A Secure Publish/Subscribe Protocol for Internet of Things

Lukas Malina[1*], Gautam Srivastava[2], Petr Dzurenda[1], Jan Hajny[1] and Radek Fujdiak[1]

[1] Brno University of Technology, Brno, Czech Republic malina@feec.vutbr.cz, dzurenda@feec.vutbr.cz, hajny@feec.vutbr.cz, fujdiak@feec.vutbr.cz
[2] Brandon University, Brandon, Canada srivastavag@brandonu.ca

Abstract. The basic concept behind the emergence of Internet of Things (IoT) is to connect as many objects to the Internet as possible in an attempt to make our lives better in some way. However, connecting everyday objects like your car or house to the Internet can open up major security concerns. In this paper, we present a novel security framework for the Message Queue Transport Telemetry (MQTT) protocol based on publish/subscribe messages in order to enhance secure and privacy-friendly Internet of Things services. MQTT has burst onto the IoT scene in recent years due to its lightweight design and ease of use implementation necessary for IoT. Our proposed solution provides 3 security levels. The first security level suits for lightweight data exchanges of non-tampered messages. The second security level enhances the privacy protection of data sources and data receivers. The third security level offers robust long-term security with mutual authentication for all parties. The security framework is based on light cryptographic schemes in order to be suitable for constrained and small devices that are widely used in various IoT use cases. Moreover, our solution is tailored to MQTT without using additional security overhead.

Keywords: MQTT, Security, Cryptography, IoT, Digital Signature, Privacy

1 Introduction

In today's digital world, the growing need to address our security concerns for personal data grows. It does not seem to matter what type of transaction is occurring or what type of technology is being used, security is always involved. A recent push of Internet of Things (IoT) has changed the landscape of data sharing. One very well known IoT protocol is Message Queue Transport Telemetry (MQTT). MQTT came into the limelight for its use with Facebook Messenger and its direct links to IoT [4]. What makes MQTT popular is its light footprint on both energy and computation [3]. Where it lacks however is in robust security. Although there is some level of security available for MQTT, it does not meet the high threshold needed for the future of IoT. Moreover, the security available does not have the efficiency in both energy and computation that is required.

There is no reason to even debate the need of security for transport protocols. However, using well known security protocols in an IoT setting has new challenges and hurdles. Often in IoT, security is seen as a trade-off with quality of service (QoS) pending on the level of service an implementation wishes to maintain. Many secure cryptographic algorithms require much more demanding resource-use than standalone IoT devices can provide Therefore, security of these devices is often at a compromised level. Another major

*The final publication appears in proceedings of ARES 2019. DOI:10.1145/3339252.3340503

challenge is the ability to update IoT devices in the field. Critical security based issues that would need to be applied simultaneously to all devices on the network can be hindered by unreliable networks which many IoT devices run on. In this paper, we present a secure version of MQTT that maintains its low energy and computational footprint while also providing robust security at a level suitable for the IoT age. Although we do not discuss the MQTT protocol in depth in this work, we recommend interested readers to the MQTT documentation in [25, 24] and to a survey work like [23].

The rest of the paper is organized as follows. We introduce some background and motivation for studying MQTT in 2 and finish the section off with a discussion in the available security in current releases of MQTT. Next, 3 tackles all the related work on the subjects touched on in this paper. We start our main results in 4 with an in depth description of our enhanced version of MQTT. We then give a thorough performance evaluation in 5 and then end with concluding remarks in 6.

2 Background and Motivation

Created in 1999, MQTT was invented as a joint collaboration between **IBM** and **Arcom** (now **Cirrus Link**). Initial design parameters included minimal battery loss and minimal bandwidth use to connect oil pipelines over satellite connections [26]. In [26], the following characteristics were specified:

- Simple to implement
- Provide Quality of Service Data Delivery
- Lightweight and Bandwidth Efficient
- Data Agnostic
- Continuous Session Awareness

MQTT uses a client-broker publish/subscribe implementation. A main hub is used to control the messages known as the message broker between publishers and subscribers as shown in Figure 1.

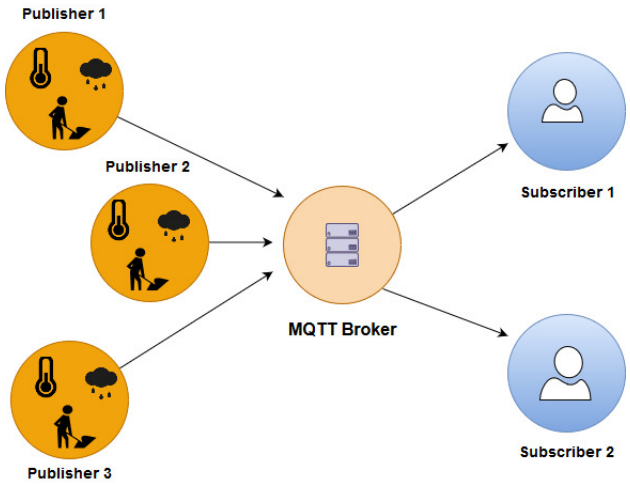

Figure 1: MQTT Schematic

We have also seen many current and important use cases where robust security of the MQTT protocol would be paramount.

- **Facebook Messenger**: With over one billion users worldwide, the security implications of MQTT are critical.
- **Conoco Phillips Pipeline**: 30,000 devices on 17,000 km of pipeline monitored and controlled via MQTT on satellite links.
- **St. Jude Medical**: There are over 100,000 pacemakers monitored via MQTT creating the best use case of this section dealing with a true IoT situation.
- **Toshiba Consert**: another pure IoT use case, running a smart energy grid solution built on the backbone of MQTT
- **Sprint Velocity**: Connected Car and Telematics provider.

2.1 Approaches to MQTT Security

When looking at available MQTT security, we can divide the resources into multiple layers. Each layer can be used to prevent different types of attacks. The main goal of MQTT is to be lightweight and easy to use for those trying to implement an IoT communication network. The built-in security in MQTT is just in essence a collection of security standards on different layers like SSL/TLS for transport security. Security is challenging so it is no surprise MQTT just used some generally acceptable standards for its release. Those standards however lack the lightweight characteristic that would be essential for an IoT MQTT implementation.

