electronics

MDPI

Article

Privacy-Enhanced MQTT Protocol for Massive IoT

Axelle Hue [1,*], Gaurav Sharma [2,*] and Jean-Michel Dricot [1,*]

[1] Ecole Polytechnique de Bruxelles, Université Libre de Bruxelles, 1050 Brussels, Belgium
[2] Département d'Informatique, Université Libre de Bruxelles, 1050 Brussels, Belgium
[*] Correspondence: hue.axelle@gmail.com (A.H.); gaurav.sharma@ulb.be (G.S.); jean-michel.dricot@ulb.be (J.-M.D.)

Abstract: The growing expectations for ubiquitous sensing have led to the integration of countless embedded sensors, actuators, and RFIDs in our surroundings. Combined with rapid developments in high-speed wireless networks, these resource-constrained devices are paving the road for the Internet-of-Things paradigm, a computing model aiming to bring together millions of heterogeneous and pervasive elements. However, it is commonly accepted that the *Privacy* consideration remains one of its main challenges, a notion that does not only encompasses malicious individuals but can also be extended to honest-but-curious third-parties. In this paper, we study the design of a privacy-enhanced communication protocol for lightweight IoT devices. Applying the proposed approach to MQTT, a highly popular lightweight *publish/subscribe* communication protocol prevents no valuable information from being extracted from the messages flowing through the broker. In addition, it also prevents partners re-identification. Starting from a privacy-ideal, but unpractical, exact transposition of the Oblivious Transfer (OT) technology to MQTT, this paper follows an iterative process where each previous model's drawbacks are appropriately mitigated all the while trying to preserve acceptable privacy levels. Our work provides resistance to statistical analysis attacks and dynamically supports new client participation. Additionally the whole proposal is based on the existence of a non-communicating 3rd party during pre-development. This particular contribution reaches a *proof-of-concept* stage through implementation, and achieves its goals thanks to OT's *indistinguishability* property as well as hash-based topic obfuscations.

Keywords: IoT; MQTT; privacy; oblivious transfer

check for updates

Citation: Hue, A.; Sharma, G.; Dricot, J.-M. Privacy-Enhanced MQTT Protocol for Massive IoT. *Electronics* **2022**, *11*, 70. https://doi.org/10.3390/electronics11010070

Academic Editor: Dongkyun Kim

Received: 23 November 2021
Accepted: 20 December 2021
Published: 27 December 2021

1. Introduction

The future goal of computing is to surround us. This is particularly true as the numbers of devices connected to the internet and disseminated all around us or in our homes has been steadily growing for the past few years. Combined with the rapid development of high-speed wireless networks, the booming expectations for ubiquitous data processing have paved the road for the emergence of a new computing paradigm: the *Internet-of-Things* (IoT). Via a complex network of sensors, actuators, and Radio Frequency Identification Tags (RFID) smoothly embedded in all domains of our daily lives, this model looks to achieve extensive data collection and processing in order to create a global operating picture of the physical reality and improve our quality of life [1] all the while minimizing human involvement [2]. Aiming to regroup billions of heterogeneous devices together, the IoT is expected to connect 75.4 billion elements by 2025, for an annual global economic impact going from 2.5 to 6.2 trillions USD [2].

While this paradigm presents a fertile ground for numerous innovative and pervasive applications, the amount and the nature of all data shared between the various platforms, from location tags to health monitoring, have brought forth the necessity for strict data handling requirements. In particular, a survey conducted in 2013 by IEEE highlighted that 46% of all respondents considered privacy issues to be the main obstacles to the widespread use of IoT devices [3]. While privacy laws such as the EU *General Data Protection*

Regulation (GDPR) maintain an essential role in ensuring service providers' compliance and accountability, it is arguable that their underlying principles ought to be directly represented in the technology [1] and hence guarantee a form of "inherent" privacy protection. Such results can be obtained thanks to the efficient implementation and combination of Privacy-Enhancing Technologies (PET). However, it is worth noting that the definitions of privacy threat are not only limited to the scope of malicious adversaries. Indeed, recent years have also unraveled the financial gain a service provider can obtain from shared user information (e.g., in advertisement profit). Assuming such *honest-but-curious third-party* models, it is conceivable that extra privacy-preserving considerations should be applied to these scenarios.

Given the highly popular application-layer Message Queue Telemetry Transport (MQTT) protocol [4], privacy threats can then be considered at the scale of the protocol's broker, i.e., the intermediary server in charge of redirecting the data transmitted by one device to an arbitrary large number of interested clients. The bottleneck of all communications, the MQTT broker is indeed by default able to read any messages relying on its forwarding services, to identify their source and their destination. Our contribution hence resides in an actual privacy-enhanced communication protocol where messages are encrypted to preserve their confidentiality. Inspired from the OT-PET, this design also prevents partners re-identification. In other words, by observing all flowing messages, the broker is unable to re-associate both ends of the communication, meaning that our contribution ensures that an honest-but-curious third-party cannot identify who is communicating and with whom. Additionally, the privacy of message content and subject is also protected. Finally, it also presents the highly practical advantage to be compatible with the pre-existing MQTT v5.0 standard, by only requiring actions on the client ends. One objective of this work is to design a protocol keeping lightweight IoT devices in mind while still meeting the target privacy level.

The rest of the paper is organized as follows: Section 2 provides a detailed overview on the MQTT protocol, its threat model *vis-à-vis* an honest-but-curious MQTT broker and existing solutions. Subsequently, Section 3 presents the details about oblivious transfer and proposed communication protocol based on OT-MQTT. Finally, Section 4 summarizes this protocol's implementation, and this work is concluded in Section 5.

2. MQTT and Threat Model

MQTT is an application-layer communication protocol built on the TCP/IP transport protocol. MQTT is suitable for IoT lightweight devices and achieves low network bandwidth by minimizing header size. Equipped with bidirectional communications, this protocol guaranties high degrees of reliability. MQTT also provides optional security with tools such as TLS and OAuth authentication. The MQTT communication approach is based on a highly decoupled structure: the *publish/subscribe* model. Concretely, this amounts to a system where both ends of the communication are *autonomous* (i.e., both have no notions about the existence of the other) and rely on a third party, called the *broker*, to distribute the messages to all its intended destinations [5]. In particular, this aspect plays a significant role in the scalability potential of this protocol: the sender only needs to send its data once without knowing how many clients will be served by the broker. Moreover, adding an additional client to the scheme is completely transparent.

On the one hand, a device called a *Publisher* pushes (or "*publishes*") its data towards the broker on a certain path named a "*topic*" (e.g., 'temperature'). On the other hand, any *Subscriber* interested in this specific type of data notifies its interest in the aforementioned topic by *subscribing* to it. In between, as an intermediary, the broker centralizes, filters, and correctly redistributes each newly available message. In particular, this aspect plays a significant role in the scalability potential of this protocol: the sender only needs to send its data once without knowing how many clients will be served by the broker. Moreover, adding an additional client to the scheme is completely transparent. As a consequence of this approach, please note that correct topic definitions are essential to support adequate filtering and routing. More formally, a *topic* corresponds to a hierarchical string that can be extended by two wildcards, namely "+" (to replace a single topic level) and "#" (to receive all subtopics from that level on) [5]. An example of this functionality is given in Table 1.

Table 1. Example of topic wildcard usage.

