12] is the proposal of a secure version of both MQTT and MQTT-SN (for sensor networks). This is achieved by means of Attribute Based Encryption (ABE), based both on a Key Policy (KP-ABE) and on a Ciphertext Policy (CP-ABE), and Elliptic Curve Cryptography. However, ABE presents a main drawback: it is not efficient, as it requires that nodes register with the broker by sending a set of attributes, something that may cause undesired overhead. Moreover, the publisher should design in advance the access policy based on an access tree and send it to the broker, acting as Public Key Generator (PKG), which is further involved in the process to generate the key policy in case of KP-ABE. CP-ABE introduces some overhead too, and the complexity of its scheme is higher concerning storage and computation. Finally, other minor challenges related to ABE are key coordination, key escrow as well as key and attribute revocation.

A similar security approach, based on attributes, for publish-subscribe messaging patterns can be found in [13]. Such a proposal is addressed towards group communication security, rather than to end-to-end communications like in our case, moreover it limits the number of attributes to $Olog(N)$, where N is the maximum number of members. The method is applied to a chat group and to a personal health scenario, but it requires a further entity, a so-called group controller, which communicates with the MQTT network through HTTPS RESTful calls, something that can be a bottleneck of the whole communication.

Another solution featuring the same aforementioned limitations is the one proposed in [14]. The authors propose a composite security framework for MQTT by leveraging on Attribute-Based Encryption and AES S-boxes. This solution has the same drawbacks mentioned above, such as the overhead in setting up the access policy by an external trusted authority; moreover, it resorts to two different solutions, one based on public key cryptography and the other one based on secret key cryptography, and so a double decryption process on the part of the subscriber is necessary.

In [15] Identity-Based Cryptography (IBC) is employed to devise a secure publish-subscribe protocol based on MQTT. In this type of cryptography a sender, who can access the public parameters of the system, can encrypt a message using, for example, the text-value of the receiver's name or email address as a key. In turn, the receiver gets its decryption key from a central authority, which needs to be trusted as it generates secret keys for every user. As a consequence, in this solution further entities are necessary, in this case IoT gateways and external administrators, acting as private key generators and administrating different trust zones. Another drawback of the solution in [15] is that the proposed architecture features IoT gateways and devices having to handle two IBC-based private keys, and this generates further overhead.

The contribution in [16] tries to make MQTT secure by means of Access Control Lists (ACLs) embedded in a Mosquitto broker. However, this solution requires different usernames and passwords for different data and this would cause an unbearable kind of overhead, scaling with the number

of different data to be transmitted. In [17] another security solution for a Mosquitto MQTT implementation is proposed. It is based on Transpot Layer Security (TLS), like one of those surveyed in [18], embedded into security controller hardware. This is a limitation as it requires the modification of the physical controllers and leverages on the same Public Key Infrastructure necessary for software implementations of TLS.

In [19] an authorization mechanism for MQTT systems based on OAuth 1.0a is proposed. It is based on a fine-grained solution exactly like our proposal, i.e., it allows the authorization to access a single topic. However, it requires that clients generate new signatures at every request and that they make an extra step to ask an authorization server for a request token. Moreover, as already mentioned, this solution inserts a fourth actor in the MQTT scenario besides the broker, subscribers and publishers: an authentication and authorization server that creates an overhead both in terms of exchanged messages and of overall communication management. Finally, this mechanism provides only authorization and not confidentiality, while our solution features both of them.

Also the contribution in [20] highlights the vulnerabilities of the Internet of Things and provides an MQTT-based solution that employs RSA and Elliptic Curve cryptography. However, the solution is based on public key cryptography and on certification authorities to generate two kinds of certificates: one for the topics and one for the clients, and this turns to be again an overhead in terms of management and exchanged messages.

The contribution in [21] is the most similar to our proposal. It is based on Augmented Password-Only Authentication and Key Exchange (AugPAKE) [9] and it is called AugMQTT. It is quite efficient because it negotiates a session key between the publisher and the broker, and another session key between the broker and the subscriber, without requiring certificates neither their validation nor their revocation. The proposal is implemented through Mosquitto 1.4.9[3], while our solution is realized by means of ActiveMQ, whose advantages are detailed in Sec. V. Moreover, our solution provides also an authentication token to create and publish onto a certain topic and an authorization token that is known only by the publisher and some chosen subscribers via a secondary secure channel, for example shown visually through a display at the publisher's side. This is a solution similar to the one proposed in [22], where a side multimedia channel is employed to transmit and verify a Short Authentication String.

Also the work in [23] is similar to ours, but in this case in the usage of a token to authorize the access to a particular topic. However, the authors employ a fourth entity to deliver access tokens, a so-called authentication server. This leads to further overhead both in the management of this server and in the exchanged messages. The exchanged messages for setting up this security framework are 4-5, like in our proposal.

III. Overview of the Exploited Technologies

In this section, first we briefly illustrate the MQTT protocol, the communication paradigm it is based on and its packet format. Then we introduce the AugPAKE algorithm and detail its message exchange to setup a shared session key between two parties.

[3]http://mosquitto.org/blog/2016/06/version-1-4-9-released/

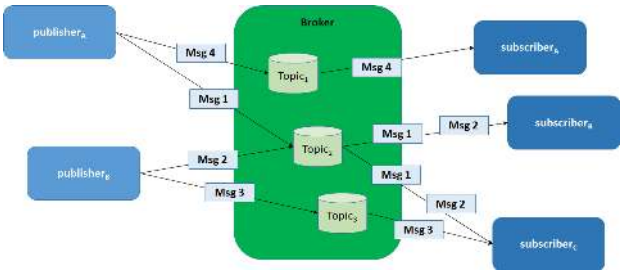

Fig. 1. Communication architecture in a typical MQTT scenario.

A. MQTT

Message Queue Telemetry Transport (MQTT) is a lightweight message-queuing communication protocol that relies on a publish-subscribe paradigm and works at application level. At transport layer, the communications between MQTT pairs are structured in sessions and are based on TCP connections. In this subsection we first describe the MQTT communication paradigm, the management of topics in MQTT and finally MQTT packet format.

1) MQTT communication: The publish-subscribe paradigm allows for asynchronous communications between a producer node (*publisher*), for example a sensor of temperature, and a consumer node (*subscriber*), e.g., a node employed by a user to know the temperature detected by the sensor. The communication is mediated by a third node, the so-called *MQTT broker*, which records a list of topics, receives data from the publisher nodes and, at the same time, delivers data to subscriber nodes. In practice, topics are queues of messages that allow the exchange of information with defined semantics. The overall communication scheme is depicted in Fig. 1, where it can be seen how publishers send messages onto different topics and subscribers retrieve data from different topics, all stored at the broker side.