Next we will give a brief overview of the different types of security currently available in MQTT and will also look at some shortcomings we wish to provide solutions to in 4 and 5.

2.1.1 Authentication

MQTT provides the ability to provide a username/password field in the `CONNECT` packet for authentication. This is an optional feature. The username is an `UTF-8` encoded string whereas the password can be of maximum length 65535 bytes. There are no requirements on the password in the latest release of MQTT, however you are unable to send a password alone without a username.

In a more advanced setting, every MQTT client has a unique client identifier. We can use this client ID with maximum size of 65535 characters (each character 1 byte) in conjunction with a username and password to authenticate the client. We can also use a `X.509` client certificate for authentication which will be discussed in 2.1.4.

2.1.2 Authorization

To restrict a client from publishing and/or subscribing to authorized topics, authorization is controlled on the broker side. MQTT currently has the ability to have the following permissions controlled by the broker

- Allowed Topic (specific topic, all topics)
- Allowed Operation (publish, subscribe, both)
- Allowed QoS (0,1,2,all)

By prefixing information to the client ID, the topics can be controlled easily (`clientID/weather` or `clientID/#`). Furthermore, since in most cases the Broker is not considered to be either energy or computationally challenged, authorization can easily be implemented to the needs of the specific use cases.

2.1.3 TLS/SSL

By default, MQTT relies on the TCP transport protocol which does not use encrypted communication. To allow any encryption many brokers in use today will use TLS in place of plain TCP. We find the use of TLS not feasible for constrained devices as is seen in most IoT scenarios [9]. TLS both is very computationally heavy and can require high memory usage as well.

2.1.4 X.509 Client Certificate Authentication

For added security in MQTT, `X.509` client certificates are recommended to be used. Only valid clients are allowed to set up secured connections. However, there is an added cost with this. The added layer of security would require a MQTT client provisioning for this and the ability for client certificate revocation mechanisms. Moreover, in the few publisher to many client scenario, the ability to monitor the life-cycle of client certificates has added challenges for constrained IoT subscriber devices.

2.1.5 OAuth 2.0

`OAuth` is an open protocol to allow secure authorizations and is usable from mobile, desktop, and web based applications. The `OAuth` 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service [8]. `OAuth` was clearly designed for HTTP. This fact automatically leaves MQTT out of scope for it. Constrained to IoT frameworks where known issues with `OAuth` often require human intervention, `OAuth` and IoT just do not make sense in the same sentence.

2.1.6 Payload Encryption

With payload encryption, in MQTT there is the option to encrypt the payload as part of `PUBLISH` packets. The payload would always be encrypted on the Publisher side, whereas the decryption process may occur in the subscriber in End-to-End encryption but also can be accomplished at the Broker. In either setup, Asymmetric or Symmetric encryption schemes could potentially be established. Moreover, just the payload data would be encrypted in either scenario and all other data (client info, certificates, topic info) would remain un-encrypted. Pending on which encryption/decryption scheme is used, the scheme itself can be very resource intensive causing issues for constrained devices again. Also there would need to be secure provisioning of the keys to all of the MQTT clients. Finally, these solutions do not prevent man-in-the-middle or replay attacks.

2.1.7 Message Data Integrity

Message data integrity can be added to the payload of `PUBLISH` packets using any of digital signatures/MAC/checksum type solutions. Alongside a strong encryption framework, data integrity solutions do not add too much extra additional security. However, data integrity checks are a good addition to message encryption. Even if the attacker can decrypt the message (and re-encrypt it), the integrity check would still fail if the message was altered.

3 Related Work

Nowadays, many comprehensive studies dealing with security in IoT have been published [17, 21, 12]. Further, surveys such as [14] and [15] provide overviews of privacy issues and challenges related to IoT technologies. Recently, Malik *et al.* [13] analyze key bootstrapping protocols based on public key cryptography in IoT. This survey discusses

several authenticated key delivery approaches such as raw public key, certificated based, identity based, certificate-less and so on.

In general, there are several papers that study security in IoT using the MQTT protocol. Ramos *et al.* [10] propose a security tests based on modern fuzzing in the MQTT based solutions. This work focuses on testing and does not propose security features such as data confidentiality, authenticity and user authentication. Amaran *et al.* [2] study various lightweight ciphers employed in MQTT. The work focuses on AES, LBLOCK, PRESENT and KLEIN ciphers. Their benchmarks show that LBLOCK is an efficient alternative to AES. The paper does not consider security aspects of chosen ciphers and their modes. In [11], the authors study various encryption settings in MQTT. Their tests cover data encryption in three cipher modes (ECB, CBC and OCB) and Link Layer security (LLsec) that is using `AES-CCM`. They also discuss the suitability of different security options for MQTT-enabled use cases, namely, wind turbine sensors and data networks. Both works do not focus on key bootstrapping. Esfahani *et al.* [6] deal with lightweight authentication mechanisms based on hash and XOR operations that are suitable for M2M communications in the Industrial IoT environment. Their solution does not use expensive operations and is based on preshared keys between sensors and servers (routers) that employ TPM for storing secret keys. Singh *et al.* [22] propose a secure version of MQTT and MQTT-SN (for sensor networks). They use Key/Ciphertext Policy-Attribute Based Encryption (KP/CP-ABE) based on Elliptic Curve Cryptography. CP/KP-ABE schemes are based on bilinear pairing operations. These operations are usually computationally expensive and those pairing-based schemes are not suitable in IoT networks employing constrained devices.

Furthermore, there are few papers that focus solely on key establishment and authentication between entities in IoT using MQTT. Shin *et al.* [19] propose a simple security framework for MQTT that avoids using certificates. Their proposed solution incorporates the augmented Password-based Key Exchange (AugPAKE) protocol. The AugPAKE protocol is based on the modification of the Diffie Helman key exchange protocol. In this protocol, the client uses his/her password in order to authenticate to the server (a verifier) and the verifier does not need to store client passwords but only a public verifier created from the passwords. After successful authentication and key agreement between the client and the server, the secure session is established. During the key establishment process in the Shin proposal [19], the client has to compute 2 exponentiation (or point multiplication), one modular division, one modular multiplication, one addition and 4 hash functions. The server computes 3 exponentiation (or point multiplication), one modular multiplication, and 4 hash functions. The AugPAKE protocol exchanges 4 messages.