Available Topics	Subscription to Sensor/+/Room1	Subscription to Sensor/House1/#
sensor/		
sensor/house1		sensor/house1
sensor/house1/room1	sensor/house1/room1	sensor/house1/room1
sensor/house1/room2		sensor/house1/room2
sensor/house2		
sensor/house2/room1	sensor/house2/room1	
sensor/house2/room2		

2.1. Threat Model

MQTT as a ligthweight, simple, and generic protocol offers no authentication or payload encryption schemes in its default configuration. This seems like a dire oversight as the support of wildcards creates an inherent flaw: any adversary with an access to the broker can eavesdrop on all data flowing through it. While some issues can be mitigated using Intrusion Detection Systems (IDS), this particular shortcoming shows that MQTT was not designed with security as a primary requirement. Anthraper and Kotak highlighted a series of security issues, summarized here [5]:

- Confidentiality/Privacy: in the absence of proper payload encryption, subscribing to any random topic might give access to sensitive data (such as GPS coordinates).
- Integrity: in default configuration, any adversary intercepting the communication can modify, on the fly, the body of a message and, for example, add some malicious firmware updates. Such messages would then be forwarded to the subscribers.
- Availability: Without additional actions, MQTT is vulnerable to a denial-of-service attack (DoS). In particular, if an attacker succeeds in acquiring the victim's client ID, they can substitute their own connection to the legitimate one and hence starve the victim of all its incoming data.
- Authentication: This functionality is entirely optional in MQTT. An authentication via username and password is available but, as there is no packet encryption, such credentials can be easily intercepted.
- Authorization: MQTT dependence on the WebSocket service makes it inherently vulnerable to malicious script injections.

2.2. Related Work

Considering the above mentioned threats, several MQTT security patches have been developed. First, and only for unconstrained devices, most MQTT implementations now support optional SSL/TLS communications for adequate encryption at the transport level [4]. In addition, to account for lightweight devices, Singh et al. also proposed a secure MQTT (SMQTT) version using *Attribute-Base Encryption* (ABE) over elliptic curves *in lieu* of certificates [6]. Moreover, in 2014, Neisse et al. developed a solution based on their *Model-based Security Toolkit* (SecKit) to enforce compliance with the EU security policy rules and integrated it directly at the MQTT layer [7]. Furthermore, it was proved possible to use the *Password-only Authentication and Key exchange* (PAKE) algorithm to secure all broker-client communications [8]. Finally, the MQTT standard supports the modern authentication protocol *OAuth* [4].

As far as the broker is concerned, it is important to note that most of the solutions cited here above actually rely on the idea of an honest middleman. For example, the SecKit solution must be integrated directly in the broker. On the other hand, the SSL implementation in Mosquitto (an open source MQTT broker) does not offer end-to-end encryption while the PAKE implementation only concerned broker-client sessions: in both cases, the entire packet is readable by the intermediary server [8].

Consequently, in terms of privacy-awareness, we argue that it is reasonable to assume an extension of that model, namely an **honest-but-curious broker**. In other words, the broker is expected to always behave in accordance with the protocol rules, all the while trying to learn as much as possible on the data exchanged. Besides generic privacy leaks, this also brings up another issue, this time related to topics. Indeed, while payloads could easily be ciphered, topics must imperatively remain readable by the broker for efficient filtering and distribution. Nonetheless, topic strings can also be the origin of privacy leaks: for instance, after repetitive observations, a light switch toggle in a given house could be used to infer the user's sleeping habits [8].

For those reasons, this contribution aims to ensure the following points by focusing on the **honest-but-curious broker**'s threats:

- no significant information should be inferred from the topics flowing through the broker;
- the broker should not achieve partner re-identification (i.e., it cannot identify who was communicating and with whom).

Additionally, the privacy-enhanced protocol presented here was designed with constrained devices in mind and each proposed solution shall be discussed accordingly.

3. Communication Architecture and Protocol

The methodology used to develop this protocol amounts to an iterative process. In other words, this section goes from the most privacy-preserving protocol to a more practical one that still presents an acceptable level of privacy-awareness. Our protocol is entirely articulated around the OT primitive which is an application-layer privacy-preserving technique, briefly discussed below.

Oblivious Transfer is an important cryptographic primitive employed in several secure and privacy-preserving protocols in various application domains, such as contract signing or Private Information Retrieval [9]. Similar to zero-knowledge proof (ZKP), it is a 2-party communication scheme, this time between a *sender* S and a *receiver* R. Intuitively, an OT communication can be summarized as follows: the sender holds a secret that the receiver wants to learn and to do so, queries S about several entries. The subtility then lies in that R does not want S to know which data were really targeted (among all queries) or even if the secret did reach its destination. Symmetrically, R should only be able to decipher the data it actually requested.

Rabin first introduced the OT concept by describing a situation where S sends a secret to R such that R has a $1/2$ probability to receive it and S has no knowledge of what actually happened [10]. Then, Even, Goldreich, and Lepel proposed the "*1-out-of-2*" scheme (OT_2^1) in which S sends two 1-bit secrets (m_0, m_1), but only one of them is retrieved by R [11]. This simple idea has then been extended with the notion of 'choice', meaning that R initially communicates an hidden index (a 'choice' between m_0 and m_1) to S and, ultimately, S shall not be able to learn anything about that value [12]. Finally, Brassard, Crépeau, and Robert formalized a generalization called the "*k-out-of-n*" scheme (OT_n^k): S possesses n secrets and R wants to retrieve k of them simultaneously (with $k < n$) [13]. This specific generic protocol is detailed in Figure 1.

Finally, as far as properties are concerned, any correct OT_n^k-implementation ought to satisfy some security requirements such as receiver's privacy, sender's indistinguishability, and sender's accountability [9].

Inputs: System Parameter $SP = Setup(1^{\lambda})$ with λ a security parameter;
Secret set $M = m_1, m_2, ..., m_n$;
Choice set $G = l_1, l_2, ..., l_k$ with $k < n$.

Protocol:
R $\rightarrow$ S: Using SP and G, R outputs (T, sk) with sk a secret key. T includes a token representing the choice set, a proof information Σ (to check if $|G| \leq k$) and a number k. R **sends** T **to** S**.**
S $\rightarrow$ R: Verify if $|G| \leq k$ (Verification Algorithm). If the verification holds, S encrypts the data using SP, T's token and M (Encryption Algorithm).
S **sends the ciphertext set** CT **to** R**.**

Outputs: Using SP, CT, sk, and G, R returns a deciphered message m_i if and only $i \in G$.

Figure 1. The OT_n^k scheme [9].

3.1. Applying the OT Protocol to MQTT

This model, called model_0 and based on the OT_n^k primitive, constitutes the "ideal protocol" and this discussion's starting point. Consequently, to clarify the analogy with the protocol described in Figure 1, let us consider a Publisher (OT's *Sender*) that has the capability to push data to n different topics (or n *secrets*), and several Subscribers (OT's *Receivers*), each interested in k topics (or k *choices*). Additionally, we assume $n > 1$ (each producer has at least two potential sources of data) and $k < n$ (consumers only subscribe to existing topics). The main idea behind this first model is hence that a Subscriber shall receive all possible data but can only decipher the ones they originally chose, thanks to an OT_n^k implementation.

