MQTT-A: A broker-bridging P2P architecture to achieve anonymity in MQTT

MQTT-A: A broker-bridging P2P architecture to achieve anonymity in MQTT

Francesco Buccafurri, *Member, IEEE,* Vincenzo De Angelis, and Sara Lazzaro

***Abstract*—The demand for privacy in the current digital era is continuously growing. This is particularly true in the context of IoT, in which huge amounts of data are handled. Communication anonymity is a fundamental requirement when high privacy levels should be guaranteed. On the other hand, very little attention has been devoted to this problem in the past scientific literature, when referring to MQTT, which is the de-facto standard for IoT communication. In this paper, we try to cover this gap. Specifically, we propose a new protocol, called MQTT-A, which extends the MQTT bridging mechanism to support the anonymity of both publishers and subscribers. This task is accomplished through the P2P collaboration of intermediate bridge brokers, which forward the requests of clients so that the final broker cannot understand the actual source/destination. Moreover, an anonymity-preserving topic discovery mechanism is provided, which allows clients to discover available topics and associated brokers, preventing client identification. Importantly, all the MQTT-A messages are exchanged by leveraging standard MQTT primitives and the bridging mechanism natively offered by MQTT. This allows us not to require changes in the standard MQTT infrastructure. To validate the performance of our solution, we performed a deep experimental campaign by deploying the bridge brokers on cloud platforms in various countries of the world. The experimental validation shows that, the price of latency we have to pay because of the trade-off with anonymity is quite reasonable. Moreover, no significant impact on goodput occurs in the case of good network conditions.**

***Index Terms*—Anonymity, Privacy, IoT, MQTT bridging, P2P.**

I. Introduction

INTERNET OF THINGS [1] is an evolving paradigm in which smart objects are connected to each other to deliver services. Since IoT devices can be resource-constrained, traditional communication protocols such as HTTP cannot be adopted to connect them. Therefore, researchers proposed new lightweight protocols allowing communication in scenarios in which limited bandwidth is available and energy consumption is a serious issue. MQTT [2] is the most popular protocol in the IoT context. As highlighted by [3], MQTT is the best candidate for M2M communication due to its lightweight features and ability to work efficiently in low-power and limited memory devices as compared to its counterpart, CoAP. Specifically, MQTT has numerous applications such as home automation [4], sensor networks [5], smart manufacturing [6], and smart-city services [7]. Moreover, to address the issues related to smartphones such as their limitations in terms of battery life and network bandwidth, the Facebook social network [8] employs MQTT for instant messaging [9]–[12]. Also AWS (Amazon Web Services) IoT uses MQTT due to several features such as fault tolerance, support for intermittent connectivity, and high efficiency in terms of the network bandwidth consumption and device memory [13].

Despite its large adoption, MQTT presents several security and privacy issues [14]–[16]. Actually, as pointed out by [17], these issues concern the IoT context in general. Indeed, due to resource constraints, smart devices are unable to support highly secure protocols and cryptographic algorithms. This may in principle lead to serious threats, due to the pervasiveness of IoT devices and their interaction with the physical world. Specifically, at the application layer, [18] identifies four issues: mutual authentication [19]–[22], information privacy [23]–[27], data management [28]–[30], and application-specific vulnerabilities [31]–[33].

Most of the privacy-preserving solutions proposed in the literature are intended to ensure that users are aware of who is collecting their personal data and how data are collected and used [26].

Nonetheless in collecting them, third parties may also collect metadata related to communications with users' IoT devices. In fact, not concealing such metadata may create a huge privacy leakage. This is because metadata could allow third parties to link data from different IoT devices belonging to the same user. This way, third parties may be able to track and profile users, thus compromising their privacy.

Communication anonymity is a fundamental requirement when high privacy levels should be guaranteed. On the other hand, very little attention has been devoted to this problem in the past scientific literature, also regarding MQTT. In this paper, we try to cover this gap.

Specifically, we aim to prevent the identification of MQTT publishers and subscribers when sending/receiving data, in every phase of the protocol, including the topic discovery. Observe that the direct application of existing anonymous access to the network like Tor [34] or I2P [35] is not possible, as MQTT clients cannot support existing protocols of this family. This is due to the heavy computational effort and time overhead they require for operations such as layered encryption of messages and tunnel construction in the network. In addition, these protocols require clients to perform some operations using public-key encryption, which is much heavier

F. Buccafurri, V. De Angelis, and S. Lazzaro are with the DIIES Dept., University Mediterranea of Reggio Calabria, Via dell'Universita 25, 89124 Reggio Calabria, Italy
E-mail: {bucca,vincenzo.deangelis,sara.lazzaro}@unirc.it

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

than symmetric encryption.

On the other hand, for privacy reasons and censorship resistance, also in the context of IoT, anonymity requirements are becoming emerging, to the extent that IoT devices are more and more the digital counterpart of humans.

Therefore, a practical solution MQTT-compliant borrowed from ideas belonging to the field of anonymous communication networks [36] is highly desirable. This is the contribution of this paper.

According to the standard definition [37], there are three types of communication anonymity that we can reach. Sender anonymity means hiding the identity of a sender inside a set of potential senders. Similarly, recipient anonymity means hiding the identity of a recipient inside a set of potential recipients. Finally, relationship anonymity means hiding the fact that a given sender and a given recipient are communicating. It is easy to see that guaranteeing just one, between sender and recipient anonymity, is enough to obtain relationship anonymity.

In MQTT, publishers play the role of senders and subscribers play the role of recipients. Our solution reaches both sender and recipient anonymity, then relationship anonymity too. Furthermore, the proposed approach can be applied just from one side (publishers or subscribers) achieving just one between sender and recipient anonymity (but still enough to obtain relationship anonymity).

The problem with the standard MQTT approach is that clients communicate directly with the broker, thus it is able to infer important information about them [38].

To solve this problem, we take advantage of the bridging architecture offered by MQTT (see Section II for details). In this architecture, clients do not communicate directly with the broker that hosts the topics, but through an intermediate broker called *bridge broker*. This natively implements a weak form of anonymity, which is, in general, not sufficient. Indeed, the bridge broker merely forwards requests of clients without hiding some patterns (e.g., number of requests, topics of interest, and so on) sufficient for client re-identification.

The core of our proposal is to set up an anonymous peer-to-peer (P2P) network composed of bridge brokers such that the final broker cannot discover which actual bridge broker has started the communication. Our solution is inspired by [39]. We chose this protocol since it is lighter than alternative solutions (e.g., [34]) and, then, more suitable for IoT applications. Moreover, our solution is designed to be transparent to MQTT clients, leaving all the complexities to the brokers. This leads to two main advantages. First, clients can use our protocol regardless of whether they implement MQTT or MQTT-SN [40]. Second, also resource-constrained clients can leverage our protocol.

The experiments performed in Section VIII show that an (acceptable) price in terms of latency has to be paid if both sender and recipient anonymity are desired (and then relationship anonymity too). However, when just one between sender and recipient anonymity is enough (thus achieving also relationship anonymity), the latency required by our protocol is halved. In terms of goodput, no appreciable difference is observed in the case of good network conditions (i.e., low round-trip time between bridge brokers).

To provide a complete and effective solution, we did not neglect an aspect not strictly related to MQTT communication, but nevertheless of critical importance regarding anonymity. We refer to the problem of topic discovery, that is, how to allow clients to know which topics are available and the brokers that host them. This aspect is not treated in the standard MQTT protocol ([41], [42]), in which it is assumed that clients know in advance topics and where to find them. This can be true for some applications where publishers can advertise the topics to which they send data or subscribers can advertise their interest in a given topic. However, when anonymity is desired, this advertisement mechanism cannot be trivially adopted. Therefore, a suitable mechanism has to be investigated. This is another outcome of this paper.

To summarize, the main highlights of this paper are the following:

- We provide an anonymity protocol supporting publisher and subscriber anonymous communication with a remote broker. We call this protocol MQTT-Anonymous (MQTT-A).
- We define an anonymity-preserving discovery protocol that allows the clients to know the available topics and the brokers hosting them.
- All the exchanged information used to implement the above protocols is provided through standard MQTT messages. This avoids infrastructural changes in the architecture and requires no particular effort for clients and brokers.

The structure of the paper is the following. In Section II, we provide some background notions about the MQTT protocol and the bridging mechanism. The considered scenario and the motivations leading to this work are discussed in Section III. In Section IV, we propose the anonymity-preserving discovery protocol and in Section V, we provide the details about our anonymity protocol. In Section VI, we analyze the computational complexity and the overhead of the proposed approach. Some refinements of the protocol to manage QoS and path intersection are provided in Section VII. We perform an experimental validation in Section VIII by comparing MQTT-A with standard MQTT. We analyze the security of the proposed approach in Section IX. In Section X, we overview the literature about the security problems in MQTT. Finally, in Section XI, we draw our conclusions.

II. Background

MQTT is a client-server publish/subscribe messaging transport protocol [2]. Two kinds of agents are involved in the message exchange: MQTT clients and MQTT brokers. In turn, MQTT clients can be of two types: publisher (producer of information) and subscriber (consumer of the provided information). An MQTT client can play both roles of publisher and subscriber even at the same time. The MQTT protocol requires that the information provided by a publisher must be associated with a topic, which in general is used to categorize the information itself. Therefore, a subscriber can manifest

that it is interested in receiving certain information by just specifying its topic.

A topic can present one or more levels separated by a forward slash ('/'). This way, topics can be organized in a hierarchical structure. Nevertheless, there is no standardized semantic model for MQTT topics, therefore the name of a topic can be chosen freely by publishing or subscribing entities [43].

Unlike the classical client-server architecture, in the MQTT architecture, publishers and subscribers do not communicate directly between them, but through an MQTT broker. This mechanism is sketched in Figure 1. As depicted in the figure, first all the subscribers interested in a given topic (say T) should send the broker a subscribe message $sub(T)$ specifying such a topic (dashed arrows). Then, when a publisher sends a message to the broker, it specifies the information I and the topic T to which that information should be published. This message is reported in Figure 1 as $pub(T, I)$ (solid arrows). Then, the broker forwards the above message to all the subscribers interested in that topic.

Concerning MQTT messages, they present a limited amount of overhead, since the protocol header is only 2 bytes. Moreover, MQTT limits the payload dimension up to a maximum size of 256 MBytes. These constraints make MQTT suitable for low-bandwidth networks and resource-constrained clients, such as IoT devices.

Furthermore, in both publish and subscribe messages, an MQTT client can specify the desired Quality of Service (QoS) level. MQTT supports three different Quality of Service (QoS) levels (0,1, and 2). In detail:

- **level 0** requires that messages are delivered *at most* once;
- **level 1** requires that messages are delivered *at least* once;
- **level 2** requires that messages are delivered *exactly* once.

Suppose a publisher sets QoS level QoS_p when it publishes to a topic t. Similarly, a subscriber chooses QoS level QoS_s when it subscribes to the same topic t. Two cases may occur: (i) if $QoS_p > QoS_s$, then the broker forwards the data to the subscribing client using QoS_s; (ii) if $QoS_p \leq QoS_s$, then the broker forwards the data to the subscribing client using QoS_p.

Another feature provided by the MQTT protocol is represented by the *retained messages*. A retained message is an MQTT message, with the retained flag set to true. When a broker receives a retained message labeled with topic t, it stores it for that topic. Thus, when a client subscribes to the topic t it will receive the retained message immediately after the subscription. A broker can store only one retained message per topic. Therefore, to update the retained message for a topic, it is sufficient for a publisher to send a new message, to that topic, with the retained flag set to true.

A core MQTT feature is represented by the bridging mechanism that allows two MQTT brokers to connect with each other.

This configuration is adopted to connect an edge broker (which will act as a bridge) to a public broker. This way, the bridge broker will act as an MQTT client for the public broker, thus being able to send a subscribe or publish message to the latter.

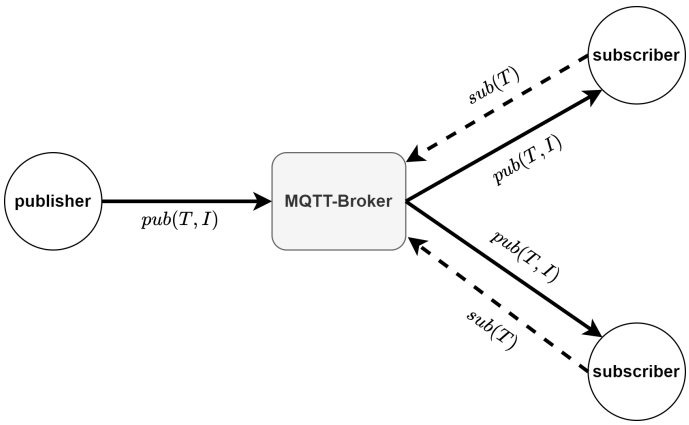


Fig. 1: MQTT architecture.

Usually, clients are connected to the bridge broker through the local network, and the bridge broker can decide on which local topics to apply the bridging mechanism. This is due to the fact that not all the MQTT traffic locally generated is meant to be sent to a remote broker.

The bridging mechanism is sketched in Figure 2. Therein, the subscribers send the subscribe message $sub(T)$ directly to the public broker (dashed arrows). Conversely, the publisher sends the publish message $pub(T, I)$ to the bridge broker (solid arrow). Then, by exploiting the bridging mechanism, the bridge broker can forward $pub(T, I)$ to the public broker. In turn, the latter forwards $pub(T, I)$ to all the interested subscribers.

Finally, MQTT allows the remapping of the local topics to public broker's topics. This procedure takes place in the bridge broker, in such a way that it is transparent to MQTT clients. It is worth noting that, through the bridging mechanism, information produced behind different brokers can be aggregated in a single place at a public broker, thus allowing subscribers to retrieve it with a single connection.

III. Scenario and motivations

The aim of this work is twofold. First, we want to offer anonymity guarantees to both publishers and subscribers. Indeed, in a classical MQTT architecture, the broker can observe which subscribers are interested in which topics and which publishers send messages labeled with those topics.

Clearly, encryption-based data-confidentiality does not solve the anonymity problem. Indeed, the broker is still able to observe which publishers and subscribers are communicating. Therefore, attacks based on background knowledge about even one client can allow the broker to infer the topic and, as a consequence, information about all the clients communicating on this topic.