Calabretta *et al.* [5] propose a security solution for MQTT based on AugPAKE protocol that is similar to a framework in [19]. Their solution adds an authentication token and an authorization token that are used to check the message authenticity at the broker side. Authorization tokens are communicated via a secondary secure channel, e.g. visual texts on displays. Haase and Labrique [7] propose another security solution based on the asymmetric (augmented) PAKE protocol. Their solution uses a verifier-based password-authenticated key-exchange protocol that is tailored for Industrial IoT (IIoT). The paper reviews state of the art PAKE protocols and provides an appropriate solution for authenticated key establishment in IIoT with constrained devices. Their proposed AuCPace protocol performs 8 messages (4 each direction). The client needs 3 exponentiation (or point multiplication), one `PBKDF` function and 6 hash functions. The server takes 4 exponentiation (or point multiplication) and 6 hash functions. These solutions usually do not support group encryption.

In our paper, we propose a novel security framework which supports 3 security levels for MQTT services. The protocol is based on secure and efficient cryptographic schemes that ensure authenticated key establishment and short digital signatures and is novel in

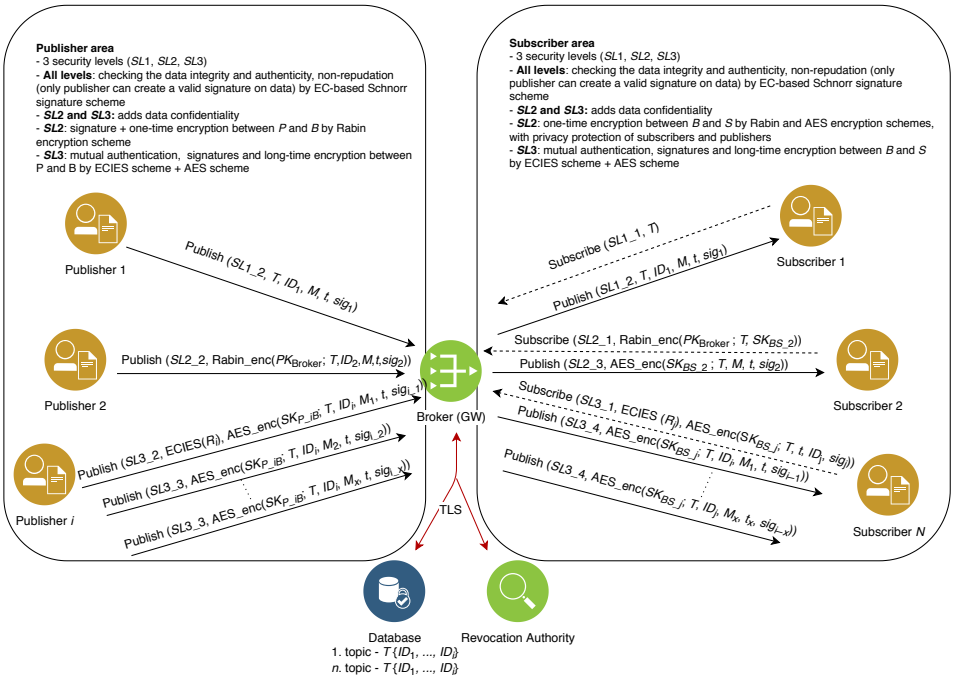

Figure 2: Basic principle of novel security framework for MQTT.

both approach and results.

4 Secure Pub/Sub Protocol

In this section, we propose a novel security framework for MQTT services. The following subsections presents notation and cryptography used, high level description and concrete proposal's phases.

4.1 Notation and Cryptography Used

The solution is based on secure cryptographic schemes and primitives such as Elliptic Curve Integrated Encryption Scheme (`ECIES`) [1, 20], a modified elliptic curve variant of the Schnorr digital signature scheme [18], the Rabin encryption cryptosystem [16] and Advanced Encryption Scheme (AES) in an authenticated mode (e.g. GCM). These schemes are chosen based on their efficiency and security. Used notation in our proposal is defined in Table 1.

4.2 High-level Overview

The basic principle of our framework is shown in Figure 2. The communication model consists of four parties: Publisher (P), Broker (B), Subscriber (S) and Revocation Authority (RA) as the third trusted party. Publishers are usually constrained devices that sense data and produce messages that are sorted by topics in the system. These messages are sent to brokers who forward them to end users (subscribers). B is represented by a central gateway which controls and routes messages by their topics in the system. S represents an end user who processes and uses messages from P. We divide the topology into 2 areas: a publisher area and a subscriber area. The proposed protocol provides 3 Security Levels ($SL1$, $SL2$, $SL3$) in both areas. All security levels ensure the data

Table 1: Used notation.

Notation	Definition
a	private key for the Schnorr signature scheme
AES_enc(SK; $plaintext$)	AES encryption in GCM mode
AES_dec(SK; $ciphertext$)	AES decryption in GCM mode
B	Broker
c	ciphertext for the Rabin decryption ($<n$)
$ciphertext$	general ciphertext for decryption
d	random number
$E(\mathbb{F}_q)$	elliptic curve over finite field $\mathbb{F}_q$
`ECIES`(R_i)	authenticated key establishment `ECIES` scheme
G	point generator of elliptic curve
H()	hash function, e.g. SHA3-256
ID	ID of entity (a Schnorr public key)
k	private key for the `ECIES` scheme
K	public key for the `ECIES` scheme
KDF()	Key Derivation Function
$plaintext$	plaintext for encryption
M	concrete message in string
n	modulus for the Rabin encryption scheme
P	Publisher
PK	public key (n) for the Rabin encryption scheme
Q	EC point
q	modulus for the Schnorr signature scheme
r, s	prime numbers for the Rabin encryption scheme that satisfying $s \equiv 3 \pmod 4$
Rabin_enc(PK; $plaintext$)	Rabin encryption
Rabin_dec($privkey$; $ciphertext$)	Rabin encryption
RA	Revocation Authority
S	Subscriber
sig	EC-based Schnorr signature (z, e)
$SL1$	denotation for the 1. security level
$SL2$	denotation for the 2. security level
$SL3$	denotation for the 3. security level
SK	secret key for symmetric encryption
T	topic in string
t	timestamp
w	encoded $plaintext$ into an integer ($<n$)
$\mathbb{Z}$	ring of integers
$\times$	elliptic curve scalar (point) multiplication
$\in_R$	randomly chosen in ... (e.g., in finite field)
$\cdot$	field multiplication