In order to access the data in the broker, a SUBSCRIBE message is sent from a subscriber to the broker itself, specifying the requested topic. The broker, in turn, adds the requesting node to the distribution list of the nodes having access to that particular topic. This is done only if the requesting node fulfills the access policy for that particular topic, which by default allows every node.

Concerning the writing onto a topic, a PUBLISH message is sent from a publisher node to the broker. This allows a publisher to write data on an existing topic or to create a particular topic if this does not exist in the broker yet. Also in this case, by default every node is allowed to create any topic within the broker.

This communication architecture decouples client nodes (publishers or subscribers) and the single server node (the broker), both from a spatial and a temporal point of view. The broker is the only entity having full knowledge of the whole composition of the MQTT network.

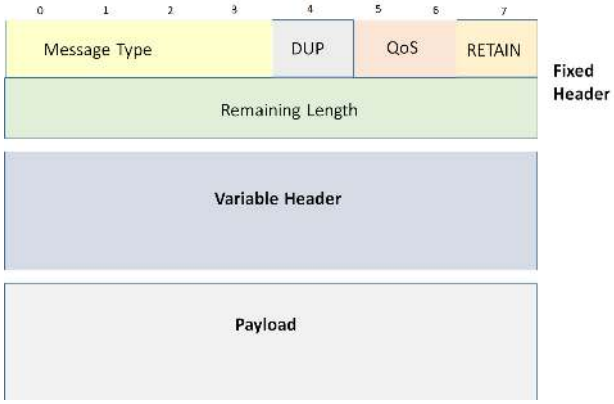

Fig. 2. Structure of an MQTT message.

2) MQTT topics: Subscribers can request to access more than one topic and topics may be organized into a hierarchical structure using slash separators (more or less like a path in a file system). In the hierarchical structure of topics, wildcards can be used, both singularly or jointly. There are two main wildcards in MQTT:

- *single level wildcard* (+): in this case, all subtopics mirroring the hierarchical structure of the filter are considered, whatever subtopic is put in the layer where the wildcard has been used. For example, a subscription to the topic $myhome/groundfloor/+/temperature$ should return the temperature data of whatever room is present at the ground floor.
- *multiple level wildcard* (#): in this case all subtopics are considered starting from the topic level that has been replaced by the wildcard. For example, a subscription to $myhome/groundfloor/\#$ would return all information available from all the sensors in all the rooms at the ground floor.

3) MQTT packet format: MQTT is a byte-based protocol and the complete structure of a packet is depicted in Fig. 2. An MQTT message is encoded by means of the network byte and bit ordering, and is subdivided into three parts:

- fixed header;
- variable header;
- payload.

The fixed header is mandatory and contains the following fields:

- the Message Type field, which identifies the type of the packet. Some examples of this could be a CONNECT message for setting up a session between a publisher or a subscriber and the broker, the PUBLISH message to store data at the broker or receive data from the broker, the SUBSCRIBE message to request from the broker the data contained into a certain topic, etc.
- the last four bits of the first byte, from 4 to 7, represents flags and they contain specific indicators for each packet type. They are reserved, except for the PUBLISH message when they are divided into the following subfields:
  - the 1-bit DUPLICATION flag which indicates that the message could have already been received;
  - two bits for codifying three different layers of QoS: at-most-once delivery, at-least-once delivery and exactly-one delivery;
  - the 1-bit RETAIN flag, which indicates to the broker to keep the message and the relative QoS for future subscribers.
- the Remaining Length field, indicating the number of remaining bytes in the message, i.e., the length of the optional Variable Header and of the optional payload, has a variable length ranging from 1 to 4 bytes.

The information contained in the Variable Header varies according to the message type and usually consists of additional control information. For example, in a CONNECT message the Variable Header contains the MQTT version, the flags to indicate whether username and password are carried in the payload, etc., while in a PUBLISH message the fields of the Variable Header are normally the following:

- the most significant byte of the length of the topic name string and message ID,
- the least significant byte of the length of the topic name string and message ID,
- the name of the topic.

The content of the payload depends on the application itself and it usually contains data to be transmitted or retrieved from a topic maintained at the broker. For example, a PUBLISH message contains data to be written into a certain topic at the broker side, a SUBSCRIBE message contains the list of topics from which the subscriber would like to receive data, etc.

4) MQTT security: As stated in previous sections, MQTT specifications in the OASIS standard do not include mandatory requirements for what concerns security aspects such as authentication, authorization, confidentiality, and the like. This lack is due to some peculiar aspects of MQTT:

- the focus on message dispatching rather than on other features;
- the attention to maintain the protocol as light as possible and with a small overhead;
- MQTT was initially employed only for telemetry purposes and these took place in private networks;
- MQTT is employed in very heterogeneous scenarios that have different security requirements.

The only optional possibility, allowed in the standard, to have a sort of authentication is the specification of a username and password during the initial connection phase between a node, be it a publisher or a subscriber, and the broker. The presence of these fields is notified to the broker by setting the relevant option flags in the variable header of the CONNECT message. However, even if this solution provides a certain degree of authentication, both the username and the password fields are transmitted in plain text, making the whole procedure very weak with respect to eavesdropping attacks, by using even a simple protocol sniffer.

Therefore, MQTT solutions usually rely upon the security services of other layers. As an example, OASIS explicitly suggest the use of TLS and certificates, whenever possible, in order to have a viable solution for authentication, data

integrity and confidentiality. However, this introduces additional workload and overhead to setup the secured connections and to cipher all traffic, and this is not opportune because of computational and power consumption limitations of MQTT devices.

Another possible solution, at network layer, would be to employ IPSec; however, also in this case, an extra overhead should be taken into account for extra headers to be added, even considering 6LowPAN compression capabilities.

Considering the application layer, MQTT brokers may decide to use a centralized authentication system, external to the broker itself. Concerning such applications, MQTT standard explicitly mentions the possibility of using LDAP or OAuth authentication systems. However, in both cases, a third-party external system is needed for authentication and granting access tokens. This solution introduces an overhead of communication and a further entity in the whole architecture, leading necessarily to an additional layer of complexity, both in terms of configuration and management.

B. AugPAKE

In this subsection we summarize the AugPAKE algorithm, which we exploit in MQTT-Auth as described in Sec. IV. AugPAKE [9] is a client-server session key establishment protocol, relying on some initial shared parameters such as: G, a cyclic group of prime order q, g, a generator of G, p, a prime number and such that $p = aq + 1$, with a an integer, H and H', two hash functions, with H' returning an integer and H returning a binary string of length k. The complete list of symbols used in the description of the algorithm is presented in Table I, together with a brief description of each notation.

TABLE I
NOTATIONS USED IN THE AUGPAKE ALGORITHM AND THEIR MEANING.