However, implementing the OT_n^k scheme requires at least two communication rounds and a bidirectional communication channel between Publisher and Subscribers. This last point is easily achievable without extensive modification to the MQTT protocol. Indeed, it is possible to imagine a scheme where any new participant Publisher would automatically be subscribed to a reserved "*ClientName/choice*" topic, dynamically creating it if needed (no prior initialization needed [14]). Subsequently, Subscribers would then use that channel to communicate their choice.

Figure 2 provides a high level overview of this first model. While this protocol is quite straightforward, two aspects should be analyzed carefully:

- **Topic formats:** The first thing to ensure here is that the Subscribers do not receive all messages pushed to the "choice" topic (including their own) as it would congest the network unnecessarily. For that reason, the proposed topic format is "*ClientName/choice*" for the Receiver–Sender communication and "*ClientName/data/#*" otherwise. On the one hand, this solution guarantees the separation between both types of messages as the choice topic always corresponds to a higher topic level than any other data. On the other hand, it also protects it from the "+" wildcard as it is never matched by *ClientName/+/...*.
- **Manage data availability:** MQTT is based on the idea that a given data will be pushed as soon as available. However, to work correctly, the OT_n^k scheme requires all n types of data to be sent simultaneously as the Sender does not know what data are required by the other parties. A first solution consists of sending a given piece of data as soon as it is available considering that only the topic-intended Subscribers would try and be able to correctly decrypt the packet anyway. This might cause a privacy leak: the broker can still learn by observing what topic is active at a given moment and at which

frequency (e.g., learn when a light switch is activated). Moreover, if a given Receiver switches off or unsubscribes right after receiving a given message, it is also possible to infer some information on that client's initial choices, thus violating their privacy. Hence, the solution proposed here is to **always publish to all topics by pushing real data on active topics, and random values on the others**.

Finally, based directly on the OT_n^k scheme, this model fully benefits from the OT privacy protection. As far as the honest-but-curious broker is concerned, it cannot know what topics were actually active nor associate a selective set of topics (choices) to any specific Subscriber.

Nevertheless, model_0 is not realistic as it presents a few major drawbacks:

- **Data are sent on all topics at each push:** as mentioned earlier, to guarantee the privacy on active topics and not assuming anything on the different clients' interests, a Publisher must send n messages each time new data are to be pushed. Subsequently, it also means that the broker must redistribute n messages to **all** Subscribers. This might cause a major scalability issue as it would greatly increase the network load.
- **Multiple subscribers with different choices:** this model is once again not easily scalable as each Subscriber requires a different encryption to account for his own private choice (depending of its own k-topic of interest and private key). Indeed, please note that Figure 2 describes a one-to-one situation, meaning that the sending client must repeat this entire protocol with every different token received on the "choice" topic. Similar to the previous drawback, this causes **massive network overload**, especially considering that, while a Publisher must serve all clients' choices, each Subscriber also receives the message encrypted for the others because of the # wildcard, without being able to decipher them correctly. Finally, this phenomenon also highlights a **conceptual contradiction**: while MQTT aims to decouple the client-to-client communication, this protocol bypasses all advantages offered by a broker and actually performs worse than simple one-to-one communications. In particular, the majority of messages received on the other side of the communication are useless and must be discarded.

Publisher associated with *clientExample/data/#*	**One given Subscriber**
Subscribes to *clientExample/choice*.	
	Subscribes to *clientExample/data/#*. Computes its choice token and secret key (T, sk). Publishes its choice on to *clientExample/choice*.
Receives the Subscriber's choices. Encrypt **available** data if active topic. Encrypt random message, otherwise. Publishes each message on its associated topic.	

Figure 2. Model 0—Overview.

3.2. *OT-Inspired MQTT Protocol*

This next idea rests on the plausible assumption that Publisher and Subscribers must have a pre-agreed knowledge of each other. In particular, a Subscriber (namely, an application) commonly requires some basic information (e.g., the position) on a given sensor plugged on to the Publisher to fully ensure its functionalities. This **implies the existence of**

a non-communicating 3rd party involved during pre-deployment, for instance the system or device manager, meaning that necessary assets could be shared before even relying on MQTT communications.

This aspect is crucial in the following model. Indeed, in this proposed solution (model_1.0, refer Figure 3) that directly extends model_0, we assume that all "choice" tokens and private keys in a OT_n^1 have been computed beforehand, mapped to the corresponding topic, and downloaded to all legitimate participants. That being said, please note that, to prevent any Subscribers from having access to data they do not require, only the set of interest mappings (i.e., the one related to the topics they are supposed to subscribe to) is downloaded, whereas the Publisher possesses the whole correspondence table. Moreover, Subscribers' mapping also stores the secret keys necessary for message decryption. In these conditions, the Producer can rightfully use the downloaded information to encrypt all messages (random or not) using the active topic's token. Upon reception, Subscribers automatically discard messages from topics not stored in their lookup table and try to decipher the topics whose mapping they possess. As a matter of fact, this solution is not strictly "OT" as the Publisher has perfect knowledge of what data has been sent. Nonetheless, based on an correct OT protocol, it **guarantees an OT-compliant communication at the broker's scale**: without knowledge of that mapping, an honest-but-curious third-party does not learn which are the active topics and with whom they are shared. Finally, regarding the pre-deployed mapping, it is reasonable to assume that any change in the system could be accounted for by a simple firmware update.

Publisher associated with *clientExample/data/#*	**One given Subscriber**
	Subscribes to *clientExample/data/#*.
Encrypts actual data when they become available using the choice token associated with that topic. Encrypts random value message for inactive topics with the same token. Publishes each message on its corresponding topic.	
	Tries to decrypt each message with the *sk* mapping to the topic on which the message was received.

Figure 3. Model_1.0—Overview.

Figure 4 illustrates this model and **highlights the importance of a Decryption Failure Prevention mechanism.** $Enc(k_i, m_j)$ and $Dec(k_i, m_j)$ represent the encryption and decryption, respectively, of message m_j using secret key k_i. Let us imagine a situation where a Publisher P wishes to publish a message on an active topic, "*.../topic3*". As suggested earlier, P encrypts messages for all possible topics using the token associated with topic3 and sends them to the broker. Let it be two Subscribers $S1$ and $S2$, both interested in distinct subsets of data. Upon reception of the messages distributed by the broker, $S1$ (respectively $S2$) automatically discards all data linked to topics 4 and 5 (resp. 1 and 2). However, regarding the remaining messages, both Subscribers have no knowledge of the key used and, as a first naive implementation, will try to decipher each topic with its expected secret key. Failed occurrence will lead to the discarding of the data. This last point inherently brings forth two issues. Firstly, how can the subscriber detect that a given decryption failed? Secondly, trying to decipher each message causes unnecessary decryption overhead that should be particularly avoided in the context of resource-constrained devices.