Based on the above considerations, in this paper, we focus on achieving privacy goals by hiding identities rather than contents. If client identities are really hidden, then the broker cannot link contents with clients. Privacy is then achieved, even in the case of non-encrypted contents. Consider that the trivial use of a pseudonym for the client is not enough because de-anonymization is always possible, also by taking into

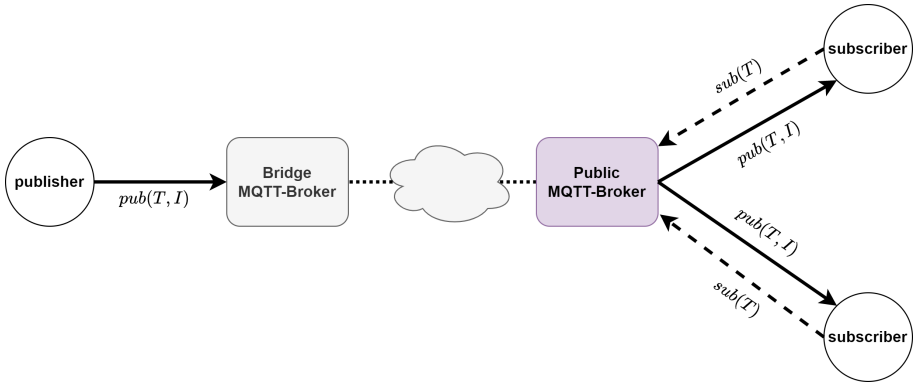


Fig. 2: MQTT bridging mechanism.

account the source and the destination of the communication (i.e., the IP addresses) are quasi identifiers [44]. Therefore, we have to guarantee that both publishers and subscribers are entities that the broker is not able to identify in the network. This is a goal typically reached in the context of anonymous communication networks [36].

To accomplish the above objective, we refer to the MQTT bridging mode (see Section II) in which clients (publishers/subscribers) interact with a public broker through a bridge broker. This natively implements a low level of anonymity, in the sense that clients do not connect directly to a broker but are hidden behind their bridge brokers acting as proxies. However, it is not enough to achieve true anonymity. First, also the identification of the local bridge could threaten privacy in the case in which the local bridge serves a restricted group of clients. Moreover, it may happen that the bridge broker aggregates similar clients (in terms of interests). Therefore, the above-mentioned issues still occur. This problem is related to the concept of *l-diversity* [45]. In general, inference-based attacks are still possible.

The idea we follow in this paper is to hide the identity of the clients by making anonymous and not identifiable the local bridge from/to which the communication comes/goes. To do this, we implement, in Section V, an anonymity peer-to-peer protocol inspired by [39], in which the peers are represented by the bridge brokers.

To make effective the approach, we also provide a protocol allowing the clients to discover which topics are offered by which public brokers. This is an important aspect to take into consideration even when anonymity features are not required. Indeed, MQTT does not implement any native mechanism to accomplish this task [41]. Furthermore, this becomes a fundamental pillar when publishers remain anonymous. Indeed, being anonymous, a publisher cannot otherwise advertise the topics to which it will publish.

To conclude this section, we provide the notations we use throughout the rest of the paper. We denote by $\mathcal{B}_R$ the set of the *bridge* brokers. They form the peer-to-peer network of the anonymity protocol. We denote by $\mathcal{B}_P$ the set of the *public* brokers. They represent the actual brokers hosting the topics to which the clients publish or subscribe. We denote by $\mathcal{B}_D$ the set of *discovery* brokers. They implement the discovery protocol allowing the clients to know which public broker offers which topic. Moreover, they enable peer discovery for the anonymity protocol.

TABLE I: Notations.

Symbol	Description
$\mathcal{B}_R$	set of the bridge brokers
$\mathcal{B}_P$	set of the public brokers
$\mathcal{B}_D$	set of discovery brokers
p	a generic publisher
s	a generic subscriber
$pub(T, I)$	publish message of the information I labeled with the topic T
$sub(T)$	subscribe message labeled with the topic T
B_R^p	bridge broker directly connected to the publisher p
B_R^s	bridge broker directly connected to the subscriber s
T_N	topic to which new topics available at a certain public broker are published
$T_{N'}$	topic to which new bridge brokers joining the network are published
T_S	topic to which the current set of all the available topics is published
$T_{S'}$	topic to which the current set of all bridge brokers is published
T_F	(*forwarding* topic) topic-prefix labeling information coming from remote bridge brokers
T_A	(*actual* topic) actual topic labeling an information
p_f	Forwarding probability
R	Generic random value to compare with p_f
L	Average length of the paths
S	Set of pairs composed of a topic and public broker hosting it
S'	Set of pairs composed of IP address and port of a bridge broker
n	number of brokers in the P2P network
c	number of collaborating malicious brokers in the P2P network

Finally, we denote by $pub(T, I)$ the publish message of the information I labeled with the topic T, and by $sub(T)$ the subscribe message labeled with the topic T.

We report in Table I the notations used throughout the rest of the paper.

IV. The discovery protocol

Through this protocol, we offer a discovery service to the bridge brokers so that they can know which topics are available

and which public brokers offer them. This service is provided by the discovery brokers. In principle, just a single discovery broker is enough to implement this protocol. Anyway, for scalability, it is more realistic to distribute the service to more brokers so that bridge and public brokers are not connected to a single possible point of failure [46].

Each discovery broker in $\mathcal{B}_D$ maintains:

- a set of pairs $S = \{(T, B_P)\}$ where $B_P \in \mathcal{B}_P$. Each pair $(T, B_P) \in S$ represents the information that the topic T is available on the public broker B_P. The set S has to be the same for all the discovery brokers in $\mathcal{B}_D$.
- a topic T_N that will contain a single pair (T, B_P) denoting that a *new* topic T is available on the public broker B_P.
- a topic T_S that will provide the current set S to the new bridge brokers joining the network. S is stored as a retained message for the topic T_S.

Since S could exceed the maximum size allowed for the content of a topic, we may consider more topics $T_S^1, \ldots, T_S^k$, each one containing a portion of S. For the sake of presentation, we consider just a single topic T_S, which is also the more realistic case.

The topics T_N and T_S are prefixed strings in the system.

The broker B_D publishes (to itself) the message S with the retained flag set to true. This way, all the clients subscribing to the topic T_S receive the retained message immediately after they subscribe. To update the set S, it is sufficient for the broker to just re-publish again S (with retained flag set to true). Doing so, as explained in Section II, only the last retained message will be stored.

Consider now a new bridge broker $B_R \in \mathcal{B}_R$ joining the system. It randomly selects a discovery broker $B_D \in \mathcal{B}_D$ and sends it the message $sub(T_S)$. In other words, B_R subscribes to the topic T_S hosted by the broker B_D. This way, B_R receives (as a retained message) the current set S. Then, it removes its subscription to T_S, so that it does not receive repeated information (i.e., the set S) when other bridge brokers join the network.

However, a mechanism to update S (when a new topic is available on some public broker) is needed to B_R and B_D. This mechanism is based on T_N.

In particular, when the bridge broker B_R joins the network and (randomly) selects the discovery broker B_D, B_R also sends $sub(T_N)$ to it. This subscription is maintained over time.

Consider now a public broker $B_P \in \mathcal{B}_P$ receiving a publish message or a subscribe message labeled with a new topic T^* through the anonymity protocol of Section V. B_P has to advertise the fact that it hosts this new topic.

To do this, B_P randomly selects a discovery broker B_D and sends the message $pub(T_N, (T^*, B_P))$. B_D adds the pair (T^*, B_P) to the set S. Moreover, since some bridge brokers are subscribers to T_N, they also receive (through the standard MQTT approach) the new pair and update the set S. To propagate the new information to all the bridge brokers, B_D sends the message $pub(T_N, (T^*, B_P))$ to all the other discovery brokers. Since each bridge broker is a subscriber to T_N on some discovery broker, all the bridge brokers eventually receive (T^*, B_P) through the standard MQTT protocol.

V. The anonymity protocol MQTT-A

The anonymity protocol we propose in this section is inspired by the Crowds protocol [39]. Our solution relies on a P2P network formed by bridge brokers collaborating to forward publish/subscribe messages from MQTT clients to the intended public broker. The collaboration among brokers exploits the bridging mechanism, which is natively supported by the MQTT protocol, thus requiring no infrastructural change in the MQTT architecture.

Similarly to Crowds, our protocol is characterized by a forwarding probability p_f, namely the probability for a message to remain within the peer network. Moreover, as in Crowds, we require that each bridge broker knows all the peers participating in the P2P network. To guarantee this, we propose an MQTT-based peer discovery protocol, thanks to which every peer maintains an updated set of all the bridge brokers forming the network. This protocol is based on the approach presented in Section IV. It leverages one or more discovery brokers holding a set of pairs $S' = (IP_{B_R}, port_{B_R})$, where IP_{B_R} and $port_{B_R}$ represent the IP address and the port of the bridge broker B_R participating in the network.

Moreover, each discovery broker stores two additional prefixed topics: $T_{N'}$ and $T_{S'}$. The topic $T_{N'}$ is used by the discovery broker itself to publish any update of the set S' (an update typically happens whenever a new peer joins the network). Instead, the topic $T_{S'}$ holds the set S' as a retained message, so that any new peer joining the network, after subscribing to this topic, immediately receives the set S'.

When a bridge broker joins the P2P network, it proceeds as follows. First, it subscribes to the two topics $T_{S'}$ (to receive the set S') and $T_{N'}$, of a randomly selected discovery broker B_D.

Then, the bridge broker sends a publish message labeled with the topic $T_{N'}$ advertising its IP address and its port to B_D. This way, every bridge broker subscribed to the topic $T_{N'}$ of B_D receives such a message and, at the same time, B_D can update the set S'. At this point, B_D sends a publish message labeled with the topic $T_{N'}$ to all the other discovery brokers, advertising the new pair composed of the IP address and the port of the new bridge broker. Since each bridge broker is a subscriber to $T_{N'}$ of some discovery broker, this procedure allows every bridge broker to learn all other brokers of the network.

We stress that, as described in Section IV, every bridge broker, joining the P2P network, must subscribe to both $T_{N'}$ and $T_{S'}$. However, while the subscription to $T_{N'}$ lasts over time, the subscription to $T_{S'}$ is undone once the set S' is received.

This concludes the description of the peer discovery protocol.

In the following, we describe in detail the anonymity protocol we propose.

We consider a scenario involving:

- a public broker $B_P \in \mathcal{B}_P$ hosting the topic T_A (where A stands for *actual*);
- a publisher p aiming to publish information to the topic T_A;

- a subscriber s interested in the topic T_A;
- the set of bridge brokers $\mathcal{B}_R$, among which the broker $B_R^p \in \mathcal{B}_R$ represents the bridge broker directly connected to the publisher p, and the broker $B_R^s \in \mathcal{B}_R$ represents the bridge broker directly connected to the subscriber s.

We assume that B_R^p (B_R^s, respectively) is able to distinguish p (s, respectively) as a client different from other bridge brokers. This can be done simply by observing that p (s, respectively) does not belong to the list of peer brokers.

Among other topics, each bridge broker stores a prefixed topic T_F known to all the bridge brokers and clients, where F stands for *forwarding*.

Consider now the publisher p that wants to publish the information I to the topic T_A. p sends the message $pub(T_F/T_A, I)$ to the broker B_R^p. B_R^p randomly selects a bridge broker, say B_R^1, among the peers participating in the network. Then, it simply forwards the message $pub(T_F/T_A, I)$ to B_R^1, via the bridging mechanism. We recall that the bridging mechanism allows B_R^p to act as an MQTT client (a publisher, in this case) towards B_R^1.

Once the message is received, B_R^1 acts as follows. First, it extracts a random number R between 0 and 1. At this point, two cases can occur.

If $R \leq p_f$, then B_R^1 randomly selects a bridge broker B_R^2 among the peers participating in the network. Then, B_R^1 sends the message $pub(T_F/T_A, I)$ to B_R^2, via the bridging mechanism.

If $R > p_f$, then the message must be published to the public broker holding the topic T_A, if any. To do this, B_R^1 finds the pair (T_A, B_P) in the locally stored set S. If such a pair exists, after remapping T_F/T_A to T_A, B_R^1 simply sends the message $pub(T_A, I)$ to B_P. Conversely, if such a pair does not exist yet, B_R^1 chooses at random a public broker $B_P \in \mathcal{B}_P$ and then sends it the message $pub(T_A, I)$. In this case, being T_A a new topic for B_P, B_P must activate the discovery protocol (see Section IV) to notify all the bridge brokers of the new topic. This way, each bridge and discovery broker will update the set S.

Consider now the case in which $R \leq p_f$. In this case, B_R^2 receives from B_R^1 the message $pub(T_F/T_A, I)$. At this point, the above procedure is applied recursively. Therefore, B_R^2 again extracts a random number R' and compares it with p_f. If $R' \leq p_f$, the message $pub(T_F/T_A, I)$ is sent to a bridge broker B_R^3 chosen at random. In this case, no topic remapping is needed. On the contrary, if $R' > p_f$, then B_R^2 sends the message $pub(T_A, I)$ to the public broker B_P, following the procedure described above.

The mechanism so far described provides anonymity to publishers. A similar mechanism can be employed so that the same anonymity guarantees are provided to subscribers.

Consider the subscriber s interested in the topic T_A. s sends the message $sub(T_F/T_A)$ to the broker B_R^s. Then B_R^s chooses at random a bridge broker, say B_R^4, and sends it the message $sub(T_F/T_A)$, via the bridging mechanism.

Once receiving the above message, B_R^4 acts as follows. First, B_R^4 stores B_R^s in the list of clients subscribed to the topic T_F/T_A, creating this topic if it has not already been stored locally. Afterward, it extracts a random number and compares it to p_f to decide whether to send the subscription message to the public broker B_P (retrieved from its local set S) or to another bridge broker.

In the first case, B_R^4 sends $sub(T_A)$ to B_P and then the procedure stops. Observe that, in this case, the original topic T_F/T_A needs to be remapped to T_A. In the second case, B_R^4 sends $sub(T_F/T_A)$ to a randomly chosen bridge broker, say B_R^5. At this point, B_R^5 recursively repeats the procedure described above. Therefore, B_R^5 stores B_R^4 in the list of clients subscribed to the topic T_F/T_A, possibly creating this topic. Then, B_R^5 extracts a random to decide where to send the received subscribe message. Eventually, after a certain number of iterations, the subscribe message will reach the intended public broker B_P.

Finally, when B_P receives a publish message, say $pub(T_A, I)$, it broadcasts the payload I to all the subscribers to the topic T_A. Suppose B_R^5 is the bridge broker directly subscribed to the topic T_A at B_P. B_R^5 will receive I and, then, it will broadcast this payload to all the clients subscribed to the topic T_F/T_A, among which there is B_R^4. Observe that, being B_R^5 the bridge broker directly connected to B_P, the topic T_A must be remapped to T_F/T_A. Again, following the same mechanism, B_R^4 will send the payload I to all the subscribers interested in the topic T_F/T_A, among which there is B_R^s. Once B_R^s receives the payload I, it will broadcast I to all the subscribers interested in T_F/T_A, among which there is s.