Symbol	Description
G	cyclic group of order q
g	generator of G
p	prime number such that p=aq+1, with a an integer
H	hash function returning a binary string of length k
H'	hash function returning an integer
C	client identifier
S	server identifier
$\|\|$	juxtaposition or concatenation of strings

AugPAKE consists of two main phases: a *setup phase*, when the client, having client identifier C, transmits securely to the server, having server identifier S, the client's password w, and an *execution phase*, which allows both parties to agree on a common shared session key.

The setup phase is composed of a single step: the client C computes the quantity W according to Equation 1

$$W = g^{w'} \bmod p \tag{1}$$

where $w' = H(C||S||w)$ and w is client C's password, and it registers W at the server S by sending a message containing both C and W.

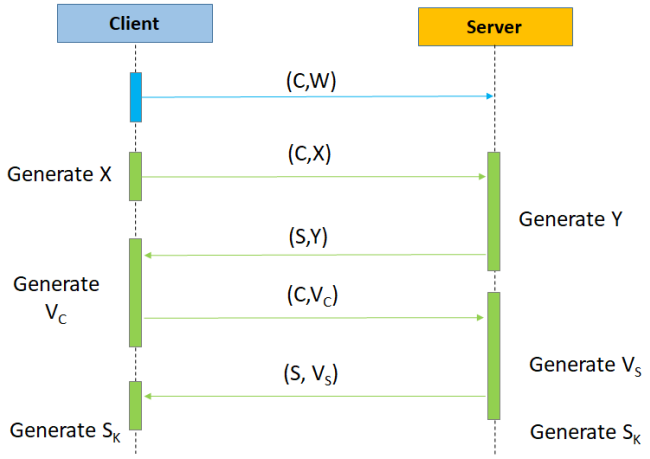

Fig. 3. AugPAKE message exchange [8]. In light blue the first setup phase and in green the second execution phase.

The execution phase involves several steps, as follows:

1) client C chooses x, computes X according to the Diffie-Hellman procedure reported in Equation 2:
$$X = g^x \bmod p, \tag{2}$$
and sends (C, X) to the server;
2) if server S receives 0, 1 or -1 (mod p) as X, it discards the message, otherwise it chooses y and computes Y according to Equation 3:
$$Y = (XW^r)^y \bmod p, \tag{3}$$
where $r = H'(\text{“01”}, C, S, X)$. The server sends (S, Y) to the client;
3) the client discards the just received message if Y is 0, 1 or -1 (mod p). Otherwise C computes K according to Equation 4:
$$K = Y^z \bmod p, \tag{4}$$
where $z = 1/(x + w'r) \bmod q$, and the authenticator V_C according to Equation 5:
$$V_C = H(\text{“02”}||C||S||X||Y||K), \tag{5}$$
which is sent to server S;
4) if the received V_C is not equal to $H(\text{“02”}||C||S||X||Y||K)$, with K computed by the server according to Equation 6:
$$K = g^y \bmod p, \tag{6}$$
S stops the procedure. Otherwise it generates authenticator V_S according to Equation 7:
$$V_S = H(\text{“03”}||C||S||X||Y||K), \tag{7}$$
sending it to the client C afterwards. Moreover, the server computes the session key SK according to the formula in Equation 8:
$$SK = H(\text{“04”}||C||S||X||Y||K); \tag{8}$$
5) finally, if client C receives V_S differing from $H(\text{“03”}||C||S||X||Y||K)$ it terminates the procedure,

otherwise it computes the session key SK always according to Equation 8.

It should be noted that the procedure works because in arithmetic modulo p it holds the equality in Equation 9:

$$Y^z = g^{(x+w'r)yz} = g^{(x+w'r)y\frac{1}{x+w'r}} = g^y \quad (9)$$

The complete AugPAKE procedure is described in Fig. 3.

IV. MQTT-AUTH

The proposed security solution for MQTT, called MQTT-Auth, takes advantage of AugPAKE in order to set up a secure session key and exploits standard MQTT message exchanges, by enriching them with the authentication of the usage of the topic, the authorization of the subscribers and the confidentiality of the transmitted data, without requiring the use of TLS (and the corresponding processing load) nor of other solution relying onto further external entities. The detailed description of the basic MQTT-Auth, as well as its extension to a hierarchy of topics and of entities, is described in the following subsections.

A. Basic version of MQTT-Auth

First of all, the AugPAKE procedure is executed for every publisher and every subscriber connecting to the broker. In particular, the MQTT broker acts as an AugPAKE server, while publishers and subscribers act as clients. AugPAKE messages are directly encapsulated into the payload of MQTT PUBLISH packets, so a session, previously established by means of CONNECT messages, between a publisher and a broker as well as between a subscriber and a broker has to be already active.

Publishers and subscribers communicate with the broker using the *AuthenticationTopic*, a new topic that is always active within the broker itself. Conversely, the broker communicates with publishers and subscribers via a temporary topic, named after the $Client_{ID}$ exchanged in the standard CONNECT phase; this topic is created at run-time and destroyed once the session key has been successfully created. These two topics are used as a bidirectional communication channel between the broker and the corresponding AugPAKE client (be it a publisher or a subscriber). This mechanism permits to maintain the standard two-message MQTT CONNECT phase unchanged, while the five-message exchange of AugPAKE, described in Subsection III-B is mapped onto standard MQTT PUBLISH messages. The mapping of the AugPAKE procedure into MQTT messages is depicted in Figure 4, where, for the sake of simplicity PUBACK messages and SUBACK messages are omitted.

After this phase, each client node i, be it either a publisher or a subscriber, shares a common secret session key SK_i with the broker. At this point, whenever a publisher wants to publish data on the broker related to a certain topic, it performs three actions in the following sequence:

- It generates the new *topic authentication token*, called $Auth_1$, which is a MAC of the topic computed with the session key SK_P shared with the broker, base-64 encoded. $Auth_1$ is computed according to Equation 10, is concatenated with the original topic into a unique string and sent in the *topic name* field of the variable header of PUBLISH messages according to Equation 11:

$$Auth_1 = MAC(SK_P, topic) \quad (10)$$

$$topic\,name \leftarrow topic\$Auth_1, \quad (11)$$

  where \$ is a proper separator character. Differently from standard MQTT, a broker records only topics authenticated by means of the MAC in $Auth_1$.
- It computes a further *authorization token*, called $Auth_2$, as a MAC of the topic, a sequence number (SN), received by the broker and associated to the $Client_{ID}$ of the subscriber as a protection against replay attacks (attempts to reuse $Auth_2$ by other clients), and optionally the ID of a possible subscriber ($Client_{ID}$). The MAC is computed using the key SK_P according to the formula in Equation 12, while the ID of the subscriber is the same as the one exchanged by the client itself and the broker during the CONNECT phase.

$$Auth_2 = MAC(SK_P, topic||SN[||Client_{ID}]). \quad (12)$$

  $Auth_2$, together with the sequence number, is communicated to all interested subscribers via a secondary secure channel. An example of this channel could be a visual means, i.e., the token is shown on the display embedded in the device itself acting as a publisher, e.g., through a QR code or other means. In case of possible multiple subscribers, different $Auth_2$ tokens could be generated using different SNs.
- It starts sending data to the broker encrypting the payload of the messages through the session key SK_P, previously exchanged with the broker itself, and using the actual topic concatenated with $Auth_1$ as the topic in the corresponding *topic name* field of the variable header.