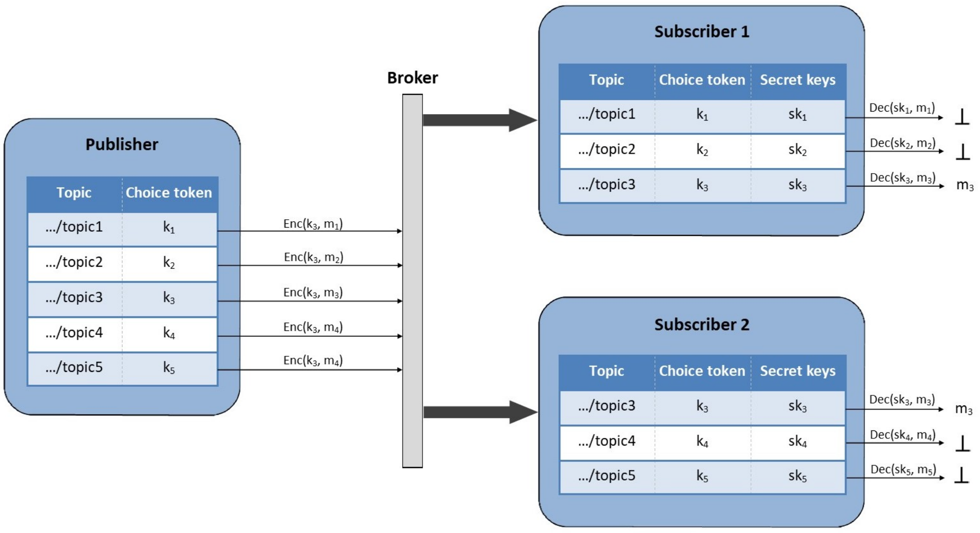


Figure 4. Model_1.0—*.../topic3* is active.

Decryption Failure anticipating mechanism: To account for the two problems cited here above, this model recommends the use of a *Decryption Failure anticipating mechanism*, meaning that a client should be able to detect its inability to decrypt a given message before even trying. The proposed mechanism unfolds as:

- **Sender's side:** upon sending each message encrypted with the key k_i, the Publisher also appends this choice token. **As this asset k_i is inherently protected under the Receiver's Privacy property of OT protocols**, that value does not need to be encrypted and could simply be concatenated to the messages or stored in a specific packet field. Indeed, without the knowledge of the secret keys, no decryption is possible.
- **Receiver's side:** let us imagine a Subscriber interested in topics i, j, and l. In order to detect a would-be random message, it suffices to compare the sent token to the one associated with the expected key k_i, k_j, and k_l. If one of these comparisons hold, the message is decrypted. Otherwise, it is discarded and there is no overhead caused by unnecessary decryption.

To conclude, model_1.0 has the merit to answer the main problems brought up by the previous one:

- **No conceptual contradictions:** model_1.0 preserves the decoupled architecture advocated by the MQTT protocol. In other words, the Publisher sends data without requiring any prior communication rounds with the Receivers. This was made possible under the assumption that all participants share a knowledge of the topic-token mapping.
- **Decreased network load:** subsequently to the previous point, the network load is decreased as the Publisher only needs to publish the encrypted data once, independently of the number of recipients.

Nevertheless, for the same reasons as model_0, it also **guaranties enhanced privacy** and prevents **partners re-identification** by the broker.

3.3. Further Improvements

To achieve further decrease on the network load and encryption overhead, here we present a modified model_1.1, still relying on the publication of various topics to mask the active one but that does not require all possible topics to be emitted. Simultaneously sending a subset of topics, called the *Publication set*, suffices to hide this information. Symmetrically, this model only requires from the Receivers to subscribe to a limited number of "dummy topics" (i.e., topics of no interest) which, together with the legitimate topics, amounts to the *Subscription set*. Additionally, and to preserve this model's original purpose, these sets' construction is also based on a **pseudo-randomized approach**:

- **On the Publisher's side:** contrary to the previous model, a topic activity does not require messages from all possible topics to be emitted. Only a fraction of the available topics are randomly chosen at each round and are needed to mask the active one. Ultimately, this reduces the Publisher's encryption overhead.
- **On the Subscribers' side:** each client is asked to subscribe to a random number of dummy topics (and not to the # wildcard). This aims to confuse the broker as it cannot make the difference between a chosen and unchosen message (thanks to OT_n^k sender's indistinguishability), and hence cannot infer what the topics of interest actually were. However, it is notable that knowing what topics were never of interest (i.e., topics that are never sent to a given client) constitutes in itself a privacy leak. As a first approximation, a possible mitigation could thus be to pseudo-periodically unsubscribe from all topics and recompute a new random "dummy topics" set.

Ultimately, this model reduces the quantity of useless exchanged messages on the network. However, while both points appear satisfying when considering a limited numbers of sending round, it must be noted that the broker has the possibility to observe an arbitrarily large number of communications and might be able to infer some information from "pivot" topics. In particular, it would be able to guess with a reasonable probability of success whether a topic is active and at which frequency (e.g., a given topic is emitted 20 times in a row every 60 rounds), or what are the topics of interest to a given client (those are indeed constant members of the *Subscription set*). As a consequence, the closer the publication/subscription sets' sizes tend to the number of possible topics, the higher the privacy. Additionally, the Decryption Failure anticipating mechanism presented in the previous model also adds upon this issue, as the active topic's choice token is constant across a given round of messages and constitutes a potential statistical vulnerability. The broker could compute a correlation between that value frequency and the sent topics in order to detect the active topic with a reasonable probability. In other words, the inherent pseudo-randomness of this approach renders it **vulnerable to statistical analysis attack**.

3.3.1. Countering Statistical Analysis Attacks

The most important observation made by this improved model, called model_1.2, is that MQTT does not require the topic names to make any sense as long as all legitimate participants can agree on their value. As a matter of fact and as mentioned earlier, topic names are part of an information that should remain private.

Recently, in 2019, Fischer, Kümper, and Tönjes published a topic obfuscation scheme, inspired by the One-Time Password (OTP) method [8]. Let it be a hash function $F : * \rightarrow \{a - z, A - Z, 0 - 9\}^*$, shared between all participants, and a random string, p. In this presented scheme, the Sender owns one list of hash per topic where H_t^i denotes the i-th hash in the list associated with a topic t. The aforementioned list is of size n_t and follows a specific construction: the first element is the hash of s, another random string, whereas $H_t^i = F(H_t^{i-1})\ \forall i > 2$, meaning that all other entries are computed as the hash of their direct predecessor. Upon obfuscation agreement, the sender creates a message M containing p, which will serve as the topics' prefix, as well as the mapping $M : t \rightarrow H_t^{n_t}\ \forall t \in$ Topics, before sending it on a predefined topic. With this information, all receivers then subscribe to */p/#*. That way, if a topic t becomes active, the sender knows that it shall be published on the $/p/H_t^{n_t-1}$ path that $H_t^{n_t}$ must be deleted from the possible mapping and that $n_t \leftarrow n_t - 1$. On the receiving side, computing $F(H_t^{n_t-1})$ and comparing it to the mapping gives the intended topic value. In the absence of viable mapping, the message is discarded. Otherwise, the new mapping value for t is updated to $H_t^{n_t-1}$ and $n_t \leftarrow n_t - 1$. This protocol runs smoothly as long as $n_t > 1$. Passing that limit, a new list must be constructed and shared between all participants.

While this first method successfully obfuscates the topics names, Fischer et al. highlighted the following points [8]:

- **Resource limitation:** in case of memory-constrained devices, it only suffices to store s, p, and n_t to recompute the hash list when needed. However, this requires additional computation power and might not be adapted for battery-powered devices.
- **Vulnerable to data analysis:** the main drawback of this implementation resides in its use of standardized hash functions. Indeed, a data analyst might be able to uncover the pattern and re-associate the exchanged messages.

To counter this last point, Fischer et al. proposed the *Advanced One-Time Password* (AOTP), extending their OTP approach, to achieve **fully randomized obfuscation** [8]. In particular, the use of a random polynomial P at each computation acts as a "salting mechanism", making it harder for a data analyst to try and identify the pattern [8].