Observe that, in our solution, the path followed by a message is not deterministically chosen by the sender, but it is probabilistically built by peer bridge brokers at each hop. Probabilistic paths may introduce two issues. The first concerns the possibility of lacking message ordering since messages may follow different paths (with different lengths). The second is about the possibility of loops being created in the P2P network. We address both problems in Section VI.

To conclude this section, in Figure 3, we provide a graphical representation of the anonymity protocol. Therein, we represent (i) (solid line on the left) the path followed by a publish message, from the publisher toward the public broker, (ii) (solid line on the right) the path followed by a subscribe message, from a subscriber toward the public broker, and (iii) (dashed line on the right) the path followed by the publish message from the public broker toward the subscriber. As depicted in the figure, the paths in cases (i) and (ii) depend on the value of R compared to p_f. Specifically, when $R \leq p_f$ publish (or subscribe) messages are forwarded to peer brokers. Otherwise, when $R > p_f$ publish (or subscribe) messages are forwarded to the final public broker.

VI. Complexity and Overhead Analysis of MQTT-A

Through this section, we provide a study of the complexity and the overhead related to the proposed approach by highlighting the price to be paid compared to the standard MQTT protocol.

As a first observation, to provide anonymity we require messages to traverse paths whose lengths depend on the probability p_f. Clearly, the higher p_f, the higher the length of each path and then the latency to transfer each message.

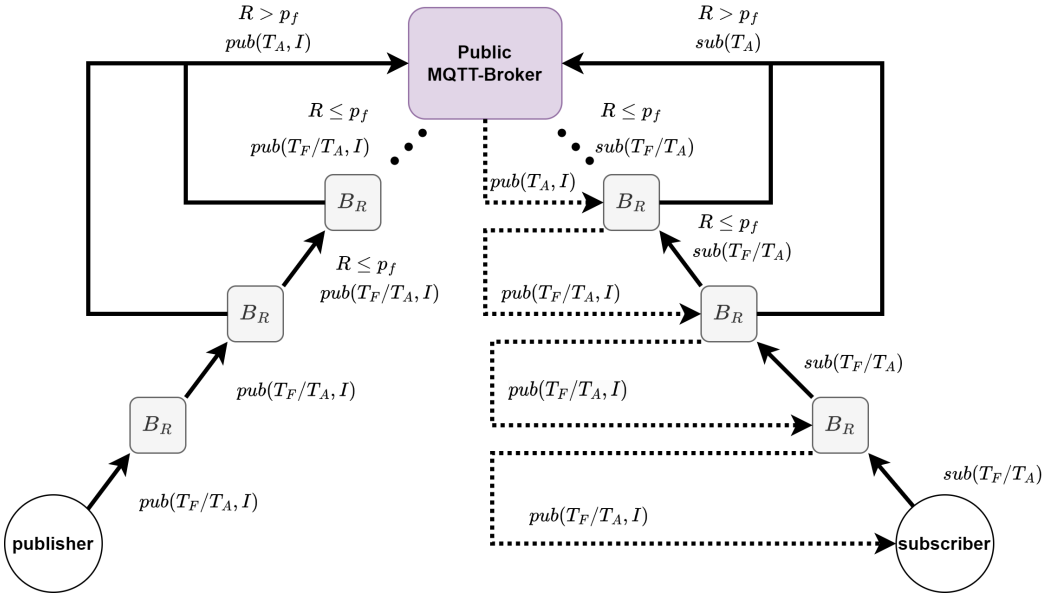


Fig. 3: MQTT-A.

The average length L of the paths, as a function of p_f, can be derived as follows. First, a single hop is present between the bridge broker directly connected to an MQTT client and another bridge broker. Indeed, the first bridge broker does not extract the random R, but always forwards the messages to another bridge broker. Then, we have to compute the average length of the path between the second bridge broker and the public broker. To compute it, we have to compute the probability for each possible length of this path. Then, with probability $1 - p_f$, the second broker directly sends the message to the public broker. This results in one hop between the second broker and the public broker. Instead, with probability $(1-p_f)p_f$ the second broker forwards the message to another bridge broker and the latter forwards the message to the public broker. This results in two hops between the second broker and the public broker. Similarly, by extending the above reasoning, we have a path of length i between the second bridge broker and the public broker with probability $(1 - p_f)(p_f)^{i-1}$.

Now, we can compute the average length by summing each possible length weighted with its occurrence probability. Formally, it results in $\sum_{i=1}^{+\infty} i(1 - p_f)(p_f)^{i-1}$.

By adding the first term including the hop between the first bridge broker and the second bridge broker, we have that the average length is given by the following equation:

$$L = \sum_{i=1}^{+\infty} i(1 - p_f)(p_f)^{i-1} + 1 = \frac{1}{1 - p_f} + 1 \qquad (1)$$

We report in Table II the average length of the path for each probability p_f.

The actual price in terms of latency introduced by MQTT-A is experimentally evaluated in Section VIII, by highlighting also how to set the value of p_f not to exceed a maximum latency value. In addition, the trade-off between security and efficiency (in terms of latency) related to the choice of p_f is investigated in Section VIII.

TABLE II: Average path length.

p_f	**Average path length**
0.5	3
0.66	4
0.75	5
0.8	6
0.835	7
0.86	8
0.875	9
0.89	10
0.9	11

Another price we pay, compared to standard MQTT, is the introduction of a topic prefix to let the broker know whether a message should be sent in the P2P anonymous network or managed locally by the broker (as in standard MQTT). This can be done with an additional bit associated with the topic to distinguish these two kinds of situations with no significant overhead.

Concerning the computational complexity required of brokers, we recall that, with respect to the standard MQTT protocol, they have just to extract a random value R and select the broker to send the message. Clearly, the cost of the first operation is negligible. Concerning the second operation (selection of the broker), we distinguish two cases. If $R \leq p_f$, then a broker simply selects randomly another broker from the set of peer brokers. Again, the cost of this operation is negligible. Otherwise (i.e., $R > p_f$), the broker retrieves, from the set S, the public broker associated with the topic of the received message. This operation can be efficiently performed by implementing S as a hash table using the names of the topics as keys. The time required to perform this operation is constant and negligible compared to the processing time of

the MQTT message.

To conclude this section, we provide some observations concerning the message ordering issue and the loop creation in the P2P network.

As highlighted by [12], although the MQTT protocol was designed to guarantee a certain QoS level for each message, it may be hard to ensure that messages are delivered in the exact order in which they were transmitted by the senders. Indeed, even though QoS 2 guarantees single message delivery, it does not guarantee message ordering. This holds also for MQTT-A. Moreover, in our protocol, the problem may occur more frequently. Indeed, it may be that two successive messages sent by a publisher follow different paths so that the first message traverses a longer path than the second message, thus arriving at its destination with no predictable order. The ordering problem in MQTT-A can be solved by adopting the same strategy as [12] consisting of the inclusion of a sequence number in the MQTT packets. Clearly, this leads to computational overhead at the application layer.

The second issue regards the possibility of loop creation in the P2P network. Specifically, it is possible that a publish message crosses twice the same bridge broker before reaching the public broker. Actually, this does not represent an issue. Conversely, concerning subscribe messages, the loop creation may represent an issue since the received (published) messages may continue to circulate in the network. However, the solution to the above problem is trivial since each subscriber message contains a packet id. Then, it is sufficient that the same packet id used by the client is maintained by the bridge brokers for a while. This way, if two subscribe messages with the same id are received by the same bridge broker, the second message is discarded and no loop arises.

VII. Path intersection and QoS management

In Section V, to simplify the description of the protocol, we considered the case of a single publisher and a single subscriber. Moreover, we assumed that the paths followed by the publish and subscribe messages labeled with the same topic do not intersect.

Now, we remove the above-simplifying assumptions and deal with the consequent impact on QoS guarantees of subscribers.

Three cases are investigated: (i) the paths followed by at least two subscribe messages labeled with the same topic intersect, (ii) the paths followed by at least one publish message and at least one subscribe message labeled with the same topic intersect, and (iii) the paths followed by at least two publish messages labeled with the same topic intersect.

As we will see in the following, the problem of path intersection requires ad-hoc strategies to guarantee the QoS level chosen by each subscriber.

We recall that QoS levels 0 and 1 allow duplicate messages, while QoS level 2 requires the message to be delivered exactly once. We stress that both publishers and subscribers can independently choose the desired QoS level for each message being sent or received. Therefore, as an example, a client p may publish to a topic with QoS level 1, while there is a subscriber s interested in the same topic but aiming to receive data with QoS level 2. Therefore, s cannot be sure about the fact that it is not receiving duplicate messages.

This QoS mismatching represents an open problem even in standard MQTT. Without loss of generality, in the following, we will consider that clients always publish messages with QoS level 2. This way, any QoS level chosen by subscribers can be met. Moreover, to improve the readability of our discussion, in the following, unless otherwise stated, we will implicitly refer to publish and subscribe messages labeled with the same topic.

A. Subscribe-Subscribe Path Intersection

In this section, we consider a scenario in which a set of subscribers is interested in the same actual topic T_A, hosted by the public broker B_P. Each subscriber may select arbitrarily a certain QoS level for its subscribe message.

Following our protocol, each subscribe message crosses a certain number of bridge brokers before reaching the public broker B_P. Suppose that, before reaching B_P, two or more subscribe messages cross the same bridge broker. In the following, we will call such a broker as *intersection broker*.

At this point, the intersection broker can choose one of the following strategies:

- *blind* strategy;
- *topic-aware* strategy;
- *topic-and-QoS-aware* strategy.

These three strategies are sketched in Figures 4,5, and 6, respectively. Specifically, these figures depict the paths followed by two or more messages (labeled with the same topic) before and after the intersection in a broker (depicted in yellow). To facilitate the understanding of the images, the path followed by each message is represented with a specific color.

An intersection broker, implementing the *blind* strategy, treats all the subscribe messages the same, regardless of whether they are labeled with the same or different topics. Therefore, subscribe messages labeled with the same topic, in principle, can be routed to different brokers. For instance, as depicted in Figure 4, the messages sent by the first subscriber and the second subscriber (red path and the blue path, respectively), once they intersect, are routed independently.

An intersection broker, implementing the *topic-aware* strategy, treats the subscribe messages labeled with the same topic differently from all other messages crossing it. In particular, referring back to our scenario, just the first received subscribe message labeled with the topic T_F/T_A is forwarded according to the protocol described in Section V. On the contrary, all other subscribe messages, labeled with the same topic, that arrive later are not further forwarded.

For example, in Figure 5, suppose the subscribe message sent by the second subscriber (blue path) reaches the broker after the subscribe message sent by the first subscriber (red path). Being the *topic-aware* strategy adopted, the message following the red path is routed to the public broker. Conversely, the message following the blue path, once it reaches the intersection broker, is not further forwarded.

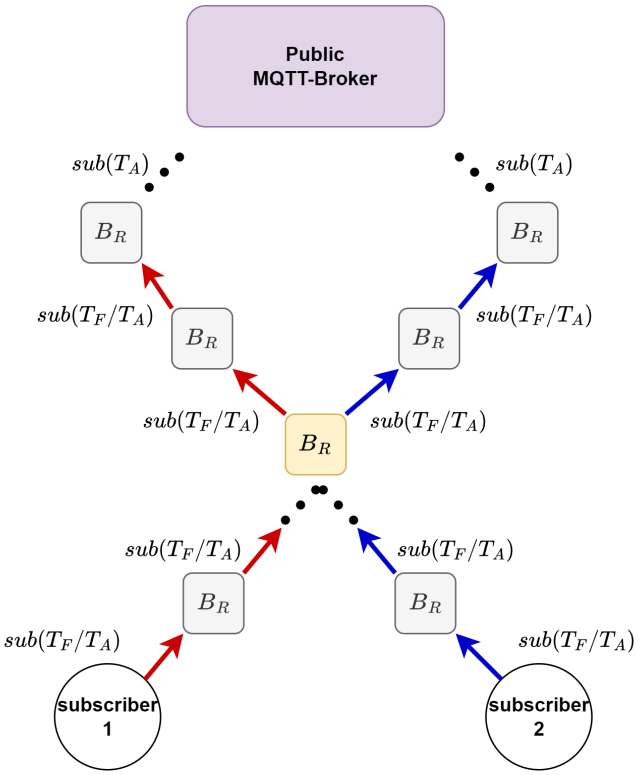


Fig. 4: Blind strategy.

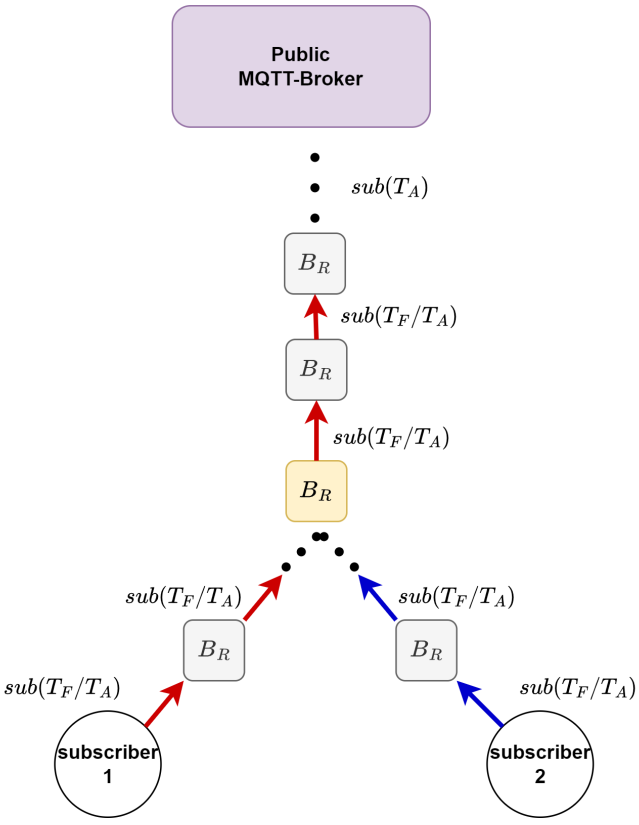


Fig. 5: Topic-aware strategy.