On the other hand, whenever a subscriber node wishes to access the data of a particular topic, after obtaining an authorization token $Auth_2$, it sends a standard SUBSCRIBE request to the broker, with the *topic name* field filled in with the concatenation of the actual topic and the SN, together with $Auth_2$. Hence, the topic field of a SUBSCRIBE message is composed as shown in Equation 13:

$$topic\,name \leftarrow topic\$SN\$Auth_2. \quad (13)$$

At this point the broker performs the following steps:

- It reads the topic, the SN and the authorization token present in the *topic name* field of the SUBSCRIBE message, and verifies the authorization token by using the shared key SK_P. This guarantees the subscriber can access the requested topic.
- Should the previous step succeed, it starts to deliver the data of the topic to the authorized subscriber by encrypting the payload of the relative PUBLISH messages with the session key SK_S, shared with the subscriber itself, using an authenticated encryption (AE) algorithm.

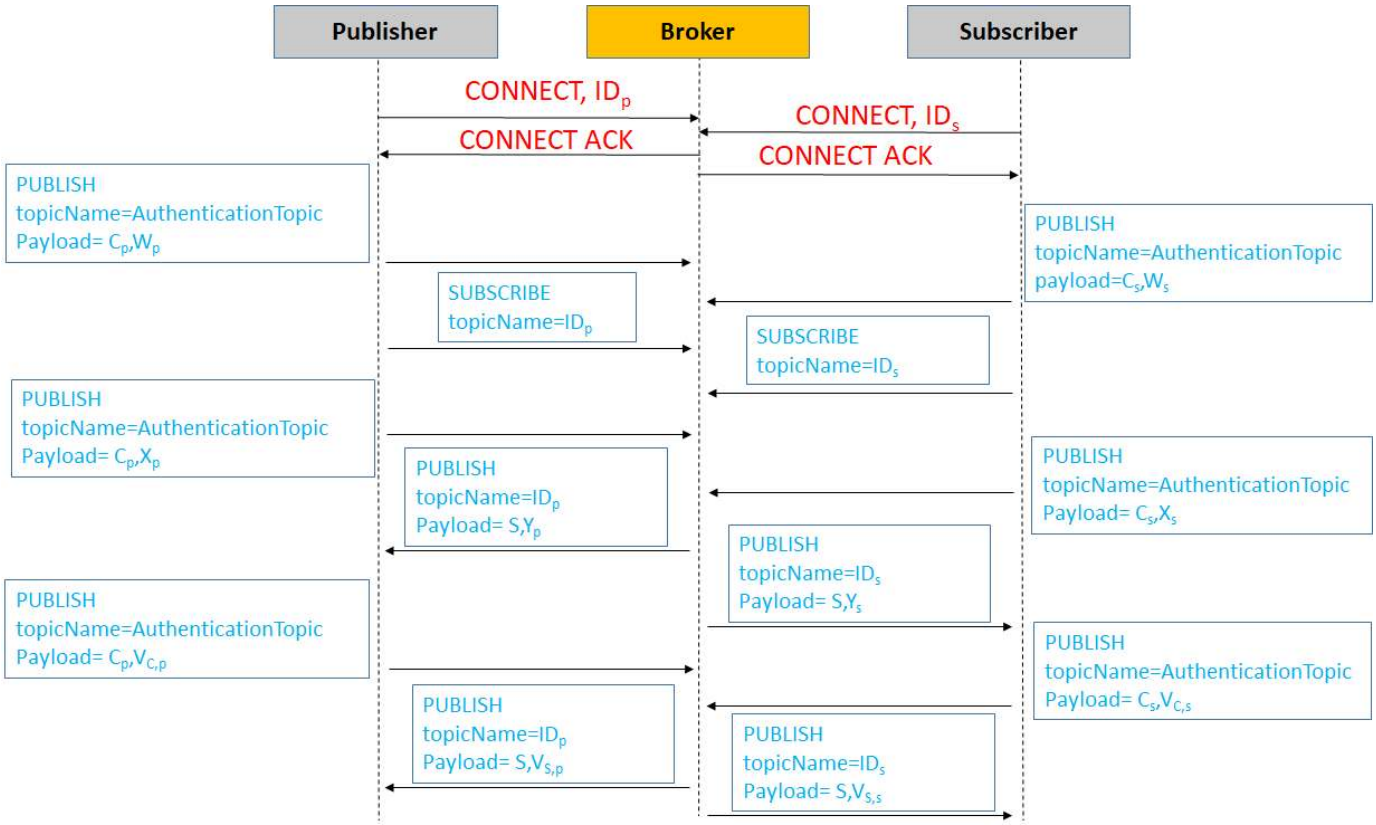

Fig. 4. AugPAKE session key establishment encapsulated into MQTT messages.

The payload of these PUBLISH messages is so defined according to the following equation:

$$payload \leftarrow AE(SK_S, payload). \tag{14}$$

The overall message exchange of MQTT-Auth is shown in Fig. 5 and allows:

- the broker to authenticate the topics published by known publishers;
- the publisher to ensure confidentiality of the data transmitted to the broker and delivered from the broker to the subscribers;
- the broker to guarantee authorization and access control, by means of the authorization token, so that only subscribers having a matching authorization token $Auth_2$ can access the corresponding topic of a given publisher.

The broker could be a single point of attack, but it does not have to know in advance which are the subscribers authorized to access a certain topic. From the confidentiality point of view, the broker should be considered as a trusted entity, since it has complete visibility of the exchanged data, as it is in charge of decrypting and encrypting MQTT payloads. However, data can be stored (cached) by the broker in an encrypted format, while decryption and re-encryption (with the subscriber key) can be done only when data have to be relayed to an authorized subscriber.

In this basic version of MQTT-Auth only the creator of a certain topic, by means of $Auth_1$, is allowed to write on that particular topic at the broker, i.e., we exploit only 1 to N communication and not N to N communication.