Our proposed Model_1.2 draws inspiration from Fischer et al.'s AOTP design and is detailed in Figure 5. In order to construct this model, let it be given a hash function $F : * \to \mathbb{Z}$, a random polynomial P, a set S of random strings s_t (the *seed* going forward). That seed is associated with a counter, called the *seed expiration counter*. The idea behind this protocol thus goes as follows: first, we ensure that both sides of the communication agree on different crucial parameters, namely P and the set S of initial random values that are used to generate the first randomized names. In other words, if we define S_t^i as the string associated with a topic t at the i-th communication round, that first value is computed as $S_t^1 = str(F(s_t))$. For $1 < j \leq$ *seed expiration counter*, it is computed as $S_t^j = str(F(num(S_t^{j-1}).P(j-1)) \; \forall t \in$ *Topics*.

Publisher associated with *clientExample/#*	**One given Subscriber**
	Asks to join the scheme by publishing on *ClientExample/newParticipant*.
	Subscribes to *ClientExample/topic*.
Publishes the seed mapping and P (encrypted) to *ClientExample/topic*.	
Computes for each topic $S_t^1 \leftarrow str(F(init_t * P(1)))$	Computes for each topic $S_t^1 \leftarrow str(F(init_t * P(1)))$
While *seed expiration counter* > 1	
	Subscribes to the all topics in the subscription set using $ClientExample/S_t^j$ for a given t.
Encrypt data when they become available using the choice token associated with that topic.	
Encrypt random value messages for all topics in the Publication set using the same choice token.	
Publish each message on their intended topic in the form $ClientExample/S_t^j$ for a given t.	
Sends synchronizing tick on *ClientExample/topic*.	
	Deciphers messages related to its topic of interests using the associated *sk*.
	Discards messages that could not be deciphered.
Updates a given topic matching with $S_t^{j+1} \leftarrow str(F(num(S_t^j) * P(j)))$.	Updates a given topic matching with $S_t^{j+1} \leftarrow str(F(num(S_t^j) * P(j)))$.
Updates $j \leftarrow j + 1$.	Updates $j \leftarrow j + 1$.
Updates $counter \leftarrow counter - 1$.	

Figure 5. Model_1.2—Overview.

Additionally, please note that MQTT is by nature compatible with a dynamic environment: players can enter and quit the game at any moment. For that reason, a last reserved topic is defined, namely "*ExampleClient/newParticipant*" to which the Publisher must subscribe. Upon entry, new participants should publish to that topic to join the protocol, forcing the reset of all parameters and meaning that all initialization assets must be exchanged again. On top of that, this mechanism **supports an unreliable network** as a reentering client would also publish a request causing said reset. Please note that this topic does not need to be encrypted.

To conclude this model, let us consider the following points:

- **Managing publication and subscription set:** This model requires the subscription set to be constant during a loop (i.e., in the time span between two parameters resets), whereas the publication set can (and should) be recomputed at each send. This is due to the fact that, otherwise, Subscribers would need to compute several hashing values in a row to reach the current obfuscated name of topics it was not previously subscribed to. This constant behavior is a non-issue as the broker is unable to detect that pattern thanks to the obfuscation.
- **Dummy topics are no longer needed:** consequently, to the previous point, one can rightly argue that the subscription set can be reduced to the "topic of interest" set as the aforementioned interest is hidden by the name randomness. This **further reduces the network load**, as a by-product.

3.3.2. Preventing Partners Association

This point will be devoted to what this work calls "interest clusters identification", meaning an ability of the broker to unequivocally determine sets of Publishers sharing a common interest, whatever that interest might be. Indeed, as is, as the same choice token is appended to all the messages from a given sending round to enable Decryption Failure anticipation, the broker can easily compute such set from observing which clients are always interested in messages bearing this specific "mark".

To prevent this re-identification in model_1.2, we thus replace the Decryption Failure anticipating mechanism by a post-decryption attempt detection (i.e., the choice token no longer needs to be appended to the messages), at the cost of power efficiency. Consequently, contrary to that previously suggested, the publication set should here again include random topics to preserve the partners' re-identifications prevention introduced earlier.

To summarize, model_1.2 improves the previous models by obfuscating all topics, meaning that legible names are never shared on the network, thus ensuring that the broker cannot retrieve sensitive information from these values. It also happens to **further decrease the network load** by enabling the restriction of the subscription set to the "topic of interest" set. Finally, it **dynamically supports new client participation**.

Regarding the **statistical analysis attacks** susceptibility brought up by the previous models, two scenarios can be considered. On the one hand, the user can decide that partners re-identification is acceptable, in which case the network load requirement can be further decreased by restricting the publication set to the only active topic. On the other hand, a user demanding anonymous communication might refuse this privacy violation. Consequently, statistical analysis attacks and partner association re-identification are countered at the cost of power efficiency via post-decryption failure detection. Ultimately, with respect to the threat model studied in this work, we strongly advocate for this last solution which is the one considered in the following subsection.

3.3.3. Practical Considerations

To conclude this discussion about OT-inspired communication protocols, we consider a few practical aspects. First, regarding backward compatibility, all the models developed here only need to be implemented on the clients' side, meaning that it is completely backward compatible and can work with any currently existing MQTT broker.

Additionally, a particular attention shall be given to the actual OT implementation. Indeed, in the theoretical protocol developed here above, all messages to be sent are encrypted with the *active choice token*, and a pre-deployed secret key on the Subscribers' side enables decryption. However, please note that this particular generic OT construction is not practical as it requires all the secret messages to belong to G_q, the order-q group on which the scheme is constructed (e.g., in [15]). As this fact implies the existence of an adequate strings-to-group-elements bijection, such construction is not applicable as is in the context of MQTT communications.

The solution to this particular problem can be found in "*The simplest Protocol for OT*", proposed by Chou and Orlandi in 2015 [16]. Inspired by the Diffie–Hellman key-exchange, this scheme uses the OT_n^1 primitive to derive symmetric keys which are then used to encrypt string messages. In essence, this protocol hence generates n keys (one per message type) from the Receiver's choice token. Conversely, from the aforementioned token, the Receiver is only able to construct one symmetric key corresponding to its choice, meaning that it can only decrypt this specific message and no other. Figure 6 illustrates the 1-out-of-n version of this scheme.

```
Setup:
Let it be an additive group $(\mathbb{G}, B, p, +)$ of prime-order $p$
and with B, the base point.
Let it be $H : ((\mathbb{G} \times \mathbb{G}) \times \mathbb{G} \to \{0,1\}^k)$, a hash function.
Sender chooses $y \leftarrow \mathbb{Z}_p$, its private key.
Sender computes $S = yB$ and $T = yS$.
Sender sends $S$ to the Receiver. If $S \notin \mathbb{G}$, abort.

Choose:
For $i$, the index for the chosen message:
    Let it be $c^i$ a vector of indices pointing to $i$.
    Receiver chooses $x^i \in \mathbb{Z}_p$, its choice secret.
    Receiver computes $R^i = c^i S + x^i B$, its choice token.
    Receiver sends $R^i$. If $R^i \notin \mathbb{G}$, abort.

Key Derivation:
For all $j$, the indexes of each message type:
    Sender computes $k_j^i = H_{(S,R^i)}(yR^i - jT)$.
    Receiver computes $k_R^i = H_{(S,R^i)}(x^i S)$.
```

Figure 6. Chou& Orlandi OT_n^1 protocol [16].