integrity, data authenticity, and non-repudiation (only the publisher can create signature and original data) by employing the signature scheme, namely, the elliptic curve based Schnorr scheme. In the publisher area, $SL2$ and $SL3$ add data confidentiality. Only B is able to decrypt the data from publishers. $SL2$ provides one-time encryption between P and B by the asymmetric Rabin encryption scheme. $SL3$ provides long-time encryption between P and B by the authenticated key establishment `ECIES` scheme, and standard symmetric cipher (AES) with the secure authenticated cipher mode, e.g., `AES-GCM`.

In the subscriber area, security levels ($SL1$, $SL3$) check the data integrity and authenticity, and non-repudiation by using the EC-based Schnorr scheme. In $SL1$, S verifies the signature on data that could be created only by P who knows a private key. $SL2$ and $SL3$ provides data confidentiality in both directions by data encryption. $SL2$ uses one-time encryption between B and S by the Rabin and AES encryption schemes and provides privacy protection. In $SL2$, S can be anonymous (e.g. by onion routing) because B does not check his/her identification. Because of publishers' public keys (IDs) are removed by the broker, subscribers are not able to decipher who creates published messages. Only B and RA, who have access to database with topics and publisher public keys, are able to link publisher's identity with the signature on data. Hence, it is assumed that B is a semi-honest party which follows the protocol. $SL3$ provides a long-time security session between B and S by `ECIES` and AES schemes (`AES-GCM`). Nevertheless, only B learns what subscribers receive. This level supports access control and mutual authentication. B checks the subscribers' identity by Schnorr signatures.

The proposed solution contains these phases: Setup Phase, Join Phase, Communication Phase - Security Level 1, Communication Phase - Security Level 2, Communication Phase - Security Level 3, and Revocation Phase.

4.3 Setup Phase

In this phase, cryptographic parameters are set. The length of parameters should be in line with keylength recommendations defined by `NIST`, `ANSSI`, `BSI`, and others, i.e., prime lengths 2048 `b`, modulus lengths 4096 `b`, ECC size 256 `b`, symmetric encryption key sizes 128 `b`. Firstly, the broker (B) generates Rabin cryptosystem parameters. B randomly generates primes r, s and stores these values as his/her private key ($privkey_{Broker}$), and then, B computes his/her public key (PK_{Broker}) where

$$PK_{Broker} = n = r \cdot s. \tag{1}$$

Furthermore, B sets the elliptic curve domain parameters (e.g. $E(\mathbb{F}_q)$, q, G, ...) for the `ECIES` and the Schnorr scheme. B generates randomly his/her private `ECIES` key where $k_{Broker} \in_R \mathbb{Z}_q$ and computes his/her private `ECIES` key where

$$K_{Broker} = k_{Broker} \times G. \tag{2}$$

It is assumed that the public keys of the broker PK_{Broker}, K_{Broker} is published in the system. All publishers and subscribers securely load and store these keys and the prescription for cryptographic schemes (lengths, the type of curves) and other shared parameters (e.g. q).

4.4 Join Phase

Publishers (P) who join into the system must generate Schnorr signature parameters and upload their identities securely to B. The identity ID of the i-th publisher is computed as follows:

$$ID_i = a_i \times G, \tag{3}$$

where $a_i \in_R \mathbb{Z}_q$ is publisher's private key for the Schnorr digital signature scheme, G is a point generator of the chosen elliptic curve. The private key a_i is securely stored in the publisher's device (e.g. in Secure Access Module - SAM). Every P can be added into the system by securely uploading his/her identity ID and his/her topic(s). Only RA and B can use the database server which stores publishers' ID and topics. This database also serves as the whitelist for B during the communication phase. It is assumed that S joins into the communication system dynamically. The levels $SL1$ and $SL2$ are designed for anonymous S who is not authenticated by B. S only downloads shared public keys and parameters from B.

In $SL3$, S joins to the system by uploading his/her identity to B. ID of the j-th subscriber is computed as follows:

$$ID_j = a_j \times G, \tag{4}$$

where $a_j \in_R \mathbb{Z}_q$ is the subscriber's private key for the Schnorr digital signature scheme. B stores all valid subscribers IDs for the next communication phases.

4.5 Communication Phase - Security Level 1

During the communication phase with Security Level 1 ($SL1$), *publish* messages are protected against tampering and modification. $SL1$ does not provide message confidentiality, hence, we assume only transmitting non-vital and non-personal messages which can be public, e.g. weather measurements, air quality, vehicle density, etc. Nevertheless, the sources of these messages (publishers) are authenticated by employing Schnorr digital signatures.

The communication flow is described in the following steps:

1. All subscribers, who want to subscribe to messages related to a chosen topic T, send the broker (B) the *subscribe* request that contains the name of the security level ($SL1\_1$) and the name of the topic (T). B collects all messages from publishers then forwards these messages (related to topics) to interested subscribers.

2. The i-th publisher (P_i) senses data and creates messages called *publish* messages. The *publish* message is sent to B. The message contains these parts: $SL1\_2$, T, ID_i, M, t, sig where T defines the topic in a readable string, ID_i is the identity serving as a public key of P_i (i.e., a value in $\mathbb{Z}_q$), M represents concrete sensed data, t is an actual timestamp (e.g., 8 bytes date and time) that prevents replay attacks, and sig is the Schnorr digital signature scheme. The signature $sig = z, e$ is computed by P_i with his/her private key a_i as follows:

$$R = d \times G, \tag{5}$$

where $d \in_R \mathbb{Z}_q$, $e = \mathrm{H}\ (SL1\_2, T, ID_i, M, t, R, G)$,

$$z = d - e \cdot a_i. \tag{6}$$

The signature $sig = z, e$ ensures the authentication and non-repudiation of the *publish* message from P_i.