B. MQTT-Auth for a hierarchy of topics and of entities

In this subsection we describe the extension of basic MQTT-Auth to the hierarchical structure of topics described in Section III-A. The mechanism we are going to describe extends the use of the authentication token $Auth_1$ and of the authorization token $Auth_2$. Moreover, this extended version of MQTT-Auth permits also to exploit the full $N - to - N$ communication provided by the standard publish-subscribe paradigm described in Figure 1. This was not possible for the basic version of MQTT-Auth described in subsection IV-A, where only $1 - to - N$ communication was allowed.

1) Extension for the authentication token: The first modification to the basic MQTT-Auth is in the content sent in the *topic name* field of PUBLISH messages sent from a publisher to the broker. In the basic version this field contained only the topic concatenated with $Auth_1$ and this limited the possibility to publish onto the topic only to the creator of the topic itself. In this extended version the *topic name* field is composed as in Equation 15:

$$topic\,name \leftarrow topic\$list_{ID}\$Auth_1, \tag{15}$$

where $list_{ID}$ represents the list of the $Client_{ID}s$ of the authoritative entities that can publish on that topic. Usually, these entities are the ancestors of the current publisher in a hierarchy of entities such as the one depicted in Figure 6,

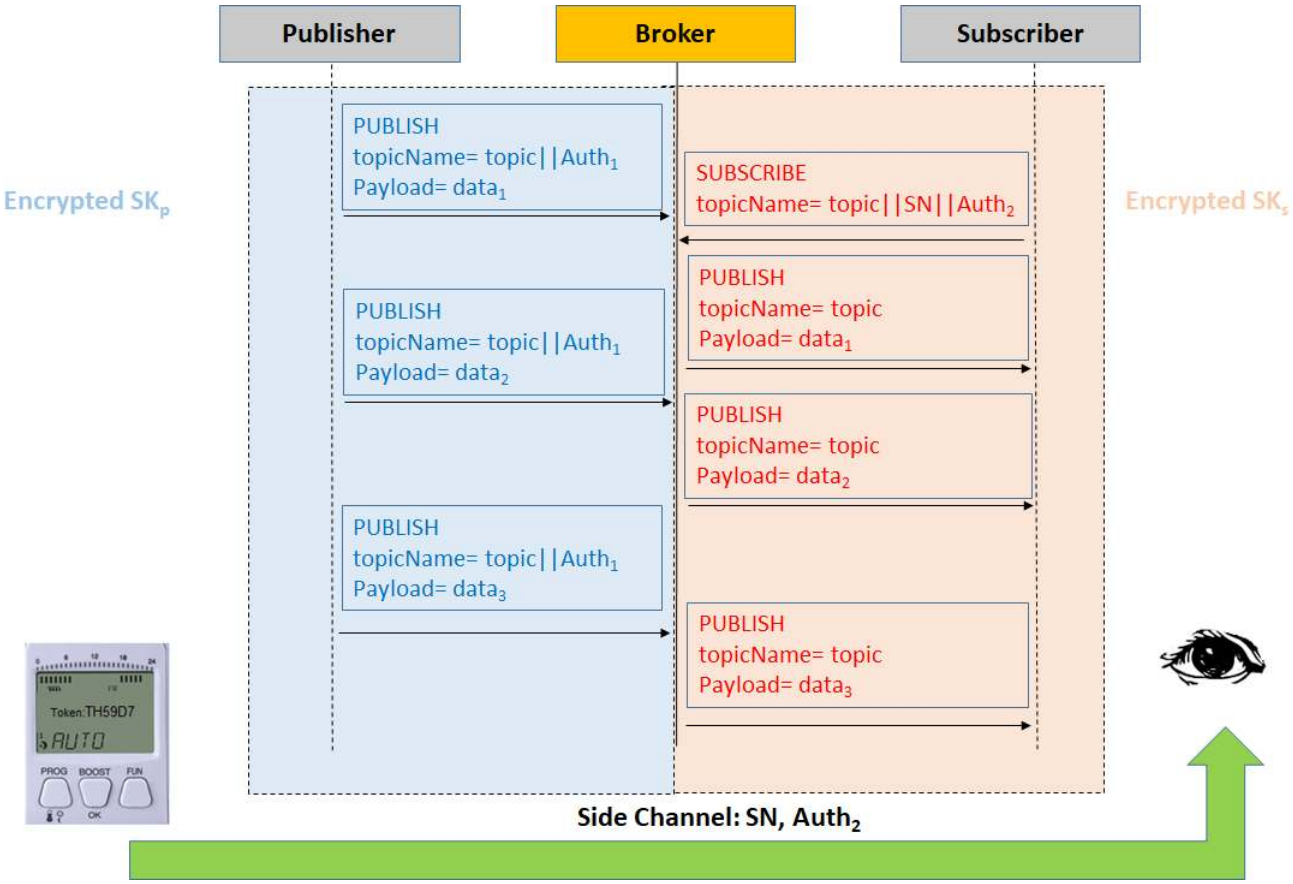

Fig. 5. MQTT-Auth message exchange. The encryption with session key of the publisher is in light blue, while the encryption with session key of the subscriber is in pink.

where the leaves represent the actual topics and the internal nodes and the root correspond to publishing entities.

2) Extension for the authorization token: Like in the basic version of MQTT-Auth, a publisher can issue $Auth_2$ tokens and SNs for different topics, since, as can be inferred from Figure 6, the same device can transmit information about different topics (e.g., temperature and pressure).

Unlike the basic version, not only a terminal device, in the example in Figure 6 a fridge or an oven, can issue the authorization token, but also entities of upper levels such as a kitchen or a living room. The computation of an authorization token $Auth_2$ on a topic string containing wildcards is allowed only for entities in the tree hierarchy that are at least one level up with respect to the first usage of a wildcard. As an example, let us consider the topic string in Equation 16:

$$/home/kitchen/+/temperature \tag{16}$$

For such a topic string only the kitchen or the home can provide a usable $Auth_2$ token. In this case, $Auth_2$ is computed onto the string containing the wildcard and by means of the proper session key that the authorizing entity has shared through AugPAKE with the broker.

In this extended scenario, the subscriber can employ the authorization token received from an upper entity to access different topics belonging to entities at lower levels. For example, if $Auth_2$ is computed onto the topic string in Equation 16, this will allow a subscriber to access the information, stored at the broker side, about the temperature of both the fridge and the oven.

Given such a scheme, it is necessary, for the broker, to know

1) which is the actual topic the subscriber wants to access,
2) which is the topic string the authorization token $Auth_2$ has been computed on,
3) whose is the session key to verify the authorization token $Auth_2$.

As a matter of fact, the actual topic the subscriber would like to access may be different and more detailed than the topic string used for computing $Auth_2$. As an example, a subscriber would like to access only the temperature of the fridge, but the token he possesses was computed over the string in Equation 16, which comprises a wildcard and is so more general, allowing one to access the temperature also of the oven. Moreover, the broker needs to know who is the owner of the session key used to compute $Auth_2$.