Nonetheless, while this protocol has the advantage of supporting string encryption, it significantly differs from the OT model described earlier. In particular, whereas the generic OT protocol advocated by model_1.2 uses a single asset (the choice token) to encrypt all data, this one uses this same asset but to derive n keys. Of these n keys, only the one corresponding to the active topic is ever used, meaning that the remaining $n - 1$ ones can be discarded. In short, a communication requiring n possible topics only needs n keys and not n^2.

This last statement inevitably leads to a discussion about the efficiency of this OT scheme. Indeed, this implementation's key derivation could arguably be based on any other pre-deployed symmetric key scheme as long as this scheme **ensures the topics' indistinguishability**. In other words, a ciphertext from the active should be entirely indistinguishable from any random-message ciphertext. Otherwise, the broker could easily re-identify every interest cluster, which would constitute a privacy breach in the context of an anonymous authentication scheme.

4. Implementation

This section details various implementation choices made in order to test the protocol developed here above.

4.1. Virtual Raspberry Pi Environment

The scope of this work concerns edge-devices in the context of massive IoT. Consequently, the aforementioned design phase aimed to take at each step the least resource-consuming approach as possible while still meeting the target privacy requirements. To that extent, a Raspberry Pi platform has been chosen as a standard device. Raspberry

Pis are single-board desktop computers not bigger than a credit card. Created by the Raspberry Pi Foundation, their price point has always been maintained under $100 to promote and facilitate the access to computing education [17]. Because of its open-source software environment and numerous additional GPIO pins, these boards are often used to control electronic components and, as such, can be indicated for IoT-related application [17]. In particular, records of IoT protocol benchmark on these platforms can be found in the scientific literature (e.g., [8]).

As far as software is concerned, these boards run on *Raspberry PiOS* (previously known as *Raspbian*), a Debian-based operating system that guarantees efficient floating point arithmetic computation and is compatible with Raspberry Pi's ARM CPUs [18]. The proposed protocol has been tested on a full-system emulation using Qemu and a *Raspbian Jessie* image.

4.2. MQTT Implementation

The broker used in this work corresponds to the highly popular *Eclipse Mosquitto Broker* implementation, an open-source broker developed and maintained by the Eclipse Foundation [19]. On the other side of the communication, all clients interact thanks to the *Eclipse Paho MQTT Client* Python library, available on the *Python Package Index* (PyPI) [20].

4.3. OT-Inspired Implementation

In terms of open-source implementations, Chou and Orlandi's 1-out-of-2 scheme has been coded in various programming languages but notably in Python3 [21] by *Nth Party Ltd, Boston, MA, USA*, a company offering privacy-enhancing software. Similar to Chou and Orlandi's original code, this specific library relies on Elliptic Curve signatures, namely the one supported by *Nth Party's Oblivious* library, available via PyPI. In particular, *Oblivious* implements the Ed25519 primitive [22].

The privacy-enhanced protocol developed in this work is based on the OT_n^1-protocol and for that reason, Nth Party's library has been changed according to the generalized protocol proven in [16] and detailed in Figure 6. To summarize, the encryption responsibility of model_1.2 is split between two distinct entities. On the one hand, the Device Manager operates at the highest level and is in charge of deriving the pertinent keys. It does so by virtually recreating Chou & Orlandi's OT_n^1 2-party exchange and storing the useful generated assets to be deployed to each legitimate client.

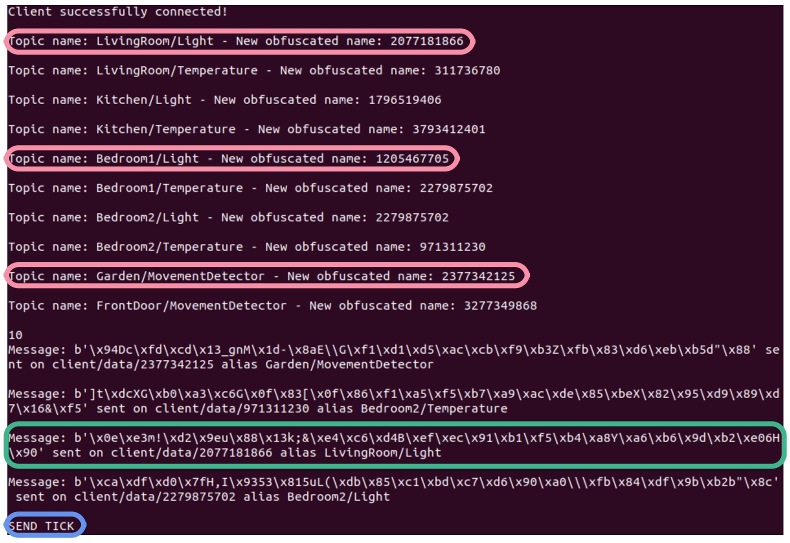
On the other hand, the aforementioned clients are in charge of the actual data encryption and decryption. On top of that, all Publishers must be able to detect Decryption Failure error as an anticipating mechanism would compromise the partners re-identification prevention (see Section 3.3.2). Both functionalities were here achieved thanks to *PyNaCl's SecretBox* object. Indeed, this symmetric key-based structure generates ciphertexts containing a 16-byte authenticator which can be used at decryption to raise an exception in the eventuality of Decryption Failure due to an invalid key [23].

4.4. Topic Obfuscation Implementation

As far as the actual operation of computing the obfuscated names is concerned, this work uses the hash function and random polynomial as implemented by Fischer et al. [8,24] (see Section 3.3.1). Note that the topic obfuscation scheme only works as long as all participants are synchronized and agree on which topic names data will be sent. Indeed, by definition of the reduced *subscription set*, some subscribers might not receive any message at a given communication round, which would impede this synchronization and destroy the mapping between Sender and Receivers. This is mitigated by sending a tick sent on "*client/topic*", but this message might arrive before the Publisher finishes processing all the previous rounds' messages. Additionally, a given message might arrive on a topic whose obfuscated name has not been yet computed by some participants. All in all, this causes an asynchronous behavior between clients, ultimately leading to message loss. This issue has thus been resolved by combining two functionalities. On the one hand, all messages sent by the Publisher make use of an MQTT native retaining feature via the *retain flag*, which tells the broker to store this specific message, its corresponding topic, and its QoS [25]. Ultimately, this means that the aforementioned message is to be sent to any client subscribing to this topic at any point in the future. On the other hand, as Subscribers might receive a tick and compute the new names before they have received messages from the previous round, these clients actually keep in their memories not only the current obfuscated value but all the previous ones too. This is necessary in order for the client to know the key they should use to try and decrypt an incoming message. This mapping is erased as soon as the obfuscation scheme is reset, whether when a new participant joins the communication or when the *obfuscation seed* expires (see Section 3.3.1). Finally, to ensure synchronization, all obfuscation parameters or ticks are published under QoS2 as it is imperative that they are received but without duplicates.

4.5. Results-Traffic Observation

First, let us consider the messages sent/received by clients on both side of the communication. Figures 7 and 8 respectively illustrated the Publisher *P* and a Subscriber *S* points of view. They also highlight in pink the topics belonging to S's topic-of-interest set; in green, the active topic; in red, an obfuscation reset; and in blue, an obfuscation synchronizing tick.