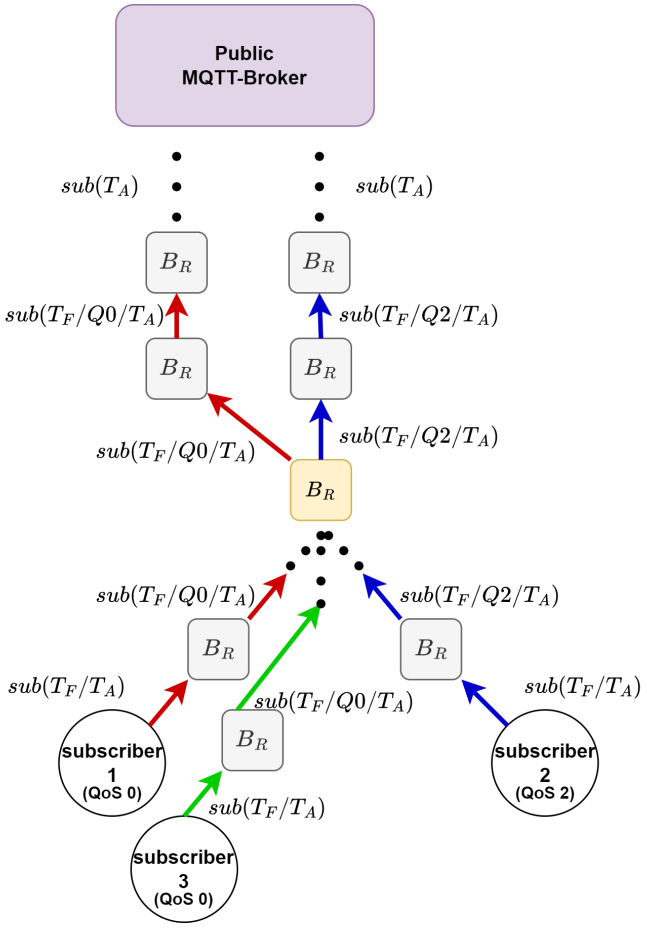


Fig. 6: Topic-and-QoS-aware strategy.

Observe that, for all other subscribe messages (with the same topic) that arrive later, the intersection broker has just to memorize the bridge brokers they come from in the list of brokers interested in topic T_F/T_A. This is enough to guarantee that all the subscribers will be able to receive the data later published to the actual topic T_A. We recall that this strategy is not QoS-aware and therefore messages labeled with the same topic but requiring different QoS are not treated differently.

An intersection broker, implementing the *topic-and-QoS-aware* strategy, distinguishes messages, labeled with the same topic, on the basis of the QoS level they require. To handle the different QoS levels, each bridge broker hosts three forwarding topics $T_F/Q0$, $T_F/Q1$, and $T_F/Q2$. Therefore, subscribe messages sent by clients interested in the actual topic T_A but with different QoS levels will follow different paths in the network. In fact, subscribe messages requiring different QoS levels result in messages labeled with different topics (either $T_F/Q0$, $T_F/Q1$, or $T_F/Q2$).

For example, in Figure 5, suppose the first and the third subscriber require QoS 0, while the second subscriber requires QoS 2. Moreover, suppose that the subscribe message sent by the first subscriber (red path) arrives at the intersection broker before the subscribe message sent by the second subscriber (blue path) that in turn arrives at the intersection broker before the subscribe message sent by the third subscriber (green

path). In this case, the subscribe messages of the first and the second subscribers are routed to the public broker through different paths since they require different QoS levels. On the other hand, the message following the green path (sent by the third subscriber), once it reaches the intersection broker, is not further forwarded. This is due to the fact that such a message presents the same QoS level as one of the previously forwarded messages (i.e, we adopt the *topic-aware* strategy).

Observe that, regarding the messages labeled with the same actual topic and QoS level, the following strategies may be applied:

- concerning topics $T_F/Q0$ and $T_F/Q1$, both *blind* strategy and *topic-aware* strategy can be applied (in Figure 6, we adopt the *topic-aware* strategy);
- concerning topic $T_F/Q2$, only the *topic-aware* strategy can be applied.

B. Publish-Subscribe Path Intersection

In this section, we consider a scenario in which a client aims to publish to the actual topic T_A (hosted by the public broker B_P) and at least one subscriber is interested in the same actual topic T_A.

Following our protocol, each (publish or subscribe) message crosses a set of bridge brokers before reaching the public broker B_P. Suppose that, before reaching B_P, at least one subscribe message and one publish message cross the same bridge broker. As before, we call such a broker intersection broker.

At this point, the intersection broker can choose one of the three strategies mentioned above (i.e., either the *blind* strategy, the *topic-aware* strategy, or the *topic-and-QoS-aware* strategy).

An intersection broker, implementing the *blind* strategy, acts as follows. When it receives the publish message it does not publish this message locally, regardless of whether it has locally memorized other bridge brokers subscribed to the same topic labeling the publish message. Therefore, the publish message is further forwarded until it reaches the public broker B_P.

An intersection broker that implements the *topic-aware* strategy acts as follows. When it receives the publish message, it first verifies whether it has locally memorized other bridge brokers subscribed to the same topic of the publish message. If this is not the case, then the intersection broker behaves as described by the *blind* strategy. Conversely, if the above condition is met, the intersection broker publishes the received message locally. This way, all the bridge brokers, subscribed to the same topic, receive the payload associated with the publish message and, in turn, this payload is forwarded up to the actual subscribers. Observe that, even though the *topic-aware* strategy is applied, the publish message has to be further forwarded to the public broker B_P, since this is the only way that message can reach all the interested subscribers.

As already described, implementing the *topic-and-QoS-aware* strategy requires a slight modification to the forwarding topics hosted by each bridge broker. Indeed, according to this strategy, each bridge broker hosts three forwarding topics: $T_F/Q0$, $T_F/Q1$, and $T_F/Q2$. When a publish message labeled with the topic T_F/T_A reaches an intersection broker, such a broker follows the *topic-aware* strategy. However, differently from before, the message is locally published only to the topics $T_F/Q0$ and $T_F/Q1$.

C. Publish-Publish Path Intersection

In this section, we consider a scenario in which the paths followed by two publish messages intersect. Unlike subscribe messages, publish messages do not require the memorization of a permanent state at each broker (i.e., the subscription of a client). Therefore, bridge brokers are unaware, unless they maintain the history of messages, of the fact that they may be crossed by publish messages (labeled with the same topic) at different time periods.

Actually, this situation requires no additional action from the side of the intersection broker. The two publish messages are treated independently and forwarded according to the standard rules of the MQTT-A protocol.

D. Strategies Comparison

To conclude this section, we examine the advantages and drawbacks of the presented strategies. Overall, the *blind* strategy represents the easiest solution to implement. However, adopting this strategy leads to higher bandwidth consumption than other strategies since messages intersecting in the same broker are simply propagated into the network without being aggregated [47]. On the contrary, the *topic-aware* strategy requires the least bandwidth of the three strategies. However, the main drawback of this strategy is that it is not QoS-aware. To explain why this may represent an issue, we consider the following example. Suppose there are two subscribers s_1 and s_2 both interested in the topic T_A but requiring different QoS, 0 and 2 respectively. We consider that the paths followed by the two subscribe messages in the network intersect. According to the *topic-aware* strategy, the intersection broker further forwards only the first received subscribe message. This way, all the brokers (including the intended public broker B_P) crossed by such a message will memorize the QoS level that it requires. Such QoS level can be either 0 or 2 depending on which subscribe message comes first at the intersection broker. Observe that, since the public broker will be aware of just one subscription out of the two, applying the *topic-aware* strategy also turns out to be advantageous anonymity-wise speaking, not just bandwidth-wise speaking.

Therefore, all the data published to the topic T_A will cross all the bridge brokers from the public broker to the intersection broker respecting the QoS level required by the subscribe message. Observe that, this may be an issue, either for s_1 or s_2. Indeed, supposing the s_1 subscription (requiring QoS level 0) comes first at the intersection broker, the QoS level required by s_2 cannot be satisfied, since it is higher than 0. On the contrary, supposing the s_2 subscription (requiring QoS level 2) comes first at the intersection broker, the QoS level required by s_1 can be satisfied, since it is lower than 2. However, this comes with a cost. Indeed QoS level 2 is satisfied in the path from the public broker to the intersection broker, while QoS

level 0 is satisfied in the path from the intersection broker to s_1. In general, the overhead introduced by the QoS level 2, in the first part (i.e., from the public broker to the intersection broker) of the path, may not be acceptable for s_1.

A similar case may happen when the paths followed by a publish and a subscribe message intersect. Suppose there are a publisher p and a subscriber s, both requiring QoS level 2. According to the *topic-aware* strategy, the subscriber s receives the publish message by the intersection broker. Actually, s will receive the same publish message also by the public broker. Therefore, despite the fact that s chose the QoS level 2, s will receive duplicate messages. This is due to the fact that, regardless of whether or not a path intersection has occurred, the publish message, as well as the subscribe message, must reach the public broker. This way, all the interested subscribers can receive the publish message and the subscriber s can receive all the messages published to the topic by all the publishers in the network, not just by p.

As the occurrence of the two situations above described may not be acceptable, bridge brokers can implement the *topic-and-QoS-aware* strategy. Indeed, referring back to our last example, when the publish message reaches the intersection broker, it is locally published just to the topics $T_F/Q0$ and $T_F/Q1$, which are also the QoS levels that can tolerate duplicate messages. Such a strategy ensures that the QoS level required by subscribers is always met, provided that publishers guarantee an appropriate QoS level. However, this comes with a cost. Indeed this strategy requires more bandwidth consumption than the *topic-aware* strategy.

VIII. Experiments

Through this section, we perform an experimental validation of MQTT-A and compare it with the standard MQTT protocol. To obtain the anonymity features, MQTT-A introduces a communication overhead. Therefore, this section aims to show that the price we pay is tolerable and the performance results acceptable. We also investigate the supported applications and some trade-offs of MQTT-A.

A. Experimental Setting

For implementation and testing, we relied on HiveMQ CE [48], which is a Java-based open-source MQTT broker. We customized the bridge brokers to implement our solution and deployed a P2P network leveraging also Digital Ocean cloud platform [49] to have remote peers. No change is required for clients and public brokers. In detail, for our experiments, we consider two network configurations, one for MQTT-A and the other for MQTT. The aim is to compare the two protocols by measuring latency and goodput.

MQTT-A Configuration In MQTT-A, we have the following components:

- **Clients**: They are implemented through the Java library provided in [50] with no change and deployed on standard laptops equipped with 1.00 GHz Intel i5-1035G1 CPU and 8 GB of RAM. They are connected to the bridge brokers through a local network. We observe that the performance of our protocol does not depend on the capabilities of MQTT clients. Indeed, we do not require any change either publisher-side or subscriber-side. Instead, the performance depends on the network conditions (latency and bandwidth of the network), length of paths (due to the forward probability p_f), and the possible overhead introduced broker-side.
- **Bridge Brokers**: They are implemented by customizing the Java library [48] and deployed to form a P2P network. In particular, they are partially deployed on standard laptops in our laboratory and partially deployed on the Digital Ocean cloud platform [49]. To obtain realistic results, we deployed the bridge brokers in different cities of the world. Specifically, we selected 6 cities: Reggio Calabria (Italy), Frankfurt (Germany), Amsterdam (Netherlands), New York (United States), Bangalore (India), and San Francisco (United States). Then, we considered different network configurations to obtain different (average) round-trip times (RTT) between bridge brokers.
  The brokers located in our laboratory in Reggio Calabria are standard laptops, some equipped with 1.80 GHz Intel i7-8550U CPU and 16 GB of RAM and others equipped with 2.70 GHz Intel i7-7500U CPU and 8 GB of RAM. The other bridge brokers are deployed on virtual machines in the cloud and they all are equipped with a vCPU (virtual CPU) and 1 GB of RAM.
- **Public Brokers**: They are deployed on HiveMQ Cloud, a cloud-based platform hosting MQTT brokers. They are standard brokers and do not require any implementation change. Clearly, the communication bridge-to-public brokers always happens through the Internet.

MQTT Configuration In MQTT, the scenario is a simplification of the previous scenario. In particular, clients are connected to bridge brokers through the local network and bridge brokers are *directly* connected to public brokers through the Internet.

Regarding MQTT-A, we chose to adopt the *blind* strategy (see Section VII), which is the simplest one but the worst in terms of performance. As a measure of performance, we considered goodput and latency. In the next two sections, we compare MQTT-A and MQTT by fixing the average RTT between bridge brokers to 55 ms and varying other parameters (packet size, sending rate, forward probability, and QoS). Then, in Section VIII-D, we study how the performance of MQTT-A varies with the RTT by highlighting the supported applications and possible trade-offs.

B. Goodput

For goodput, we mean the number of *useful* information bits received by a subscriber in the unity of time.

To measure it, we fixed a sending rate for the publishers and observed the corresponding goodput for the subscribers. We considered three different sending rates: 1 KBytes/s, 10 KBytes/s, and 100 KBytes/s. We study how the goodput varies as the forward probability of MQTT-A varies. Since the higher the forward probability the longer the paths, we expect the goodput of MQTT-A decreases as the forward probability increases. Obviously, the goodput of MQTT does not depend on the forward probability.

We performed our experiments by considering two levels of QoS (0 and 2). Since QoS level 2 requires additional communication overhead compared to QoS level 0, the goodput of both MQTT and MQTT-A is lower than QoS level 0.

The results with QoS level 0 are reported in Figures 7a, 7b, and 7c, for sending rate of 1 KBytes/s, 10 KBytes/s, and 100 KBytes/s, respectively. The results with QoS level 2 are reported in Figures 7d, 7e, and 7f, for sending rate of 1 KBytes/s, 10 KBytes/s, and 100 KBytes/s, respectively.

Regarding QoS level 0, we observe that for all the sending rates and almost all the forwarding probabilities, the percentage difference between MQTT and MQTT-A ranges from 3 to 5%. The worst case corresponds to a forward probability equal to 0.9 and sending rate of 100 KBytes/s. The corresponding percentage difference is less than 8%.

Considering QoS level 2, MQTT-A shows a slight worsening. Indeed, for almost all the sending rates and almost all the forwarding probabilities, the percentage difference is between 7-10%. The worst case corresponds to a forward probability 0.9 and sending rate of 100 KBytes/s in which the percentage difference is less than 18%.

However, high forwarding probabilities cannot be adopted also for security reasons (see Section IX). Therefore, we can conclude that the price in terms of goodput is not relevant. For completeness, we show in Figure 8, the goodput with forwarding probability equal to 0.67 and sending rate 10 KBytes/s. We can observe that the goodput is essentially the same for MQTT and MQTT-A and it is very close to the sending rate.