As regards the first two points, a possible solution would be to insert in the *topic name* field of the SUBSCRIBE message both topic strings: the subscribed one ($topic_{sub}$) and the more general one ($topic_{auth}$) used to compute $Auth_2$. In this case, the topic string in Equation 13 would be replaced by the concatenation of two *topic strings*:

$$\begin{aligned} topic\ name \leftarrow topic_{sub}\$topic_{auth}\$SN\$ \\ \$MAC(SK_P, topic_{auth}||SN[||Client_{ID}]), \end{aligned} \tag{17}$$

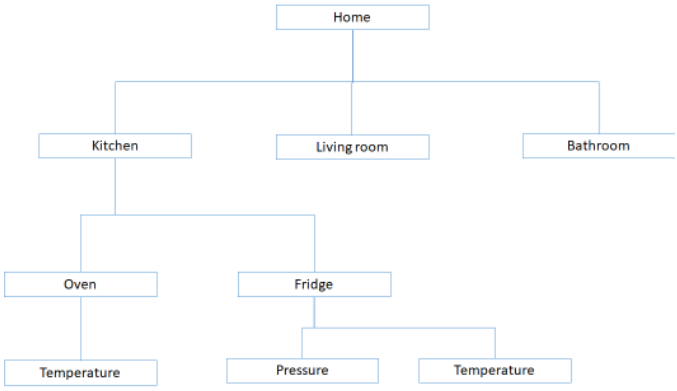

Fig. 6. Example of hierarchy of publishing entities and topics.

However, given the lightness nature of MQTT, we preferred to rely onto a more compact solution that allows the broker to verify correctly the authorization token $Auth_2$, without sending $topic_{auth}$ explicitly, and maintains the number of exchanged bytes as low as possible. The devised solution replaces the aforementioned $topic_{auth}$ with a sort of regular expression which allows one to reconstruct easily the whole $topic_{auth}$ string. This regular expression is made of the wildcard symbols described in Section III-A and the corresponding level index in the tree hierarchy, where level 0 corresponds to the root node (e.g., the Home in Figure 6). Some examples of possible regular expressions, together with their overall meaning, obtained by composing the regular expression and the knowledge of the hierarchy of entities, are reported in the following:

$$\begin{aligned} +1+2\#3 &\rightarrow /home/+/+/\# \\ \#2 &\rightarrow /home/kitchen/\# \end{aligned} \tag{18}$$

Therefore, a possible *topic name* field received by a broker could assume the form described in Equation 19:

$$\begin{aligned} topic\,name \leftarrow topic_{sub}\$reg_{exp}\$SN\$ \\ \$MAC(SK_P, topic_{auth}||SN[||Client_{ID}]), \end{aligned} \tag{19}$$

where $topic_{sub}$ is the complete topic path the subscriber would like to access, while reg_{exp}, together with the information in $topic_{sub}$, allows the broker to reconstruct the topic string ($topic_{auth}$) that was used by the authoritative entity to compute $Auth_2$. As an example, if $topic_{sub}$ corresponds to the string $/home/kitchen/fridge/temperature$ and reg_{exp} corresponds to the string $+1\#3$, then $Auth_2$ has been computed on a topic string with the following structure:

$$/home/+/fridge/\#. \tag{20}$$

Concerning the third point, we devised to introduce a further sub-field in the *topic name* of the SUBSCRIBE message, containing the $Client_{ID}$ of the authoritative entity whose session key was used to compute $Auth_2$. In such a way, the broker, can easily retrieve the session key, without relying onto the association between a created topic and a $Client_{ID}$. Given this last modification, the complete *topic name* field can be rewritten as in Equation 21:

$$\begin{aligned} topic\,name \leftarrow topic_{sub}\$reg_{exp}\$Client_{Auth}\$ \\ \$SN\$MAC(SK_P, topic_{auth}||SN[||Client_{ID}]), \end{aligned} \tag{21}$$

where $Client_{Auth}$ is the $Client_{ID}$ of the authoritative entity whose session key has been used to compute $Auth_2$.

V. Implementation and Evaluation

In this section we provide information about the practical implementation of MQTT-Auth, the evaluation of some of its performances and a comparison with other security solutions for MQTT present in the literature.

A. Practical implementation

The solution we propose has been successfully implemented and its source code is available at the project web page[4]. This implementation, based on Apache ActiveMQ as message-oriented middleware, exploits one of the most popular and powerful open source Message and Integration Patterns server written in Java. It provides a system integrating basic standards in a cross language way, by supporting Java, C, C++, Python, etc. ActiveMQ provides its services by employing Java Message Service (JMS) specifications such as clustering, group messages, Spring framework support and Enterprise containers integration. Moreover, it fully supports MQTT v. 3.1.1, employed in the project, which is extensible through plugins and whose brokers can be federated, and it can be integrated in the most common containers such as JBoss, TomEE, WebLogic.
Since ActiveMQ supports Spring, it is possible to configure the broker through XML, making this process simpler and more compressed. Also the management of security is much more flexible and pluggable compared to the Mosquitto-based solution presented in [21]. One of the most used plugin of ActiveMQ is JAAS, but it is possible to make extensions on the basis of particular requirements. Moreover, in ActiveMQ, it is available a *SimpleAuthentication* plugin, with which it is possible to define users and groups directly in the broker configuration. Over such users it is possible to define various authorization mechanisms afterwards. Furthermore, ActiveMQ permits mechanisms of authorization on single messages on the basis of the content, this is possible by defining in the broker configuration a new class that implements the *MessageAuthorizationPolicy* interface and overrides the *isAllowedToConsume* method.

We preferred to use ActiveMQ, rather than Mosquitto like in [21], because ActiveMQ has the chance to use the *mqtt+nio* connector that brings more scalability than the basic MQTT connector. Moreover, Mosquitto has the following drawbacks [24]:

- it is single-threaded and can suffer from high transmission latency;
- it cannot be configured to save every message before an acknowledgement has been sent to the publisher;

[4]http://netsec.unipr.it/project/mqtt-auth

TABLE II
CHARACTERISTICS OF THE EMPLOYED TESTBEDS.

	Testbed 1	**Testbed 2**
Hardware	Intel Core i7 6600U @ 2.6Ghz 4 core	UDOO Freescale i.MX 6 ARM Cortex-A9
Primary memory	16 GB	1 GB
Operating system	Windows 7, 64bit, SP1	UDOOUbuntu
Java version	10	10
MQTT version	3.1	3.1
Broker version	Apache ActiveMQ 5.15.2	-
Client	fusesource mqtt-client 1.6	fusesource mqtt-client 1.6

- it disables persistence by default avoiding synchronization to the disk. As a consequence some messages can be lost in case of system crash;
- the socket buffer size cannot be configured.