3. B receives the *publish* message and checks the content. Firstly, B checks if ID_i is presented in the whitelist via a query to the database server and checks if the topic is correct. Then, B verifies the signature $sig = z, e$ by restoring values R', e' as follows:

$$R' = z \times G + e \times ID_i, \tag{7}$$

$e' = \mathrm{H}\ (SL1\_2, T, ID_i, M, t, R', G)$, if $e = e'$ then the signature is valid, otherwise, the signature is not valid and the message is dropped by B.

4. All validly signed messages are resent to subscribers based on required topics. It is assumed that the subscriber devices are typically smart devices that can also check the Schnorr signature by using Eq. 7, restoring e' and checking that $e = e'$.

4.6 Communication Phase - Security Level 2

During the communication phase with Security Level 2 ($SL2$), *publish* messages are protected against tampering, modification and eavesdropping. $SL2$ provides message confidentiality and partial anonymity, hence, it is appropriate for transmitting vital and personal messages, e.g., user location, etc. The sources of these messages (publishers) are authenticated by using the Schnorr digital signatures. Moreover, transmitted data are one-time encrypted by an asymmetric cipher, i.e., the Rabin cryptosystem.

The communication flow is defined in these steps:

1. Each subscriber (S) who wants to subscribe to messages related to a chosen topic T sends to B the *subscribe* request that contains the name of the security level ($SL2\_1$) and *ciphertext*, which is computed by the Rabin encryption by using the broker's public key. The plaintext $w = T||SK_{BS\_j}$ is created from the name of the topic (T) and j-th subscriber's (S_j) one time secret key (SK_{BS_j}) which is a secret key for symmetric encryption generated by a key derivation function. The condition that $w < n$ must hold where w is an encoded integer from $T||SK_{BS\_j}$. The ciphertext (c) is computed by the Rabin encryption,

$$c = w^2 \mod PK_{broker}. \tag{8}$$

   S_j stores the one time secret key (SK_{BS_j}) for a decryption operation of the *publish* message received from B.

2. Only B with his/her valid private key $privkey_{Broker}$ (r, s) is able to decrypt the ciphertext (c) that was encrypted by his/her public key PK_{Broker}. The plaintext w is restored by using the Chinese remainder theorem as follows:

$$\begin{aligned} m_r &= c^{\frac{r+1}{4}} \mod r, \\ m_s &= c^{\frac{s+1}{4}} \mod s, \end{aligned}$$

   then we compute two integers k, l such that $k \cdot r + l \cdot s = 1$,

$$\begin{aligned} w_1 &= k \cdot r \cdot m_s + l \cdot s \cdot m_r \mod PK_{Broker}, \\ w_2 &= k \cdot r \cdot m_s - l \cdot s \cdot m_r \mod PK_{Broker}, \\ w_3 &= -w_2 \mod PK_{Broker}, \\ w_4 &= -w_1 \mod PK_{Broker}. \end{aligned}$$

   Due to using the readable string of T(topic), B is able to pick the correct plaintext from 4 possible candidates w_1, w_2, w_3, w_4. Then, B creates a short term list of interested subscribers. This list contains the topics and their secret keys that have been restored from the ciphertext. To be noted here is that an observer cannot distinguish which topics are subscribed to by subscribers. Furthermore, if subscribers employ mixnets and/or onion routing techniques then B also does not know which concrete users will receive publish messages. This feature increases the privacy level of subscribers in the system.

3. The i-th publisher (P_i) produces *publish* messages containing sensed data. The *publish* message, which is sent to B, contains two parts: $SL2\_2$ and *ciphertext* that is computed by the Rabin encryption. The ciphertext is computed by using the broker's public key. Input *plaintext* ($<n$) conducts the name of the topic (T), ID_i, M, t, and sig. The sig is computed by the EC-based Schnorr signature scheme. If $w < n$ with $|n| = 4096$ b and if $|sig| = 512$ b, $|t| = 64$ b, $|T| = 128$ b, and $|ID_i| = 256$ b then the length of one message M could be up to 3136 bits. The digital signature $sig = z, e$ is computed by P_i with his/her private key a_i as follows: R is computed where
$$R = d \times G, \tag{9}$$
where $d \in_R \mathbb{Z}_q$, $e = \mathrm{H}$ ($SL2\_2$, T, ID_i, M, t, R, G), and z is computed where
$$z = d - e \cdot a_i. \tag{10}$$
The signature $sig = z, e$ ensures the authentication and non-repudiation of data values in the *publish* message.

4. B receives the *publish* message, reads that the security level is $SL2$ and decrypts the Rabin ciphertext by $privkey_{Broker}$ such as in Eq. 9 and 9. The correct plaintext is chosen from 4 candidates by helping the readable string of T. Then, B checks the content of the *publish* message and checks if ID_i is presented in the whitelist via a query to the database server. Finally, B verifies the signature $sig = z, e$ by restoring values R', e' as follows:
$$R' = z \times G + e \times ID_i, \tag{11}$$
$e' = \mathrm{H}$ ($SL2\_2$, T, ID_i, M, t, R', G), if $e = e'$ then the signature is valid, otherwise, the signature is not valid and the message is dropped by broker B.

5. All correct messages (T, M, t, sig) are then encrypted by the subscribers' one time secret keys (e.g. SK_{BS_j}) by the AES-GCM encryption and are sent as $SL2\_3$ and *ciphertext* to the subscribers. Subscribers decrypt the messages using the AES-GCM decryption. In this phase, S does not check the Schnorr signatures. The concrete ID for signature sig is stripped by B in order to increase the privacy of P. In case of an incident (e.g. bogus message, fake data), S forwards this message with the signature to the Revocation Authority RA which can detect the identity of the receiver by the trying all IDs registered for the topic in the verification process using Eq. 7, restoring e' and checking that $e = e'$.