C. Latency

The second considered metric is the end-to-end latency measured between the instant in which a message is sent by the publisher and the instant in which it is received by the subscriber. We study how this latency varies as the forwarding probability of MQTT-A varies. We measured the latency for three different message sizes (100 bytes, 1000 bytes, and 10000 bytes), considering two QoS levels (0 and 2). The results are reported in Figures 9a, 9b, and 9c, respectively.

As a first observation, we see that there is no appreciable difference when the packet size changes in both protocols.

Second, the latency increases for both protocols by approximately the same factor (1.4-1.5), when the QoS level moves from 0 to 2.

Finally, as expected, the latency of MQTT-A increases with the forward probability (corresponding to longer paths). This is the main drawback of the proposed approach.

However, we have to consider that this condition guarantees both sender and recipient anonymity. If we relax this constraint and require just one of the two, the latency of MQTT-A is halved.

In Figure 9d, we set the forward probability to 0.67 and show the value of latency of MQTT (in blue) and MQTT-A when both sender and recipient anonymity are achieved (green) or when just one of two properties is supported (red).

We observe that in the first case (green bar) the ratio between the latency of MQTT-A and MQTT is less than 3, while in the second case, it is less than 1.5. This applies with minimum differences for the two QoS levels and packet sizes.

D. Supported Applications and Trade-offs of MQTT-A

The experiments performed in the previous sections show that the performance of MQTT-A depends on the length of the path crossed by MQTT messages. This is not surprising. Since the above results hold for a fixed RTT between bridge brokers (i.e., 55 ms), in this section, we repeat the above experiments by showing the performance of MQTT-A as the RTT varies.

Furthermore, since the performance decreases as p_f increases, we show how to set the value of p_f (given a value of RTT) to obtain a maximum latency in MQTT-A compliant with different applications. This allows us to draw some conclusions about which types of applications are supported by MQTT-A according to the network conditions (i.e., RTT) and security requirements (i.e., p_f).

Finally, in this section, we show the trade-off between security and performance of our solution.

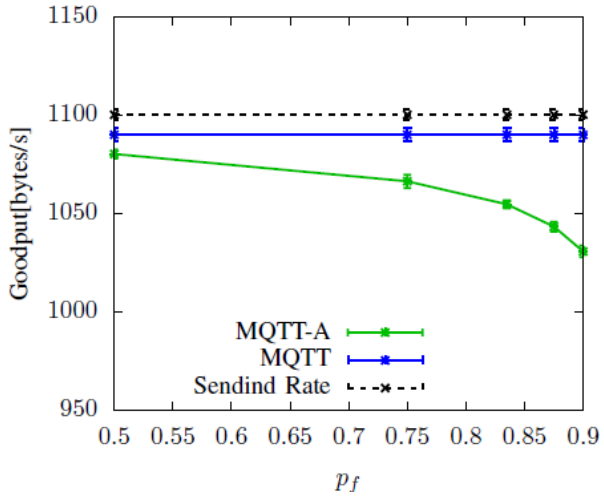
We start by studying latency as a function of RTT. The results, with QoS 0 and packet size equal to 100 bytes, are shown in Figure 10a. Therein, we plot three curves for three different p_f (i.e., 0.5, 0.835, and 0.9). We consider the values 0.5 and 0.9 since they represent the lower and upper end of the probability range considered in the above experiments. Moreover, we also consider an upper-intermediate value (0.835). Observe that, values lower than 0.5 do not meet any security requirement (any broker receiving a message can guess with a probability higher than 0.5 that the broker sending that message is directly connected to a broker in turn connected to a client). Concerning 0.9, as reported in Section IX, there is no advantage for too high values of p_f, from the security point of view. Moreover, values higher than 0.9 may result in unacceptable latencies. On the other hand, we also chose the value 0.835 to show that a small decrease in the probability value (from 0.9 to 0.835) results in a relevant improvement in performance. Observe that, these three probability values correspond to an average path length of 3, 7, and 11, respectively.

As expected, the latency increases linearly with the RTT. Concerning the p_f, a higher p_f corresponds to a higher latency. This is coherent with the results obtained in Section VIII-C.

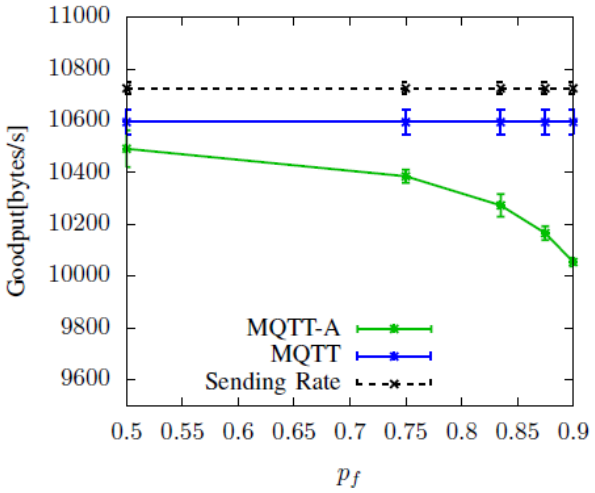
For completeness, we repeated the same experiment with QoS 2 and reported the results in the plots of Figure 10b. Clearly, the trend of the latency as a function of the RTT is the same (i.e., linear) but the absolute values are slightly higher. Again, this is coherent with the results of Section VIII-C.

Concerning the goodput, we report the results obtained with sending rate 1 KBytes/s and QoS 0 and 2, in Figures 11a and 11b, respectively. Again, we plot the goodput as a function of the RTT for three different values of p_f.

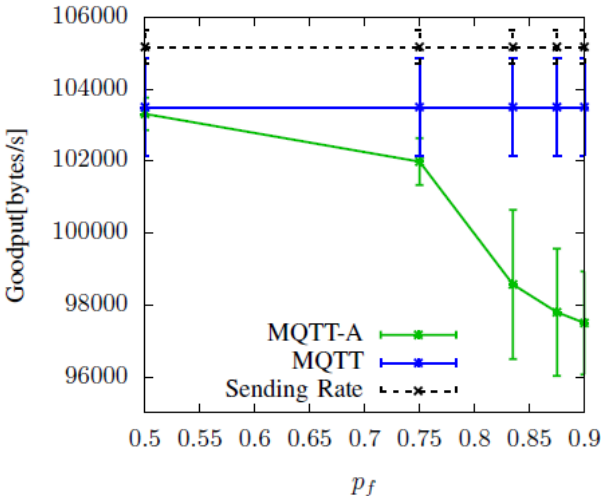
As expected, a higher RTT corresponds to a lower value of goodput. We observe that, with a high value of RTT, the results are less stable (high standard deviation). However, the trend is clearly downward with the RTT. In the worst case (i.e, QoS 2 and p_f =0.9) the goodput is about 65% of the sending rate.



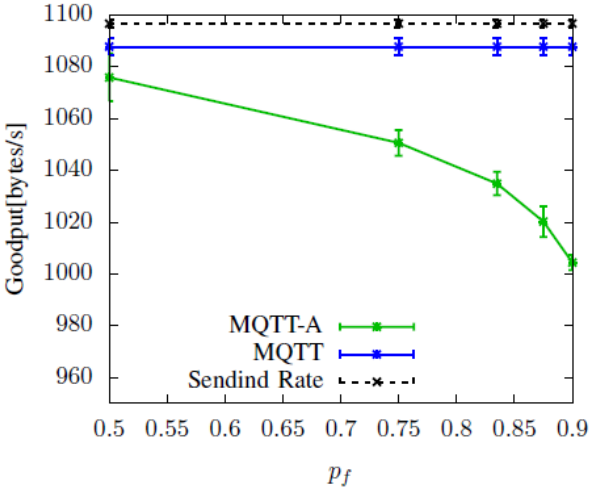
(a) Goodput with sending rate 1 KBytes/s and QoS level 0.



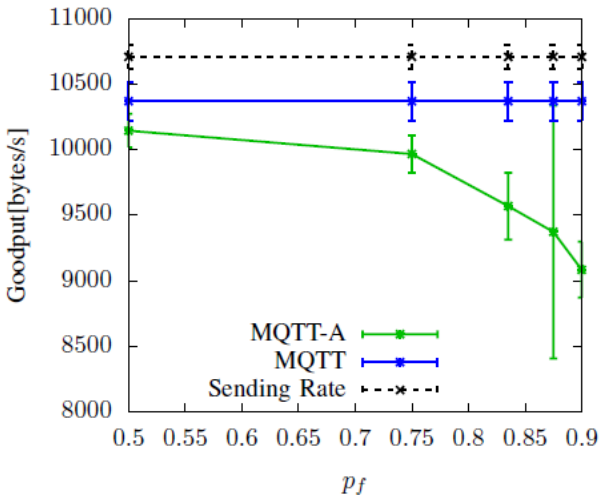
(b) Goodput with sending rate 10 KBytes/s and QoS level 0.



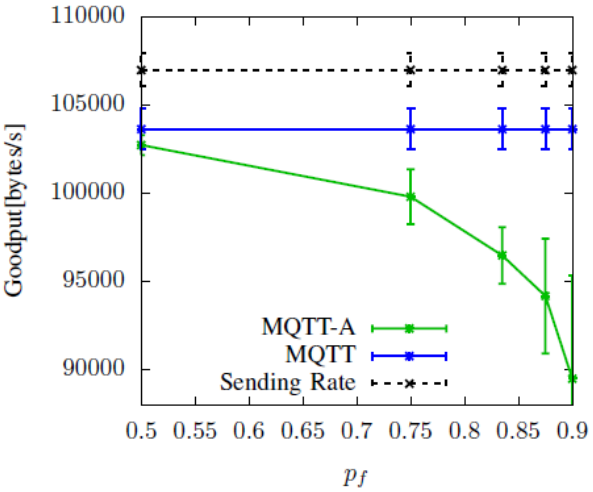
(c) Goodput with sending rate 100 KBytes/s and QoS level 0.



(d) Goodput with sending rate 1 KBytes/s and QoS level 2.



(e) Goodput with sending rate 10 KBytes/s and QoS level 2.



(f) Goodput with sending rate 100 KBytes/s and QoS level 2.

Fig. 7: Goodput.

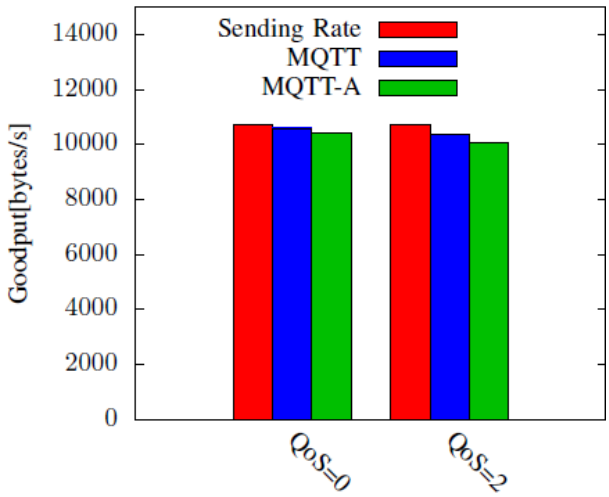


Fig. 8: Goodput with $p_f = 0.67$ and sending rate 10 KBytes/s.

From this analysis, it results, that according to the network conditions (RTT), security requirements (p_f), and QoS, our solution may be not always applicable (e.g., in the case of hard real-time applications).

To better investigate this point, in the following, we show how to set the maximum value of p_f to support different applications (with different requirements in terms of maximum tolerable latency). We used as a reference source the paper [51] to: (1) identify the type of applications, and (2) derive, for each type of application, an upper-bound value for latency. For each type of application, once the upper bound value is set, we obtain the maximum tolerated probability p_f for this type of application.

The three applications are the following (Table 6, in [51]):

1) Highly-Reliable WSN: supporting a latency of 3.090 seconds.
2) Default WSN: supporting a latency of 2.115 seconds.
3) Low-Latency WSN: supporting a latency of 0.336 seconds.

The results are reported in Figures 12a and 12b, for QoS 0 and 2, respectively. In the plots, we report the maximum p_f, given the latency requirement, for each application and a given value of RTT (representing the network condition). We consider p_f in the interval $[0.5, 0.9]$. As above discussed, values lower than 0.5 does not meet any security requirement. Then, we report p_f =0 to denote that no probability higher than 0.5 can support the considered application. Similarly, higher values than 0.9 do not introduce any security advantage. Therefore, even if higher p_f values than 0.9 can support the considered application, we set p_f to 0.9.

Concerning the Low-Latency WSN (continuous green line), we observe that this application can be implemented through MQTT-A only with good network performance (low RTT). The maximum tolerable value of p_f depends on the RTT and the QoS level. In advantageous conditions (QoS 0 and RTT=40ms), we can set p_f until 0.842.

On the other hand, the Highly-Reliable WSN (red line) can be implemented through MQTT-A in all the network conditions. Moreover, also the forward probability can be set to a high value. Observe that only in the extreme case of Qos 2 and RTT=260 ms, the maximum acceptable p_f is less than 0.9 (i.e., 0.873).

Finally, regarding the Default WSN (blue line), again MQTT-A is supported in all the network conditions with high p_f. However, we notice a slight worsening of the p_f value (compared to the case of High-Reliable WSN) in the case of RTT=260 ms both in QoS 0 and QoS 2.

Specifically, in this case, the maximum acceptable p_f (per RTT) is lower than the case of the Highly-Reliable WSN (but it still ranges between 0.8 and 0.9). Then, we can conclude that MQTT-A is fully supported also for the Default WSN application.

We would like to stress an important point. The values so far reported are obtained considering both sender and recipient anonymity. If we relax this condition requiring just one of the two properties, also the Low-Latency WSN application can be implemented through MQTT-A with higher values of RTT. In Figures 12a and 12b, we report the associated p_f for this application represented through a dashed green line.

To conclude this section, we report explicitly the trade-off between security and efficiency of MQTT-A. Specifically, according to Equation 2 of Section IX, in a network of n nodes, to resist against c colluding nodes, we have to set $p_f \geq \frac{\frac{1}{2}}{1-\frac{c}{n}-\frac{1}{n}} \simeq \frac{\frac{1}{2}}{1-\frac{c}{n}}$ (by assuming $n >> 1$).

On the other hand, as shown in Section VIII-C, as p_f increases, the total latency increases too. Then a trade-off on p_f exists.