As for the symmetric encryption, we used AES with a 256-bit key, since it is often already implemented, in hardware, within IoT devices.

B. Evaluation

We evaluated the performances of our implementation of MQTT-Auth considering the following metrics:

1) the overall time required by the procedure, from the CONNECT phase to the delivery to a subscriber of the data published onto a certain topic;
2) the time the broker spent in order to manage a SUBSCRIBE request (reception of the SUBSCRIBE, validation of the $Auth_2$ token, decryption of stored data, re-encryption of the stored data by means of SK_s and delivery of the message to the subscriber);
3) time of creation of the shared session key.

The first two metrics have been computed both for the basic version of MQTT-Auth, described in subsection IV-A, and for the extended version described in subsection IV-B.

We considered two testbeds, whose characteristics are presented in Table II. In the first testbed we used three laptops, connected via 802.11n wireless connections, representing a broker, a publisher and a subscriber respectively; while in the second testbed we used a laptop of testbed 1 as a broker, and two UDOO[5] microcontrollers, with the features presented in the testbed 2 column of table II, as subscriber and publisher. The microcontrollers had a 802.11g wireless board.

The numerical results of the three aforementioned metrics for both considered tesbeds are shown in Table III, and they represent the average over 10 repetitions of a single MQTT-Auth procedure. As regards the timing of the first two metrics, the considered payload was of $2K$ bytes. As one can see, the extended version takes more time both in the overall communication procedure and in the management of the SUBSCRIBE message, which is quite time consuming in the extended version. What is interesting is also that using testbed 2 the performances do no worsen too much, with only more or less 1 second of difference compared to testbed 1.

[5]https://www.udoo.org/

TABLE III
EXECUTION TIMES (ms) FOR DIFFERENT PARTS OF MQTT-AUTH, BOTH IN ITS BASIC AND EXTENDED VERSION.

	Testbed 1		**Testbed 2**	
Metric	**Basic**	**Extented**	**Basic**	**Extended**
Overall time	2792	2994	3890	4091
SUBSCRIBE time	234	436	234	436
Session Key time	2555	2555	3652	3652

TABLE IV
COMPARISON OF DIFFERENT SECURITY TECHNIQUES FOR MQTT CONSIDERING THE NUMBER OF EXCHANGED MESSAGES TO ESTABLISH THE SHARED SESSION KEY. NUMBERS WITH THE * SIGN REPRESENT ESTIMATIONS AS DESCRIBED IN THE TEXT.

	MQTT-Auth	Aug-MQTT	SMQTT
N. of messages	6	5*	4-5*

C. Comparison with other Security Solutions for MQTT

In this subsection we compare the proposed MQTT-Auth to two other security solutions for MQTT, already described in section II. The considered techniques are: AugMQTT, presented in the contribution in [21], and SMQTT, described in [12]. The first one is the one bearing the most similarities to our proposal, but it grants only confidentiality, whereas the second one guarantees both confidentiality and authorization, but it requires to decide in advance the access policy and requires further overhead at the broker side, both for the PKG function it has to perform and in order to manage the attributes and their parameters.

The comparison described in Table IV refers to the number of messages necessary to establish the session key between a publisher/subscriber and a broker. As it may be inferred, our solutions requires the five AugPAKE messages, encapsulated in PUBLISH messages, plus a further message for subscription to the topic named after the $Client_{ID}$ (see Fig. 4). The contribution in [21] does not describe in detail the implementation details, but it is supposed to have at least the five messages of the AugPAKE algorithm. Also the description of SMQTT in [12] does not describe in detail the implementation, but from the figures in the paper at least three messages are necessary (registration of the URI, sending of the access tree and a generic "key management"). Considering that in SMQTT the key management setups a shared session key through the Decisional Diffie-Hellman algorithm, this would result in at least other two or three messages, and this leads to the estimated number of 4-5 messages. MQTT-Auth possibly introduces a further message compared to other security solutions present in the literature, however it guarantees more security features compared to AugMQTT, and less overhead at the broker side compared to SMQTT.

VI. CONCLUSION

In this paper we have introduced a novel security mechanism for MQTT environments, named MQTT-Auth, which is based on AugPAKE, on an authentication token and on an authorization token. The latter is known only to the publisher

node and to the subscriber nodes via a secure side channel. This secure secondary channel can be for example a visual display located onto the publisher node itself, e.g., a smart fridge. Both the authentication and the authorization token are transported in the same field of the topic name. Also the initial phase of setting up the session keys between broker and publisher, as well as broker and subscribers, is encapsulated into standard MQTT messages so the required overhead is very limited. The proposed solution has also been extended in order to be usable in a hierarchy of topics and implemented by means of the ActiveMQ middleware and Java programming language. The performance study and the comparison of MQTT-Auth with other security solutions for MQTT, found in the relevant literature, show promising results.

Some possible future developments may concern the setting of a validity period for the session key, so allowing for a re-keying procedure, as well as evaluating the implemented solution both in terms of energy consumption and of latency time, in order to estimate the complete overhead of MQTT-Auth. Finally, we are working on a further development that will allow to apply MQTT-Auth also to a federation of brokers and to extend the hierarchical management of authorization tokens to a system involving also a hierarchy of keys.

ACKNOWLEDGMENT

The authors would like to thank Mr. Antonio Enrico Buonocore for carefully proofreading the paper.

REFERENCES

[1] Ducange P., Pecori R. and Mezzina P., "A glimpse on big data analytics in the framework of marketing strategies," in Soft Computing (2018), Vol. 22(1), pp. 325–342. DOI: 10.1007/s00500-017-2536-4

[2] Ajmal M. et al., "privy: Privacy-Preserving Collaboration Across Multiple Service Providers to Combat Telecom Spams," in IEEE Transaction on Emerging Topic Computing, 2018. DOI: 10.1109/TETC.2017.2771251

[3] Arshad J. et al., "COLIDE: A Collaborative Intrusion Detection Framework for Internet of Things," in IET Networks, 2018. In press.

[4] Mongiello M., et al. "A Smart IoT-Aware System For Crisis Scenario Management," Journal of Communications Software and Systems, vol. 14, n. 1, p. 91-98, Apr. 2018. DOI: 10.24138/jcomss.v14i1.533

[5] Perrone G., Vecchio M., Pecori R. and Giaffreda R., "The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices," in Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security - Volume 1: IoTBDS 2017, ISBN 978-989-758-245-5, pp. 246-253. DOI: 10.5220/0006287302460253

[6] Kolias C., et al., "DDoS in the IoT: Mirai and Other Botnets," in Computer, vol. 50, no. 7, pp. 80-84, 2017. DOI: 10.1109/MC.2017.201

[7] MQTT Version 5.0 specifications. Available at: http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. Accessed: January, 2018.