4.7 Communication Phase - Security Level 3

During the communication phase with Security Level 3 ($SL3$), *publish* messages are protected against tampering, modification and eavesdropping. In $SL3$, P and B and S establish long-term secure session channels in order to provide message confidentiality that is suitable for frequently transmitting vital and personal data (e.g. personal power consumption, e-healthcare data) from publishers to authenticated subscribers. All messages are authenticated by the verification of the Schnorr digital signatures. Contents are encrypted by a symmetric cipher operating in authenticated mode, i.e., `AES-GCM`. $SL3$ also offers secure multicast. The communication flow is described in these steps:

1. The subscriber (S_j), who wants to subscribe to messages related to a chosen topic T, sends B the *subscribe* request which contains the message denotation ($SL3\_1$), ECIES public value (R_j), *ciphertext*, and sig_j. The *ciphertext* is computed from the plaintext (T, t, ID_j) by the AES encryption with using the secret key. The AES key is established by ECIES on the S_j side as follows: S_j knows the broker's public

key K_{Broker} established during the Setup phase. S_j generates a random number $d \in_R \mathbb{Z}_q$ and computes

$$R_j = d \times G. \tag{12}$$

Then, S_j derives shared secret $S(= Q_x)$ as:

$$Q = (Q_x, Q_y) = d \times K_{Broker}. \tag{13}$$

The secret key is derived by using a key derivation function as SK_{BS_j} =KDF(S).

Further, the Schnorr signature $sig_j = z, e$ is computed by S_j with his/her private key a_j as follows: R is computed as

$$R = d \times G, \tag{14}$$

where $d \in_R \mathbb{Z}_q$, e = H ($SL3\_1$, *ciphertext*, R, G), and z is computed such as

$$z = d - e \cdot a_j. \tag{15}$$

.

S_j stores the established secret key SK_{BS_j} for the next decryption operations of the publish messages from B.

2. B receives the *subscribe* message. B recognizes the message by $SL3\_1$ and establishes the secret key by using `ECIES` as follows:

$$Q = (Q_x, Q_y) = R_j \times k_{Broker}. \tag{16}$$

B derives shared secret $S(= Q_x)$ and restores the secret key as SK_{BS_j} =KDF(S). Furthermore, B decrypts the ciphertext encrypted by `AES-GCM` by using the key SK_{BS_j}. The restored plaintext contains T, the actual timestamp t for checking the newness, and the subscriber's ID_j for checking if the subscriber has access to this topic. B verifies the Schnorr signature sig_j by using Eq. 7 for restoring R', recomputing e' = H ($SL3\_1$, *ciphertext*, R, G) and checking that if $e = e'$, otherwise, the subscriber's request is rejected. B adds SK_{BS_j} into the long term list of interested subscribers for concrete topics.

3. The i-th publisher (P_i) senses data and creates messages called *publish* messages. The *publish* message is sent to B. The message contains three parts: $SL3\_2$, `ECIES` (R_i) and symmetric *ciphertext*. P_i knows the broker's public key K_{Broker} established during the Setup phase. P_i generates a random number $d \in_R \mathbb{Z}_q$ and computes

$$R_i = d \times G. \tag{17}$$

Then, P_i derives shared secret $S = Q_x$ as:

$$Q = (Q_x, Q_y) = d \times K_{Broker}. \tag{18}$$

The secret session key is derived by using a key derivation function where SK_{P_iB} = KDF(S). The secret key SK_{P_iB} is used by `AES-GCM` to encrypt the topic (T), ID_i, M, t, and sig_i that is computed by the EC-based Schnorr signature scheme by the same procedure described in the $SL1$ and $SL2$ with the exception of adding $SL3\_2$ into the hash function. P_i stores the established secret key SK_{P_iB} for the next encryption operations of the publish messages that are sent to B.

4. B receives the *publish* message, reads that the security message is $SL3\_2$, and starts to establish the secret key by `ECIES` as follows:

$$Q = (Q_x, Q_y) = R_i \times k_{Broker}. \tag{19}$$

B derives shared secret $S = Q_x$ and restores the same secret session key by SK_{P_iB} =KDF(S). Further, B decrypts the ciphertext encrypted by`AES-GCM` by the established secret key SK_{P_iB}. The restored plaintext contains the readable string of T (topic), ID_i, M, t, and sig_i. B checks the content of the *publish* message. Furthermore, B checks if ID_i is presented in the whitelist via a query to the database server and checks if the topic is correct.

5. All publish messages $SL3\_4$ are encrypted by subscribers' secret keys (e.g. SK_{BS_j}) and are sent to the subscribers. Each S decrypts the message by using AES and checks the Schnorr signatures by using Eq. 7 for restoring R', recomputing e' and checking that if $e = e'$, otherwise, the message is rejected.

4.7.1 Secure Multicast Mode

The broker B can decrease the number of encryption operations by using the secure multicast mode. The basic principle is depicted in Fig. 3. First, B sends all subscribers in the G-group the group secret key (SK_{group_G}) by message $SL3\_5$. Then, B encrypts only once the next x message (T, M_x, t_x, sig_{1_x}) by SK_{group_G} and broadcasts this ciphertext with denotation $SL3\_6$ to all members in the group.

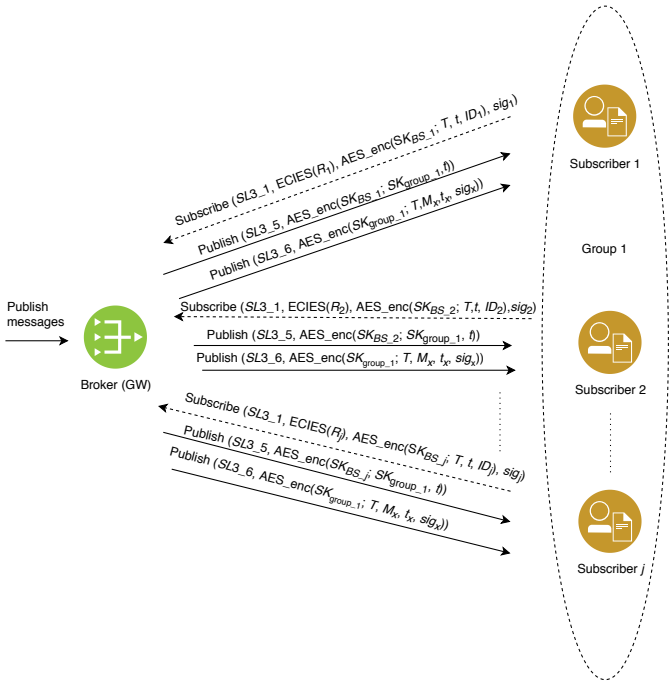

Figure 3: Secure Multicast Mode in Communication Phase - Security Level 3.