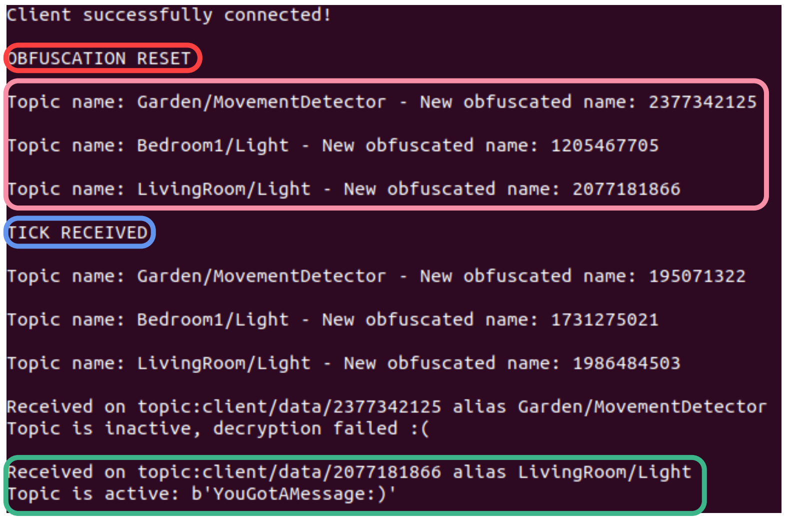


```
Client successfully connected!

Topic name: LivingRoom/Light - New obfuscated name: 2077181866

Topic name: LivingRoom/Temperature - New obfuscated name: 311736780

Topic name: Kitchen/Light - New obfuscated name: 1796519406

Topic name: Kitchen/Temperature - New obfuscated name: 3793412401

Topic name: Bedroom1/Light - New obfuscated name: 1205467705

Topic name: Bedroom1/Temperature - New obfuscated name: 2279875702

Topic name: Bedroom2/Light - New obfuscated name: 2279875702

Topic name: Bedroom2/Temperature - New obfuscated name: 971311230

Topic name: Garden/MovementDetector - New obfuscated name: 2377342125

Topic name: FrontDoor/MovementDetector - New obfuscated name: 3277349868

10
Message: b'\x94Dc\xfd\xcd\x13_gnM\x1d-\x8aE\\G\xf1\xd1\xd5\xac\xcb\xf9\xb3Z\xfb\x83\xd6\xeb\xb5d"\x88' se
nt on client/data/2377342125 alias Garden/MovementDetector

Message: b']t\xdcXG\xb0\xa3\xc6G\x0f\x83[\x0f\x86\xf1\xa5\xf5\xb7\xa9\xac\xde\x85\xbeX\x82\x95\xd9\x89\xd
7\x16&\xf5' sent on client/data/971311230 alias Bedroom2/Temperature

Message: b'\x0e\xe3m!\xd2\x9eu\x88\x13k;&\xe4\xc6\xd4B\xef\xec\x91\xb1\xf5\xb4\xa8Y\xa6\xb6\x9d\xb2\xe06H
\x90' sent on client/data/2077181866 alias LivingRoom/Light

Message: b'\xca\xdf\xd0\x7fH,I\x9353\x815uL(\xdb\x85\xc1\xbd\xc7\xd6\x90\xa0\\\xfb\x84\xdf\x9b\xb2b"\x8c'
 sent on client/data/2279875702 alias Bedroom2/Light

SEND TICK
```

Figure 7. Publisher's View.

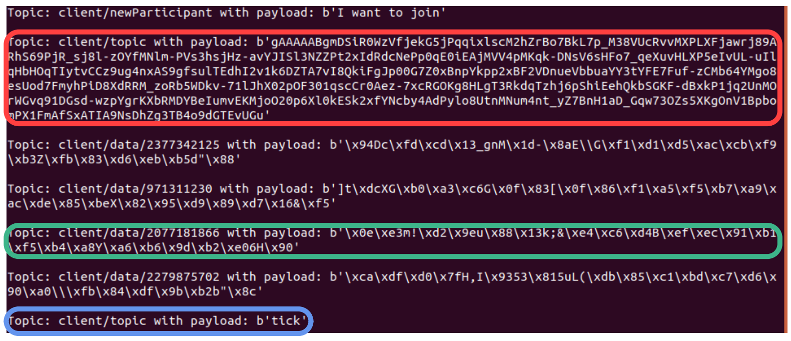
Furthermore, two observations are worth noting in Figure 8. On the one hand, as planned, this figure shows that the Subscriber was indeed only able to decrypt the message related to the active "*client/LivingRoom/Light*" topic, whereas its secretBox implementation raised a Description Failure for the inactive "*client/Garden/MovementDetector*". On the other hand, this particular example illustrates a case where messages related to the first obfuscation arrive after the synchronizing tick and the next name computation. As expected and by keeping in their memory all previous name mappings, S was still able to correctly process the messages that arrived with a delay.

Finally, Figure 9 shows what a 3rd party observer (e.g., the broker) would intercept from that communication. In particular, please note that, on the one hand, the obfuscation parameters' agreement is indeed encrypted and, on the other hand, thanks to the OT encryption, it is indeed also not possible either for the server to identify the active topic (see green box) from all other messages.



```
Client successfully connected!

OBFUSCATION RESET

Topic name: Garden/MovementDetector - New obfuscated name: 2377342125

Topic name: Bedroom1/Light - New obfuscated name: 1205467705

Topic name: LivingRoom/Light - New obfuscated name: 2077181866

TICK RECEIVED

Topic name: Garden/MovementDetector - New obfuscated name: 195071322

Topic name: Bedroom1/Light - New obfuscated name: 1731275021

Topic name: LivingRoom/Light - New obfuscated name: 1986484503

Received on topic:client/data/2377342125 alias Garden/MovementDetector
Topic is inactive, decryption failed :(

Received on topic:client/data/2077181866 alias LivingRoom/Light
Topic is active: b'YouGotAMessage:)'
```

Figure 8. Subscriber's view.



```
Topic: client/newParticipant with payload: b'I want to join'

Topic: client/topic with payload: b'gAAAAABgmDSiR0WzVfjekG5jPqqixlscM2hZrBo7BkL7p_M38VUcRvvMXPLXFjawrj89A
RhS69PjR_sj8l-zOYfMNlm-PVs3hsjHz-avYJISl3NZZPt2xIdRdcNePp0qE0iEAjMVV4pMKqk-DNsV6sHFo7_qeXuvHLXP5eIvUL-uIl
qHbHOqTIytvCCz9ug4nxAS9gfsulTEdhI2v1k6DZTA7vI8QkiFgJp00G7Z0xBnpYkpp2xBF2VDnueVbbuaYY3tYFE7Fuf-zCMb64YMgo8
esUod7FmyhPiD8XdRRM_zoRb5WDkv-71lJhX02pOF301qscCr0Aez-7xcRGOKg8HLgT3RkdqTzhj6pShiEehQkbSGKF-dBxkP1jq2UnMO
rWGvq91DGsd-wzpYgrKXbRMDYBeIumvEKMjoO20p6Xl0kESk2xfYNcby4AdPylo8UtnMNum4nt_yZ7BnH1aD_Gqw73OZs5XKgOnV1Bpbo
mPX1FmAfSxATIA9NsDhZg3TB4o9dGTEvUGu'

Topic: client/data/2377342125 with payload: b'\x94Dc\xfd\xcd\x13_gnM\x1d-\x8aE\\G\xf1\xd1\xd5\xac\xcb\xf9
\xb3Z\xfb\x83\xd6\xeb\xb5d"\x88'

Topic: client/data/971311230 with payload: b']t\xdcXG\xb0\xa3\xc6G\x0f\x83[\x0f\x86\xf1\xa5\xf5\xb7\xa9\x
ac\xde\x85\xbeX\x82\x95\xd9\x89\xd7\x16&\xf5'

Topic: client/data/2077181866 with payload: b'\x0e\xe3m!\xd2\x9eu\x88\x13k;&\xe4\xc6\xd4B\xef\xec\x91\xb1
\xf5\xb4\xa8Y\xa6\xb6\x9d\xb2\xe06H\x90'

Topic: client/data/2279875702 with payload: b'\xca\xdf\xd0\x7fH,I\x9353\x815uL(\xdb\x85\xc1\xbd\xc7\xd6\x
90\xa0\\\xfb\x84\xdf\x9b\xb2b"\x8c'

Topic: client/topic with payload: b'tick'
```

Figure 9. Broker's view.