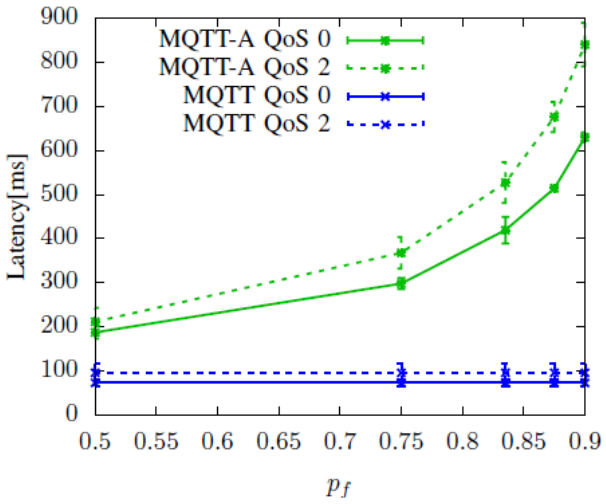
In Figures 13a and 13b, we report the latency obtained in correspondence with the forward probability allowing us to resist against the fraction $\frac{c}{n}$ of malicious brokers in the P2P network. The two figures consider QoS level 0 and QoS level 2, respectively, and include five plots, for different RTT values.

We chose these five values of RTT (40 ms, 55 ms, 140 ms, 180 ms, and 260 ms) since they cover a wide range of realistic network conditions when bridge brokers are distributed world-wide. Specifically, we obtained these values by deploying the brokers in different countries as explained in Section VIII-A.

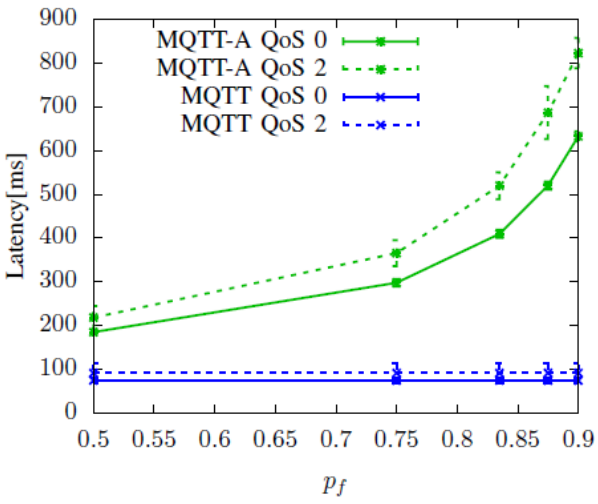
These plots allow us to set the probability p_f taking into account latency constraints, network conditions, and security requirements. We would like to highlight that the growth of the latency is very slow as the fraction $\frac{c}{n}$ increases, until a given value, then it increases exponentially. We observe that with no significant price in terms of latency, MQTT-A can tolerate up to 33% of nodes in the P2P network to be malicious.

IX. Threat Model and Security Analysis

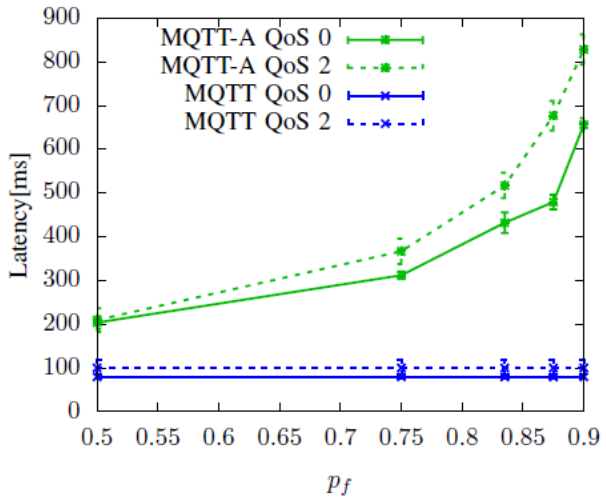
Since our anonymity protocol is based on [39], we consider the threat model of the original paper. However, a very significant difference exists. In [39], the recipient of the communication to protect is an end server that in our solution corresponds to a public broker. Anyway, such a public broker is not the actual recipient. Instead, the actual recipient is a subscriber to a topic on this public broker. Then, this difference has to be taken into account in our analysis. For example, in Table 1 of [39], when considering the protection of the



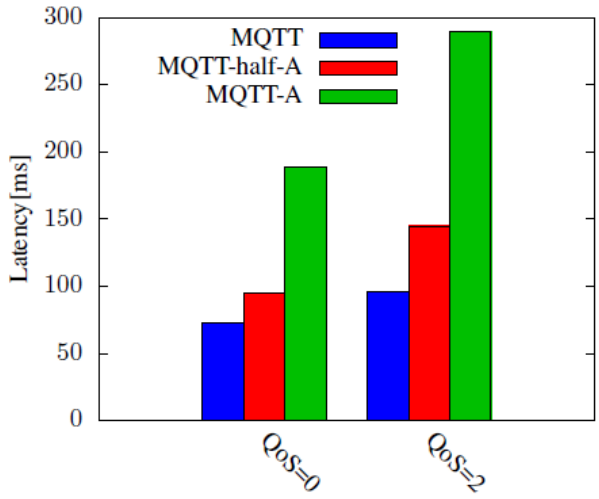
(a) Latency with packet size of 100 bytes.



(b) Latency with packet size of 1000 bytes.

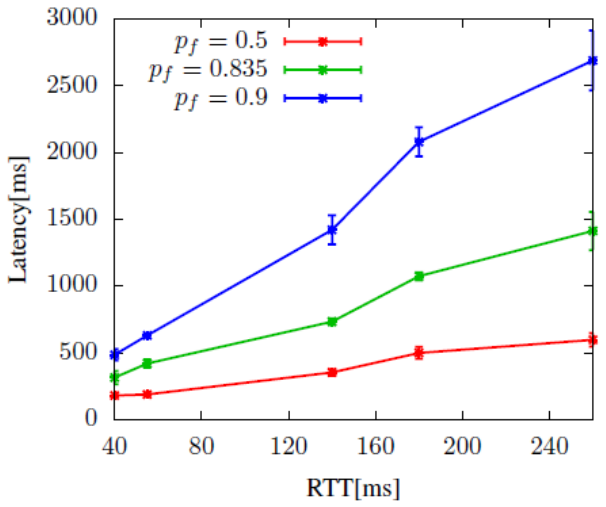


(c) Latency with packet size of 10000 bytes.

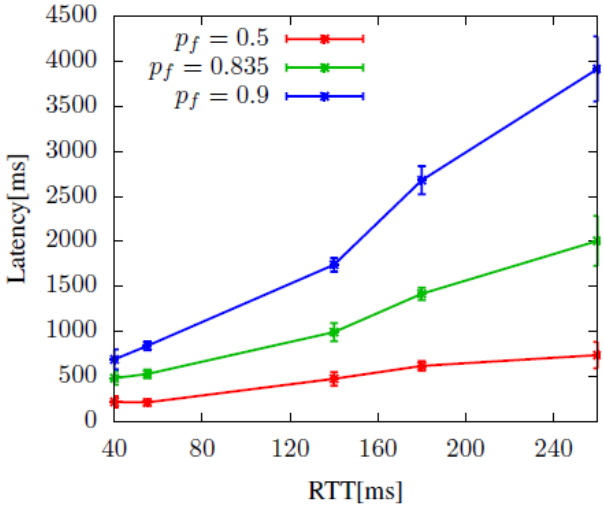


(d) Latency with $p_f = 0.67$ and packet size of 100 bytes.

Fig. 9: End-to-End Latency.

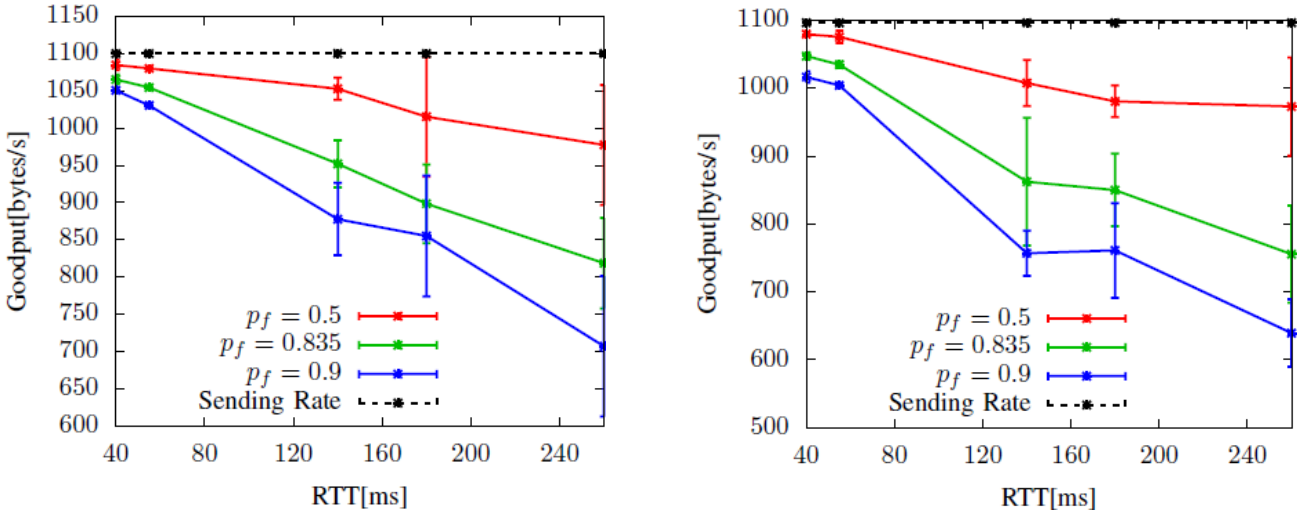


(a) Latency with QoS level 0.



(b) Latency with QoS level 2.

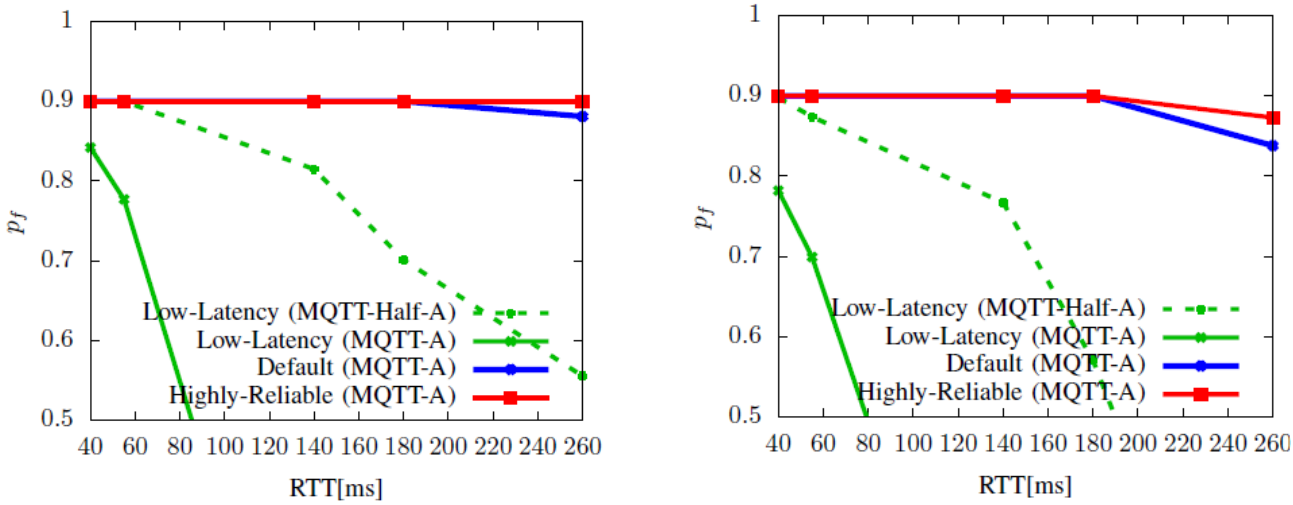
Fig. 10: End-to-End Latency.



(a) Goodput with QoS level 0.

(b) Goodput with QoS level 2.

Fig. 11: Goodput.



(a) QoS level 0.

(b) QoS level 2.

Fig. 12: Maximum probability for each application type, considering different RTT values.

recipient against the end server as an attacker, it results in N/A since the recipient is the end server itself. On the other hand, in our solution, it is not true anymore.

We now describe the considered threat model.

We consider the same adversaries and security properties of [39]. Clearly, they are properly adapted (and contextualized) to our approach to take into account the fact that the recipient of the communication is not a central server (i.e., the public broker), but a subscriber connected to a bridge broker participating in the P2P network.

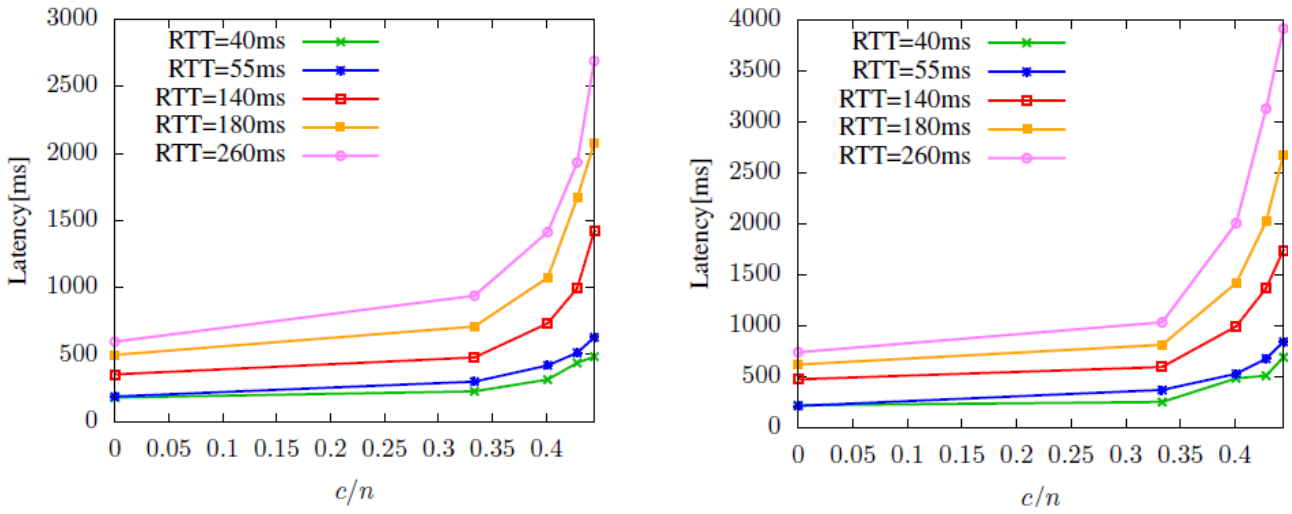
Attackers.