[8] Calabretta M., Pecori R., Veltri L., "A token-based protocol for securing MQTT communications," SoftCOM 2018, 26th International Conference on Software, Telecommunications and Computer Networks, Split-Supetar, Croatia, 13-15 September 2018. In press.

[9] S. H. Shin and K. Kobara, "Efcient Augmented Password-Only Authentication and Key Exchange for IKEv2," IETF RFC 6628, Experimental, June 2012. Available at https://tools.ietf.org/rfc/rfc6628.txt

[10] Apache ActiveMQ homepage. Available at: http://activemq.apache.org. Accessed: January 2018.

[11] A. Mathur, T. Newe, W. Elgenaidi, M. Rao, G. Dooly and D. Toal, "A secure end-to-end IoT solution," Sensors and Actuators A: Physical (2017), Vol. 263, pp. 291-299, DOI: 10.1016/j.sna.2017.06.019

[12] M. Singh, M. A. Rajan, V. L. Shivraj and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, 2015, pp. 746-751. DOI: 10.1109/CSNT.2015.16

[13] D. Thatmann, S. Zickau, A. Förster and A. Küpper, "Applying Attribute-Based Encryption on Publish Subscribe Messaging Patterns for the Internet of Things," 2015 IEEE International Conference on Data Science and Data Intensive Systems, Sydney, NSW, 2015, pp. 556-563. DOI: 10.1109/DSDIS.2015.52

[14] L. Bisne and M. Parmar, "Composite secure MQTT for Internet of Things using ABE and dynamic S-box AES," 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, 2017, pp. 1-5. DOI: 10.1109/IPACT.2017.8245126

[15] W. Peng, S. Liu, K. Peng, J. Wang and J. Liang, "A secure publish/subscribe protocol for Internet of Things using identity-based cryptography," 2016 5th International Conference on Computer Science and Network Technology (ICCSNT), Changchun, 2016, pp. 628-634. DOI: 10.1109/ICCSNT.2016.8070234

[16] Y. Upadhyay, A. Borole and D. Dileepan, "MQTT based secured home automation system," 2016 Symposium on Colossal Data Analysis and Networking (CDAN), Indore, 2016, pp. 1-4. DOI: 10.1109/CDAN.2016.7570945

[17] C. Lesjak et al., "Securing smart maintenance services: Hardware-security and TLS for MQTT," 2015 IEEE 13th International Conference on Industrial Informatics (INDIN), Cambridge, 2015, pp. 1243-1250. DOI: 10.1109/INDIN.2015.7281913

[18] Mahmoud Ammar, Giovanni Russello, Bruno Crispo, "Internet of Things: A survey on the security of IoT frameworks," Journal of Information Security and Applications, Vol, 38, 2018, pp. 8-27, DOI: 10.1016/j.jisa.2017.11.002.

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

[20] A. Mektoubi, H. L. Hassani, H. Belhadaoui, M. Rifi and A. Zakari, "New approach for securing communication over MQTT protocol. A comparison between RSA and Elliptic Curve," 2016 Third International Conference on Systems of Collaboration (SysCo), Casablanca, 2016, pp. 1-6. DOI: 10.1109/SYSCO.2016.7831326

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

[22] R. Pecori, and L. Veltri, "3AKEP: Triple-authenticated key exchange protocol for peer-to-peer VoIP applications," Computer Communications, Vol. 85, 2016, pp. 28-40, DOI:10.1016/j.comcom.2016.04.005

[23] A. Bhawiyuga, M. Data and A. Warda, "Architectural design of token based authentication of MQTT protocol in constrained IoT device," 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA), Lombok, 2017, pp. 1-4. DOI: 10.1109/TSSA.2017.8272933

[24] "Benchmark of MQTT servers", version 1.1, January 2015, mqtt.jorammq.com www.scalagent.com.

Marco Calabretta is a computer science engineer, currently working as Business Intelligence consultant. He graduated at Magna Graecia University in Catanzaro (Italy) with a bachelor degree in computer science and biomedical engineering, and a thesis entitled "Design of workflow systems for biomedical data analysis". He got his master degree from eCampus University in Novedrate, CO (Italy) in computer science and automation engineering. His thesis project describes a secure system for IoT device-to-device communications, entitled "Study and implementation of secure protocols for MQTT brokers". He worked for Accenture and he is currently working as consultant for IQVIA Italia.

Riccardo Pecori received the M.Sc. degree in Telecommunications Engineering (Magna cum Laude) from the University of Parma in 2007 and got his Ph.D. in Information Technology from the same university in 2011. After that he has been Adjunct Professor of various courses dealing with telecommunication networks, informatics, didactics of telecommunications, cybersecurity, etc., for both University of Parma and eCampus University. Since 2015 he has been Assistant Professor of Computer Science at eCampus University where he teaches Computer Security, Network Security, Internet of Things and Information Technology for Psychologists. Since April 2017 he has been editor of the journal "Future Generation Computer Systems". He has been Technical Program Committee member of various conferences dealing with computer science and telecommunications, organizing also a special session on "Social Internet of Things" at ISWCS 2017. His research interests regard network security, security in the Internet of Things, educational and social Big Data analysis as well as identification of relevant sets in complex systems.

Massimo Vecchio received the M.Sc. degree in Information Engineering (Magna cum Laude) from the University of Pisa and the Ph.D. degree in Computer Science and Engineering (with Doctor Europaeus mention) from IMT Lucca Institute for Advanced Studies, in 2005 and 2009, respectively. Starting from May 2015, he is an associate professor at eCampus University, while in September 2017 he has also joined FBK CREATE-NET to coordinate the research activities of the OpenIoT Research Unit. He is the project coordinator of AGILE (www.agile-iot.eu), a project co-founded by the Horizon 2020 programme of the European Union. His current research interests include computational intelligence and soft computing techniques, the Internet of Things paradigm and effective engineering design and solutions for constrained and embedded devices. Regarding his most recent editorial activity, he is a member of the editorial board of the Applied Soft Computing journal and of the newborn IEEE Internet of Things Magazine, besides being the managing editor of the IEEE IoT newsletters.

Luca Veltri is an assistant professor at the Department of Engineering and Architecture of the University of Parma, Italy, and he teaches classes on Communication Networks, and Network Security. He is also the director of the UniPR Co-Lab, a cross-department research center in educational technology at University of Parma. From 1999 to 2002, before joining the University of Parma, he has been with CoRiTeL, a research consortium founded by Ericsson Telecomunicazioni, where he leaded different research projects in networking and multimedia communications. He participated also in several research projects funded by the European Union, by the European Space Agency, and by the Italian Ministry of University and Research. His current research interests include Internet of Things, Software-Defined Networking, and Network Security. He is co-author of more than 70 papers on international conferences and journals. He is an IEEE Member.