4.8 Revocation Phase

The revocation authority RA works as the third trusted party who can decide about publisher revocation. In case that a publisher produces false or malicious data, then, the

Table 2: Performance Comparison of Our Solution and Related Works

Solution	Publisher (# operations, messages)	Broker (# operations)	Subscriber (# operations, messages)
AugPAKE protocol by Shin *et al.* [19]	2SM+2M+1A+4H+1AES (4+N messages)	3SM+1M+4H+2AES 3SM+1M+4H+2AES	2SM+2M+1A+4H+1AES (4+N messages)
AugPAKE protocol by Calabretta *et al.* [5]	2SM+2M+1A+4H+ 1MAC+1AES (5+N messages)	3SM+1M+4H+2MAC+2AES 3SM+1M+4H+2MAC+2AES	2SM+2M+1A+4H +1MAC+1AES (5+N messages)
AuCPACE protocol by Haase and Labrique [7]	3SM+1KDF+6H+1AES (8+N messages)	4SM+6H+1AES 4SM+6H+1AES	3SM+1KDF +6H+1AES (8+N messages)
Our solution - $SL1$ (subscribe)	0	0	0 (1 message)
Our solution - $SL1$ (publish)	1SM+1M+1A+1H (1 message)	2SM+A1+1H	2SM+1A+1H (1 message)
Our solution - $SL2$ (subscribe)	0	1RAD	1M (1 message)
Our solution - $SL2$ (publish)	1SM+2M+1A+1H (1 message)	1RAD+2SM +1A+1H+1AES	1AES (1 message)
Our solution - $SL3$ (subscribe)	0	3SM+1A+1H+1AES	3SM+1M+1A +1H+1AES (1 message)
Our solution - $SL3$ (publish)	3SM+1M+1A+1H+1AES (1+N message)	1SM+2AES	2SM+1A+1H+1AES (1+N message)

broker or the subscriber can send a request to the RA in order to remove the publisher from the system. The RA make a decision based on the severity of the case and eventually revokes a malicious P from the system by removing his/her ID from the whitelist that links IDs to topics.

5 Performance Evaluation and Security Discussion

This section presents the performance evaluation of the proposed solution and brief gives an informal security analysis.

5.1 Performance Evaluation

Table 2 provides the comparison of performance costs that are measured by the number of math operations and cryptographic primitives required by our proposed solution (for all 3 security levels) and related works. We denote Rabin decryption operation as RAD ($\approx$ the time of modular exponentiation), scalar multiplication as SM, modular multiplication/division as M, modular addition/subtraction as A, hash function as H, message authentication code operation as MAC, and AES encryption/decryption as AES. The relatively fast operations are omitted, e.g., random number generating, XOR.

The $SL1$ version provides only data non-repudiation, integrity and authenticity but is very lightweight with several operations on each side. The $SL2$ version adds confidentiality and enhances privacy protection of subscribers and publishers. For asymmetric encryption and secure key transfer, we choose the Rabin cryptosystem that requires only one modular multiplication at the subscriber and publisher sides respectively where less powerful devices are expected.

The most robust security level $SL3$ of our solution must perform in total: 12 scalar multiplication, 2 modular multiplication, 4 modular addition, 4 hash functions, and 6 AES operations. The Shin *et al.*'s [19] and Calabretta *et al.*'s [5] solutions perform in total: 10 scalar multiplication, 6 modular multiplication, 2 modular addition, 16 hash functions and 6 AES operations. The Calabretta *et al.*'s solution [5] needs 6 MAC operations in addition. The solution of Haase and Labrique [7] needs 14 scalar multiplication, 24 hash functions and 4 AES operations. The most expensive operations are scalar and modular multiplications (14 operations in our $SL3$ compared to 16 operations in the Shin*et al.*

and Calabretta *et al.* schemes, and 14 operations in the Haase and Labrique scheme). Therefore, the performance costs of our system and the related works are comparable. Furthermore, the precise times of performance for operations depend on used devices on the subscriber, broker and publisher sides. Nevertheless, our solution in $SL3$ needs fewer messages for the authentication of entities and the establishment of the secure channel than the AugPAKE-based protocols proposed by Shin *et al.* [19], Calabretta *et al.* and the AuCPACE protocol proposed by Haase and Labrique [7], therefore, our solution is more suitable for the publish/subscribe-based MQTT protocol overall.

5.2 Security Discussion

In this section, we discuss the security properties that are supported by our solution:

- **Data non-repudiation, integrity and authenticity** - all versions ensure this property by the Schnorr signature scheme. Only publisher who holds the private key is able to sign the message.
- **Replay attack protection** - all publish messages and subscribe messages in $SL2$ and $SL3$ contain the actual time stamp that prevents replaying these messages by an attacker.
- **MitM attack protection** - the pre-distributed public keys of the broker for Rabin and `ECIES` cryptosystems in $SL2$ and $SL3$ during the setup and join phases prevent man in the middle attacks.
- **Eavesdropping protection** - $SL2$ and $SL3$ provide data confidentiality in order to prevent data eavesdropping. We employ the secure cipher with secure authenticated mode, i.e., `AES-GCM`.
- **Privacy protection** - $SL2$ enhances the privacy protection of publishers and subscribers. ID (a public key) of the publisher is stripped by the broker B. Hence, subscribers receiving data from B are not able to detect the source. Furthermore, subscribers can use mixnet or onion routing techniques in order to subscribe data anonymously. We assume that subscribers will have access to the database with public keys and topics used in $SL2$.
- **Publisher authentication** - all versions $SL1$, $SL2$ and $SL3$ authenticate the publisher by his/her signature on data.
- **Publisher revocation** - every publisher who sends malicious data can be revoked in the system by withdrawing his/her public key (ID) from the whitelist. In the privacy-friendly mode of $SL2$, the subscriber can send a malicious publish message to the revocation authority RA in order to decide about the revocation of the publisher from the system. The revocation authority has access to the database with public keys and topics used in $SL2$.
- **Subscriber authentication** - in $SL3$, a subscriber is authenticated by using his/her public key and the signature that is presented in the subscriber request message. These subscribers' public keys are registered by the broker during the join phase.
- **Broker authentication** - only the broker with his/her private key is able to decrypt the publish and subscribe messages from publishers and subscribers who use public broker keys (`ECIES` and Rabin).