- **Local Eavesdropper**: An attacker that compromises the bridge broker directly connected to a publisher or subscriber.
- **Collaborating bridge brokers**: A set of bridge brokers, participating in the P2P network, that collaborate to identify publishers and subscribers.
- **Public broker**: The public broker hosting the actual topics that MQTT clients are interested in. It represents the end server of [39].

Clearly, as [39], our proposal is not oriented to the protection against a global adversary able to observe the entire flow of messages exchanged in the network. As a matter of fact, MQTT (and our proposal too) is designed for wide-area networks, in which the existence of a global adversary is unrealistic.

Security properties [37].

- **Sender Anonymity**: The identity of the publishers is hidden.
- **Recipient Anonymity**: The identity of the subscribers is hidden.
- **Relationship Anonymity**: The attacker cannot discover that a publisher and a subscriber are communicating with each other.



(a) QoS level 0. (b) QoS level 2.

Fig. 13: Trade-off between security and latency.

According to [37], it is sufficient to guarantee either sender anonymity or recipient anonymity to obtain relationship anonymity.

Now, we analyze how the attackers perform against the security properties. To do this, we consider the following scenario.

We have a publisher p directly connected to a bridge broker B_R^p that publishes data to the topic T_A hosted by the public broker B_P. Similarly, we consider a subscriber s directly connected to a bridge broker B_R^s interested in the topic T_A hosted by B_P.

In favor of security, we neglect the low level of anonymity introduced natively by the bridging mechanism and assume that sender anonymity is broken when the adversary identifies B_R^p (in place of p). Similarly, we assume that recipient anonymity is broken when the adversary identifies B_R^s (in place of s).

Resistance against the local eavesdropper. In this case, the attacker compromises either B_R^p or B_R^s. When the attacker is B_R^p, it can observe directly the incoming publish messages coming from p, then sender anonymity is not achieved. However, B_R^p is not able to identify the subscriber s interested in the offered topic. Indeed, even in the case in which B_R^p receives a subscribe message to forward towards B_P with the same topic, it cannot distinguish B_R^s from the other bridge brokers of the network. Therefore, if the P2P network is big enough recipient anonymity is guaranteed and then also relationship anonymity.

Similar considerations can be applied when the attacker is B_R^s. In this case, since it can observe directly the incoming subscribe messages coming from s, then recipient anonymity is not achieved. However, B_R^p cannot be identified by B_R^s, thus preserving sender anonymity and then relationship anonymity.

Resistance against collaborating bridge brokers. Regarding this type of adversary, [39] provides a detailed probabilistic analysis that can be applied also to our solution. For the sake of presentation, we report directly the two main results of the analysis and do not repeat the calculations.

The first result is that, given n nodes forming the peer network, we obtain *probable innocence* with respect to sender anonymity against c collaborators, if

$$n \geq \frac{p_f}{p_f - \frac{1}{2}}(c+1), \text{ where } p_f \geq \frac{1}{2} \tag{2}$$

Probable innocence means that, from the point of view of the attacker, the sender appears no more likely to be the originator of a message than to not be the originator.

According to this result, a higher probability p_f allows us to resist a higher number of corrupted nodes.

The second result is that if n is sufficiently high, we obtain *absolute privacy* for sender anonymity with a probability approaching 1. However, the growth of probability can be slow if p_f is large since it is more likely to involve a corrupted node in the path. Therefore, there exists a security trade-off regarding the value of the probability p_f.

Absolute sender privacy against an attacker means that the attacker cannot in any way distinguish the situations in which a potential sender actually sent communication from those in which it did not.

Clearly, in our application, the role of the sender considered in [39] is played both from B_R^p and B_R^s. As a consequence, if we have a sufficiently high number of bridge brokers, then our solution offers both sender and recipient anonymity and then relationship anonymity.

Resistance against the public broker. In this case, the attacker is B_P. Similarly to [39], sender anonymity is achieved since the publish message sent by B_R^p cannot be distinguished by B_P from a publish message originated from any other bridge broker. Unlike the previous adversary, this result does not depend on the probability p_f.

Regarding recipient anonymity, while in [39] it is not applicable since the recipient is the server itself, in our application we can consider the recipient from the point of view of B_P simply as a sender of a subscribe message. Then, by

applying the same reasoning done for B_R^p, also B_R^s cannot be distinguished from any other bridge broker.

Then, we obtain sender, recipient, and relationship anonymity against this adversary.

A. Beyond Anonymity: Active Attacks

The goal of this work is to propose a protocol able to guarantee anonymity to MQTT clients. Therefore, in previous analyses, we focused on adversaries eavesdropping on network messages (possibly colluding with each other) with the aim of identifying their senders and recipients.

Through this section, we discuss some other aspects which can be considered out of the scope of this paper since they do not affect the anonymity guarantees offered by MQTT-A. We stress that the issues that will be addressed in the following regard the standard MQTT protocol, as well as MQTT-A.

The first aspect we would like to address regards message confidentiality and integrity. To guarantee them, MQTT (as well as MQTT-A) should be used over TLS/SSL. However, this solution provides message confidentiality and integrity only between clients and brokers (or between brokers). Unfortunately, no standard end-to-end (i.e., between clients) solution exists for the MQTT protocol. This implies that a malicious broker could read the content of any MQTT message in the clear or modify it without authorization.

To counter such attacks, clients should exchange keys between them, possibly through other protocols. Obviously, concerning our protocol, if clients require to be anonymous only with respect to brokers (thus not needing to be anonymous with respect to each other) the above-mentioned approach can be applied to MQTT-A. However, if this is not the case, clients should be able to exchange keys in an anonymous form. This is currently an open problem also in standard MQTT.

Concerning the lack of message integrity, some other relevant attacks are possible. In particular, a malicious broker may tamper with the correct message order, by either replaying messages multiple times or by simply delaying or discarding messages. A way clients can detect the occurrence of such attacks is by including a counter within each message (see Section VI). Nevertheless, this solution requires that message integrity is (end-to-end) preserved, otherwise a malicious broker can easily change the counter, thus making the attack undetectable.

However, even in the absence of a solution that ensures the end-to-end integrity of messages, MQTT-A clients can take a trivial countermeasure to detect whether they are victims of active attacks. For instance, a publisher can (anonymously) subscribe to the public broker, hosting the topic to which it is publishing, to detect the presence of any unauthorized tampering attempts in the path from the publisher to the public broker. Obviously, this subscription can be done multiple times. Indeed, because of the probability p_f and the random choice of peer brokers, each time the subscription path would be different. This way, if the publisher sees the same messages, that it published before, coming from at least one of the subscription paths, then it can conclude that it is not a victim of active attacks. Obviously, a similar solution can be also employed by subscribers to detect attempts of active attacks in the path from the public broker to themselves. Again, the subscriber can make more than one subscription to the same topic (thus exploiting different paths for each subscription). This way, by comparing the received messages, it can detect whether it is a victim of active attacks.

X. Related Work

Security in MQTT is an open problem [52]–[54]. On the one hand, implementing security mechanisms is crucial to protect end-to-end clients. On the other hand, since MQTT is adopted when constrained devices are involved [55], complex security solutions cannot be applied. Therefore, MQTT does not provide any built-in security mechanism.

A first issue regards *confidentiality*. The basic approach consists of using TLS to establish secure channels [56]. However, it has a negative impact on performance and energy consumption [57]. Therefore, more advanced solutions have been proposed in the literature [14], [58]–[64].

Often, the confidentiality mechanisms are adopted to reach also *authorization* (with a focus on *access control*) [65], [66]. A lot of works pursuing both authorization and confidentiality are based on CP-ABE or KP-ABE [67]–[70]. Clearly, these schemes differ from the standard ABE schemes since they are tailored for being used by constrained devices.

Another security feature investigated in MQTT is *authentication* [71]–[73]. For example, in [74], a multi-factor blockchain-based solution for authenticating clients is provided.

The problem of *privacy* in MQTT [75], [76] is more related to our work. An interesting solution aimed to obfuscate the topics when public brokers are involved is provided in [77]. However, it requires pre-shared keys between clients making this solution not compliant with more general scenarios (such as that described in this paper) in which publishers and subscribers do not know each other. The same consideration applies for [76]. In principle, the above techniques aim to protect the content of communication, while our proposal is devoted to protecting peer identities. Clearly, the two approaches are not in contrast and could be combined to obtain a higher level of privacy.

The approaches so far described are all designed for an MQTT-based scenario. There are some other approaches that, even though not directly applicable to MQTT (but to the IoT domain in general), are worth mentioning since they are related to our proposal.

[78]–[80] belong to both privacy and access control categories. For example, [78] proposes a lightweight policy-based privacy-preserving scheme that aims at giving users control over their privacy by setting their own criteria for data collection.

[81]–[83] belong to both privacy and authentication categories. Specifically, they propose an enhancement of the authentication mechanism enabling the cloud server to authenticate devices anonymously. However, unlike our proposal, these solutions do not focus directly on the identification of clients through network metadata, such as the IP address.

TABLE III: Summary of the related literature

Categories	References
Confidentiality	[14], [56]–[64]
Authorization (access control)	[65]–[70]
Authentication	[71]–[74]
Privacy	[75]–[83]
Anonymity	[84] and our proposal

TABLE IV: Abbreviation Table

Abbreviation	Definition
AWS	Amazon Web Services
CP-ABE	Cypher-Policy Attribute-Based Encryption
DH	Diffie-Hellman
IP address	Internet Protocol address
KP-ABE	Key-Policy Attribute-Based Encryption
MQTT	Message Queue Telemetry Transport
MQTT-A	Anonymous Message Queue Telemetry Transport
P2P	Peer to Peer
QoS	Quality of Service
RTT	Round-Trip Time
WSN	Wireless Sensor Network

We would like to point out that, even though the above solutions achieve different goals from ours, in principle, they may be used in conjunction with MQTT-A to strengthen the privacy protection offered to users. This will be the object of future work.

The exact context in which our paper falls is *anonymity*, in which the aim is to prevent the identification of clients publishing or subscribing to some topics.

In the literature, to the best of our knowledge, no relevant and complete proposal is available in this direction. The only approach that presents some similarities with our proposal is [84]. Therein, a Tor-like [34] solution is designed in which clients connect to brokers through a path of intermediate brokers that forward messages encrypted in Onion fashion. Different from standard Tor, this approach does not require a set-up phase to build a tunnel of brokers. Thus, the message has to include the next broker (along with a Diffie-Hellman (DH) parameter) in each layer of encryption. This may result in a not negligible overhead in terms of the size of the packet. Moreover, [84] assumes that the public DH parameter of each broker is known in advance by MQTT clients. This is not a trivial task, so a proper discovery protocol (similar to the one proposed in this paper) should be provided. Finally, the approach proposed in [84] suffers from high computational overhead, as reported in the introduction (for standard Tor and I2P), because of the layered encryption. Indeed, it may be not acceptable for resource-constrained MQTT clients. Such aspects have not been considered in [84]. Indeed, in the short paper, only the rough idea of the approach is presented without providing any implementation or experimental validation. Furthermore, no discovery protocol and no security analysis are included.

We summarize the literature mentioned in this section in Table III.

XI. Conclusions

Nowadays collecting IoT data allows companies to improve the service they offer to their customers. To this aim, for instance, companies can massively collect data from smart home devices. However, this also allows companies to track and profile users since IoT devices may reveal their behaviors and preferences. Of course, this represents an important privacy issue. One way to ensure users' privacy, while preserving the right of companies to collect data, is to have users' IoT devices communicate with companies anonymously.

This is exactly the contribution of this paper. Specifically, our work regards the MQTT protocol, since it is very often employed in collecting users' personal data via resource-constrained devices in various contexts such as smart homes, and smart cities.

In this paper, we proposed a P2P anonymous network to prevent the identification of MQTT clients. The idea is to propagate publish/subscribe messages through a random path of intermediate bridge brokers so that the final public broker cannot identify the actual sender. An important feature achieved in this proposal is that all the messages are exchanged by following the standard MQTT protocol in bridging mode. This allows us to apply our solution without requiring infrastructure changes to the standard MQTT architecture. The experimental validation shows that there exists a trade-off between security and latency. Moreover, no relevant price has to be paid in terms of goodput in case of good network conditions. The above anonymity goal is reached in the paper also by providing a discovery protocol allowing clients to know the public brokers and the topics they offer. This is very important when anonymity of clients is pursued since they cannot expose their interest in topics. Again, also this protocol is performed entirely using the standard MQTT primitives.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] OASIS, "MQTT version 3.1.1," vol. 1, 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3

[3] H. AP and K. Kanagasabai, "Secure-MQTT: an efficient fuzzy logic-based approach to detect DoS attack in MQTT protocol for internet of things," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–15, 2019.

[4] R. K. Kodali and S. Soratkal, "MQTT based home automation system using ESP8266," in *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*. IEEE, 2016, pp. 1–5.

[5] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE, 2008, pp. 791–798.

[6] C.-S. Yeh, S.-L. Chen, and I.-C. Li, "Implementation of MQTT protocol based network architecture for smart factory," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 235, no. 13, pp. 2132–2142, 2021.

[7] A. Lachtar, T. Val, and A. Kachouri, "Elderly monitoring system in a smart city environment using LoRa and MQTT," *IET Wireless Sensor Systems*, vol. 10, no. 2, pp. 70–77, 2020.

[8] L. Zhang, "Building Facebook Messenger," https://www.facebook.com/notes/10158791547142200/, accessed: 2022-05-04.

[9] L. Nastase, "Security in the Internet of Things: A Survey on Application Layer Protocols," in *2017 21st international conference on control systems and computer science (CSCS)*. IEEE, 2017, pp. 659–666.

[10] K. M. Alam and A. Akram, "A Survey on MQTT Protocol for the Internet of Things," *Khulna University, Dept. of Computer Science and Engineering (CSE)*, 2016.

[11] S. Nazir and M. Kaleem, "Reliable Image Notifications for Smart Home Security with MQTT," in *2019 International Conference on Information Science and Communication Technology (ICISCT)*. IEEE, 2019, pp. 1–5.

[12] H. C. Hwang, J. Park, and J. G. Shon, "Design and implementation of a reliable message transmission system based on MQTT protocol in IoT," *Wireless Personal Communications*, vol. 91, no. 4, pp. 1765–1777, 2016.

[13] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018.