5. Conclusions

This paper presents a privacy-preserving MQTT protocol in an honest-but-curious threat model. Derived from the OT concept, the proposed model ensures confidentiality as the messages are fully encrypted, and the server cannot learn anything from topic names due to an obfuscation mechanism following a pattern known only by the intended end-clients. Additionally, a declination of this model prevents partners' re-identification, meaning that an observer cannot identify any set of clients sharing a common interest thanks to OT's indistinguishability but does so at the cost of efficiency. Indeed, implementing a Decryption Failure anticipating mechanism to save computational power is not compatible with this approach as appending the required information for this to work constitutes a statistical pivot that can be exploited by the broker. The reflection developed here clearly illustrates the subtle balance to be found between resource efficiency and the degree of privacy one wants to achieve.

For the future work, this implementation's key derivation could arguably be considered on any pre-deployed symmetric key scheme other than OT, as long as it ensures the topics' indistinguishability. In other words, a ciphertext from the active topic should be entirely indistinguishable from any random-message ciphertext. Otherwise, the broker could easily re-identify every interest clusters which would constitute a privacy breach in the context of an anonymous communication system. We thus suggest conducting further investigations on this aspect to potentially reduce the conceptual complexity of the proposed protocol.

Author Contributions: Conceptualization, A.H.; Supervision, G.S. and J.-M.D.; Writing—original draft, A.H.; Writing, review and editing, G.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

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

References

1. Li, C.; Palanisamy, B. Privacy in Internet of Things: From Principles to Technologies. *IEEE Internet Things J.* **2019**, *6*, 488–505. [CrossRef]
2. Fizza, K.; Banerjee, A.; Mitra, K.; Jayaraman, P.P.; Ranjan, R.; Patel, P.; Georgakopoulos, D. QoE in IoT: A vision, survey and future directions. *Discov. Internet Things* **2021**, *1*, 4. [CrossRef]
3. PR Newswire. IEEE Internet of Things Survey Provides Clarity Around Definition, Future Uses and Challenges. Available online: https://www.prnewswire.com/news-releases/ieee-internet-of-things-survey-provides-clarity-around-definition-future-uses-and-challenges-193865271.html (accessed on 21 May 2021).
4. MQTT.org. MQTT: The Standard for IoT Messaging. Available online: https://mqtt.org/ (accessed on 21 May 2021).
5. Anthraper, J.J.; Kotak, J. Security, privacy and forensic concern of MQTT protocol. In Proceedings of the International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Jaipur, India, 26–28 February 2019; Amity University Rajasthan: Jaipur, India, 2019. [CrossRef]
6. Singh, M.; Rajan, M.; Shivraj, V.; Balamuralidhar, P. Secure MQTT for Internet of Things (IoT). In Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, India, 4–6 April 2015; pp. 746–751. [CrossRef]
7. Neisse, R.; Steri, G.; Baldini, G. Enforcement of security policy rules for the Internet of Things. In Proceedings of the 2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Larnaca, Cyprus, 8–10 October 2014; pp. 165–172. [CrossRef]
8. Fischer, M.; Kümper, D.; Tönjes, R. Towards improving the Privacy in the MQTT Protocol. In Proceedings of the 2019 Global IoT Summit (GIoTS), Aarhus, Denmark, 17–21 June 2019; pp. 1–6. [CrossRef]
9. Lai, J.; Mu, Y.; Guo, F.; Chen, R.; Ma, S. Efficient k-out-of-n oblivious transfer scheme with the ideal communication cost. *Theor. Comput. Sci.* **2018**, *714*, 15–26. [CrossRef]
10. Rabin, M.O. How To Exchange Secrets with Oblivious Transfer. In *Technical ReportTR-81*; Aiken Computation Lab, Harvard University: Cambridge, MA, USA, 1981.
11. Even, S.; Goldreich, O.; Lempel, A. A Randomized Protocol for Signing Contracts. *Commun. ACM* **1985**, *28*, 637–647. [CrossRef]
12. Wang, X.; Li, Z. Research on the security Oblivious Transfer protocol based on ECDDH. *J. Phys. Conf. Ser.* **2020**, *1549*, 032152. [CrossRef]
13. Brassard, G.; Crépeau, C.; Robert, J.M. All-or-nothing disclosure of secrets. In *Conference on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1986; pp. 234–238.
14. HiveMQ. MQTT Topics & Best Practices–MQTT Essentials: Part 5. Available online: https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/ (accessed on 22 May 2021).
15. Tzeng, W.G. Efficient 1-out-of-n oblivious transfer schemes with universally usable parameters. *IEEE Trans. Comput.* **2004**, *53*, 232–240. [CrossRef]
16. Chou, T.; Orlandi, C. The Simplest Protocol for Oblivious Transfer. In *Progress in Cryptology—LATINCRYPT*; Lecture Notes in Computer Science; Lauter, K., Rodríguez-Henríquez, F., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; Volume 9230, pp. 40–58. [CrossRef]
17. Opensource.com. What Is a Raspberry Pi? Available online: https://opensource.com/resources/raspberry-pi (accessed on 22 May 2021).
18. Raspbian.com. About Raspbian. Available online: https://www.raspbian.org/RaspbianAbout (accessed on 22 May 2021).
19. Eclipse Foundation, Cedalo, Moquitto. Eclipse Moquitto—An Open Source MQTT Broker. Available online: https://mosquitto.org/ (accessed on 22 May 2021).
20. Pypi.org. paho-mqtt 1.5.1. Available online: https://pypi.org/project/paho-mqtt/ (accessed on 22 May 2021).
21. Nth Party Ltd. Github-otc. Available online: https://github.com/nthparty/otc (accessed on 22 May 2021).
22. Nth Party Ltd. Github-Oblivious. Available online: https://github.com/nthparty/oblivious (accessed on 22 May 2021).
23. Pynacl.readthedocs.io. Secret Key Encryption. Available online: https://pynacl.readthedocs.io/en/latest/secret/ (accessed on 22 May 2021).
24. Mafi-mcfly. Github-aotp-mqtt. Available online: https://github.com/mafi-mcfly/aotp-mqtt (accessed on 22 May 2021).
25. HiveMQ. Retained Messages—MQTT Essentials: Part 8 Author Portrait. Available online: https://www.hivemq.com/blog/mqtt-essentials-part-8-retained-messages/ (accessed on 22 May 2021).