6 Conclusion

In this work, we analyze security and cryptographic solutions for IoT use cases based on the MQTT protocol. The paper presents our novel security framework that is tailored for MQTT. Our solution provides 3 different security levels for different use cases such as, secure message publishing of non-sensitive but not-modified data, privacy-enhancing data publishing and subscribing, and secure data publishing from authenticated publishers to authenticated subscribers. Our protocol uses efficient public key cryptography schemes and modifications in order to suit the MQTT communication model without adding additional messages. When comparing to related work in the field, we show that the security message overhead of our solution is minimal.

Acknowledgements

We would like to thank the anonymous reviewers for detailed comments. This paper is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 830892, project SPARTA, and the National Sustainability Program under Grant no. LO1401. For the research, the infrastructure of the SIX Center was used.

References

[1] Abdalla, M., Bellare, M., Rogaway, P.: Dhaes: An encryption scheme based on the diffie-hellman problem. IACR Cryptology ePrint Archive 1999, 7 (1999)

[2] Amaran, M., Rohmad, M., Adnan, L., Mohamed, N., Hashim, H.: Lightweight security for mqtt-sn. International Journal of Engineering and Technology(UAE) 7(4), 223–226 (1 2018)

[3] Bryce, R., Shaw, T., Srivastava, G.: Mqtt-g: A publish/subscribe protocol with geolocation. In: 2018 41st International Conference on Telecommunications and Signal Processing (TSP). pp. 1–4. IEEE (2018)

[4] Bryce, R., Srivastava, G.: The addition of geolocation to sensor networks. In: ICSOFT. pp. 796–802. SciTePress (2018)

[5] Calabretta, M., Pecori, R., Velti, L.: A token-based protocol for securing mqtt communications. In: 2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM). pp. 1–6. IEEE (2018)

[6] Esfahani, A., Mantas, G., Matischek, R., Saghezchi, F.B., Rodriguez, J., Bicaku, A., Maksuti, S., Tauber, M., Schmittner, C., Bastos, J.: A lightweight authentication mechanism for m2m communications in industrial iot environment. IEEE Internet of Things Journal (2017)

[7] Haase, B., Labrique, B.: Aucpace: Efficient verifier-based pake protocol tailored for the iiot. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 1–48 (2019)

[8] Hardt, D.: The oauth 2.0 authorization framework. Tech. rep. (2012)

[9] Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S.L., Kumar, S.S., Wehrle, K.: Security challenges in the ip-based internet of things. Wireless Personal Communications 61(3), 527–542 (2011)

[10] Hernández Ramos, S., Villalba, M.T., Lacuesta, R.: Mqtt security: A novel fuzzing approach. Wireless Communications and Mobile Computing 2018 (2018)

[11] Katsikeas, S., Fysarakis, K., Miaoudakis, A., Van Bemten, A., Askoxylakis, I., Papaefstathiou, I., Plemenos, A.: Lightweight & secure industrial iot communications via the mq telemetry transport protocol. In: 2017 IEEE Symposium on Computers and Communications (ISCC). pp. 1193–1200. IEEE (2017)

[12] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., Zhao, W.: A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. IEEE Internet of Things Journal 4(5), 1125–1142 (2017)

[13] Malik, M., Dutta, M., Granjal, J.: A survey of key bootstrapping protocols based on public key cryptography in the internet of things. IEEE Access (2019)

[14] Malina, L., Hajny, J., Fujdiak, R., Hosek, J.: On perspective of security and privacy-preserving solutions in the internet of things. Computer Networks 102, 83–95 (2016)

[15] Porambage, P., Ylianttila, M., Schmitt, C., Kumar, P., Gurtov, A., Vasilakos, A.V.: The quest for privacy in the internet of things. IEEE Cloud Computing 3(2), 36–45 (2016)

[16] Rabin, M.O.: Digitalized signatures and public-key functions as intractable as factorization. Tech. rep., Massachusetts Inst of Tech Cambridge Lab for Computer Science (1979)

[17] Roman, R., Zhou, J., Lopez, J.: On the features and challenges of security and privacy in distributed internet of things. Computer Networks 57(10), 2266–2279 (2013)

[18] Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Conference on the Theory and Application of Cryptology. pp. 239–252. Springer (1989)

[19] Shin, S., Kobara, K., Chuang, C.C., Huang, W.: A security framework for mqtt. In: 2016 IEEE Conference on Communications and Network Security (CNS). pp. 432–436. IEEE (2016)

[20] Shoup, V.: A proposal for an iso standard for public key encryption (version 2.1). IACR e-Print Archive 112 (2001)

[21] Sicari, S., Rizzardi, A., Grieco, L.A., Coen-Porisini, A.: Security, privacy and trust in internet of things: The road ahead. Computer networks 76, 146–164 (2015)

[22] Singh, M., Rajan, M., Shivraj, V., Balamuralidhar, P.: Secure mqtt for internet of things (iot). In: 2015 Fifth International Conference on Communication Systems and Network Technologies. pp. 746–751. IEEE (2015)

[23] Soni, D., Makwana, A.: A survey on mqtt: a protocol of internet of things (iot). In: International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017) (2017)

[24] Source, O.: Mqtt documentation. http://http://mqtt.org/documentation, accessed: 2019-05-01

[25] Source, O.: Mqttnet. https://github.com/chkr1011/MQTTnet, accessed: 2018-10-01

[26] Stanford-Clark, A., Hunkeler, U.: Mq telemetry transport (mqtt). Online]. http://mqtt. org. Accessed September 22, 2013 (1999)