[14] S. Shin, K. Kobara, C.-C. Chuang, and W. Huang, "A security framework for MQTT," in *2016 IEEE Conference on Communications and Network Security (CNS)*, 2016, pp. 432–436.

[15] M. A. A. da Cruz, J. J. P. C. Rodrigues, P. Lorenz, V. V. Korotaev, and V. H. C. de Albuquerque, "In.IoT—A New Middleware for Internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 7902–7911, 2021.

[16] D. Uroz and R. J. Rodríguez, "Characterization and Evaluation of IoT Protocols for Data Exfiltration," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[17] B. B. Gupta and M. Quamara, "An overview of Internet of Things (IoT): architectural aspects, challenges, and protocols," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 21, p. e4946, 2020.

[18] A. Tewari and B. B. Gupta, "Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework," *Future generation computer systems*, vol. 108, pp. 909–920, 2020.

[19] ——, "Cryptanalysis of a novel ultra-lightweight mutual authentication protocol for IoT devices using RFID tags," *The Journal of Supercomputing*, vol. 73, no. 3, pp. 1085–1102, 2017.

[20] J. Srinivas, S. Mukhopadhyay, and D. Mishra, "Secure and efficient user authentication scheme for multi-gateway wireless sensor networks," *Ad Hoc Networks*, vol. 54, pp. 147–169, 2017.

[21] Y. Qiu and M. Ma, "A mutual authentication and key establishment scheme for M2M communication in 6LoWPAN networks," *IEEE transactions on industrial informatics*, vol. 12, no. 6, pp. 2074–2085, 2016.

[22] F. Wu, X. Li, L. Xu, S. Kumari, M. Karuppiah, and J. Shen, "A lightweight and privacy-preserving mutual authentication scheme for wearable devices assisted by cloud server," *Computers & Electrical Engineering*, vol. 63, pp. 168–181, 2017.

[23] A. Ukil, S. Bandyopadhyay, and A. Pal, "IoT-privacy: To be private or not to be private," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2014, pp. 123–124.

[24] P. Emami-Naeini, Y. Agarwal, L. F. Cranor, and H. Hibshi, "Ask the experts: What should be on an IoT privacy and security label?" in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 447–464.

[25] R. Chow, "The last mile for IoT privacy," *IEEE Security & Privacy*, vol. 15, no. 6, pp. 73–76, 2017.

[26] P. Porambage, M. Ylianttila, C. Schmitt, P. Kumar, A. Gurtov, and A. V. Vasilakos, "The quest for privacy in the internet of things," *IEEE Cloud Computing*, vol. 3, no. 2, pp. 36–45, 2016.

[27] M. Chanson, A. Bogner, D. Bilgeri, E. Fleisch, and F. Wortmann, "Blockchain for the IoT: privacy-preserving protection of sensor data," *Journal of the Association for Information Systems*, vol. 20, no. 9, pp. 1274–1309, 2019.

[28] A. Al-Qerem, M. Alauthman, A. Almomani, and B. B. Gupta, "IoT transaction processing through cooperative concurrency control on fog–cloud computing environment," *Soft Computing*, vol. 24, no. 8, pp. 5695–5711, 2020.

[29] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power IoT edge devices," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 7–12.

[30] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate IoT edge computing with lightweight virtualization," *IEEE Network*, vol. 32, no. 1, pp. 102–111, 2018.

[31] A. Ioannis, C. Chrysostomos, and H. George, "Internet of things: Security vulnerabilities and challenges," in *2015 IEEE symposium on computers and communication (ISCC)*. IEEE, 2015, pp. 180–187.

[32] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of threats? a survey of practical security vulnerabilities in real IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[33] K. Chen, S. Zhang, Z. Li, Y. Zhang, Q. Deng, S. Ray, and Y. Jin, "Internet-of-things security and vulnerabilities: Taxonomy, challenges, and practice," *Journal of Hardware and Systems Security*, vol. 2, no. 2, pp. 97–110, 2018.

[34] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Research Lab Washington DC, Tech. Rep., 2004.

[35] B. Zantout, R. Haraty *et al.*, "I2P data communication system," in *Proceedings of ICN*. Citeseer, 2011, pp. 401–409.

[36] G. Danezis and C. Diaz, "A survey of anonymous communication channels," Technical Report MSR-TR-2008-35, Microsoft Research, Tech. Rep., 2008.

[37] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management," http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf, Aug. 2010, v0.34. [Online]. Available: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf

[38] A. Hue, G. Sharma, and J.-M. Dricot, "Privacy-Enhanced MQTT Protocol for Massive IoT," *Electronics*, vol. 11, no. 1, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/1/70

[39] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM transactions on information and system security (TISSEC)*, vol. 1, no. 1, pp. 66–92, 1998.

[40] A. Stanford-Clark and H. L. Truong, "MQTT for sensor networks (MQTT-SN) protocol specification," *International business machines (IBM) Corporation version*, vol. 1, no. 2, pp. 1–28, 2013.

[41] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert, and A. Knoll, "OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols," in *2019 IEEE International Conference on Industrial Technology (ICIT)*, 2019, pp. 955–962.

[42] G. Kim, S. Kang, J. Park, and K. Chung, "An MQTT-Based Context-Aware Autonomous System in oneM2M Architecture," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8519–8528, 2019.

[43] T. C. Piller, D. M. Merz, and A. Khelil, "MQTT-4EST: RulE-basEd WEb Editor for Semantic-aware Topic Naming in MQTT," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2022, pp. 1–8.

[44] T. Dalenius, "Finding a needle in a haystack or identifying anonymous census records," *Journal of official statistics*, vol. 2, no. 3, p. 329, 1986.

[45] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 3–es, 2007.

[46] E. Longo, A. E. Redondi, M. Cesana, and P. Manzoni, "BORDER: a Benchmarking Framework for Distributed MQTT Brokers," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[47] R. Soua, M. R. Palattella, A. Stemper, and T. Engel, "MQTT-MFA: a Message Filter Aggregator to Support Massive IoT Traffic Over Satellite," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[48] HiveMQ, "Hive MQ Community Edition," https://github.com/hivemq/hivemq-community-edition/wiki, 2022, last checked: 14/11/2022.

[49] Digital Ocean, "Digital Ocean Cloud platform," https://docs.digitalocean.com/, 2022, last checked: 14/11/2022.

[50] HiveMQ, "HiveMQ MQTT Client," https://hivemq.github.io/hivemq-mqtt-client/, 2022, last checked: 14/11/2022.

[51] S. Scanzio, M. G. Vakili, G. Cena, C. G. Demartini, B. Montrucchio, A. Valenzano, and C. Zunino, "Wireless sensor networks and TSCH: A compromise between reliability, power consumption, and latency," *IEEE Access*, vol. 8, pp. 167 042–167 058, 2020.

[52] C. Patel and N. Doshi, "A novel MQTT security framework in generic IoT model," *Procedia Computer Science*, vol. 171, pp. 1399–1408, 2020.

[53] G. Perrone, M. Vecchio, R. Pecori, R. Giaffreda *et al.*, "The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices." in *IoTBDS*, 2017, pp. 246–253.

[54] D. Mendez Mena, I. Papapanagiotou, and B. Yang, "Internet of things: Survey on security," *Information Security Journal: A Global Perspective*, vol. 27, no. 3, pp. 162–182, 2018.

[55] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," in *2014 IEEE ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP)*. IEEE, 2014, pp. 1–6.

[56] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," *RFC*, vol. 2246, pp. 1–80, 1999.

[57] T. Prantl, L. Iffländer, S. Herrnleben, S. Engel, S. Kounev, and C. Krupitzer, "Performance impact analysis of securing MQTT using TLS," in *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, 2021, pp. 241–248.

[58] S. P. Mathews and R. R. Gondkar, "Protocol Recommendation for Message Encryption in MQTT," in *2019 International Conference on Data Science and Communication (IconDSC)*, 2019, pp. 1–5.

[59] W.-T. Su, W.-C. Chen, and C.-C. Chen, "An Extensible and Transparent Thing-to-Thing Security Enhancement for MQTT Protocol in IoT Environment," in *2019 Global IoT Summit (GIoTS)*, 2019, pp. 1–4.

[60] J. Ahamed, M. Zahid, M. Omar, and K. Ahmad, "AES and MQTT based security system in the internet of things," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 8, pp. 1589–1598, 2019.

[61] O. Sadio, I. Ngom, and C. Lishou, "Lightweight Security Scheme for MQTT/MQTT-SN Protocol," in *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, 2019, pp. 119–123.

[62] D. Dinculeană and X. Cheng, "Vulnerabilities and Limitations of MQTT Protocol Used between IoT Devices," *Applied Sciences*, vol. 9, no. 5, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/5/848

[63] A. Oak and R. Daruwala, "Assessment of Message Queue Telemetry and Transport (MQTT) protocol with Symmetric Encryption," in *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, 2018, pp. 5–8.

[64] S. Iyer, G. V. Bansod, P. N. V, and S. Garg, "Implementation and Evaluation of Lightweight Ciphers in MQTT Environment," in *2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, 2018, pp. 276–281.

[65] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul, and A. Panya, "Authorization mechanism for MQTT-based Internet of Things," in *2016 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2016, pp. 290–295.

[66] M. Michaelides, C. Sengul, and P. Patras, "An Experimental Evaluation of MQTT Authentication and Authorization in IoT," in *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization*, 2022, pp. 69–76.

[67] V. Gupta, S. Khera, and N. Turk, "MQTT protocol employing IOT based home safety system with ABE encryption," *Multimedia Tools and Applications*, vol. 80, no. 2, pp. 2931–2949, 2021.

[68] L. Bisne and M. Parmar, "Composite secure MQTT for Internet of Things using ABE and dynamic S-box AES," in *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*. IEEE, 2017, pp. 1–5.

[69] F. Mendoza-Cardenas, R. S. Leon-Aguilar, and J. L. Quiroz-Arroyo, "CP-ABE encryption over MQTT for an IoT system with Raspberry Pi," in *2022 56th Annual Conference on Information Sciences and Systems (CISS)*, 2022, pp. 236–239.

[70] T.-L. Liao, H.-R. Lin, P.-Y. Wan, and J.-J. Yan, "Improved Attribute-Based Encryption Using Chaos Synchronization and Its Application to MQTT Security," *Applied Sciences*, vol. 9, no. 20, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/20/4454

[71] M. Calabretta, R. Pecori, M. Vecchio, and L. Veltri, "MQTT-Auth: A token-based solution to endow MQTT with authentication and authorization capabilities," *Journal of Communications Software and Systems*, vol. 14, no. 4, pp. 320–331, 2018.

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

[73] R. S. Bali, F. Jaafar, and P. Zavarasky, "Lightweight authentication for MQTT to improve the security of IoT communication," in *Proceedings of the 3rd International Conference on Cryptography Security, and Privacy*, 2019, pp. 6–12.

[74] F. Buccafurri, V. De Angelis, and R. Nardone, "Securing MQTT by blockchain-based OTP authentication," *Sensors*, vol. 20, no. 7, p. 2002, 2020.

[75] J. J. Anthraper and J. Kotak, "Security, privacy and forensic concern of MQTT protocol," in *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India*, 2019.

[76] A. Hue, G. Sharma, and J.-M. Dricot, "Privacy-Enhanced MQTT Protocol for Massive IoT," *Electronics*, vol. 11, no. 1, p. 70, 2021.

[77] M. Fischer, D. Kümper, and R. Tönjes, "Towards improving the Privacy in the MQTT protocol," in *2019 Global IoT Summit (GIoTS)*. IEEE, 2019, pp. 1–6.

[78] M. Chehab and A. Mourad, "Towards a Lightweight Policy-Based Privacy Enforcing Approach for IoT," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2018, pp. 984–989.

[79] K. Fan, H. Xu, L. Gao, H. Li, and Y. Yang, "Efficient and privacy preserving access control scheme for fog-enabled IoT," *Future Generation Computer Systems*, vol. 99, pp. 134–142, 2019.

[80] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "Towards a novel privacy-preserving access control model based on blockchain technology in IoT," in *Europe and MENA cooperation advances in information and communication technologies*. Springer, 2017, pp. 523–533.

[81] R. Vinoth, L. J. Deborah, P. Vijayakumar, and B. B. Gupta, "An Anonymous Pre-Authentication and Post-Authentication Scheme Assisted by Cloud for Medical IoT environments," *IEEE Transactions on Network Science and Engineering*, 2022.

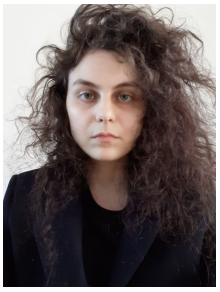
[82] A. K. Singh, A. Nayyar, and A. Garg, "A secure elliptic curve based anonymous authentication and key establishment mechanism for IoT and cloud," *Multimedia Tools and Applications*, pp. 1–52, 2022.

[83] A. Rasheed, R. R. Hashemi, A. Bagabas, J. Young, C. Badri, and K. Patel, "Configurable anonymous authentication schemes for the Internet of Things (IoT)," in *2019 IEEE International Conference on RFID (RFID)*. IEEE, 2019, pp. 1–8.

[84] Y. Protskaya and L. Veltri, "Broker Bridging Mechanism for Providing Anonymity in MQTT," in *2019 10th International Conference on Networks of the Future (NoF)*, 2019, pp. 110–113.

Francesco Buccafurri. Full professor of computer science at the University Mediterranea of Reggio Calabria, Italy. In 1995 he took the PhD degree in CS at the University of Calabria. In 1996 he was visiting researcher at Vienna University of Technology. His research interests include cybersecurity, privacy, social networks, e-government, and P2P systems. He has published more than 160 papers in top-level journals and conference proceedings. He serves as a referee for international journals and he is a member of several conference PCs. He is Associate Editor of Information Sciences (Elsevier) and IEEE Transactions on Industrial Informatics and played the role of PC chair and PC member in many international conferences. He is member of the IEEE computer society.

Vincenzo De Angelis. PhD student in information engineering at the University Mediterranea of Reggio Calabria, Italy. He received the BS degree in information engineering and the Master's degree in telecommunication engineering in 2017 and 2019, respectively. His research interests include information security, blockchain, cloud, and applied cryptography. He is author of a number of papers published in international journals and conference proceedings. He was PC member of a number of conferences and Guest Editor of a special issue in an international Journal.



Sara Lazzaro. PhD student in information engineering at the University Mediterranea of Reggio Calabria, Italy. She received the Master's degree in telecommunication engineering in 2021. Her research interests include information security and privacy. She is author of a number of papers published in international journals and international conference proceedings.