Security Architecture and Protocols for Secure MQTT-SN

CHANG-SEOP PARK1, (Member, IEEE), AND HYE-MIN NAM2

1Department of Software Science, Dankook University, Yongin 16890, Republic of Korea
2Department of Computer Science, Dankook University, Yongin 16890, Republic of Korea

Corresponding author: Chang-Seop Park (csp0@dankook.ac.kr)

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology under Grant NRF-2019R1A2C1002185.

ABSTRACT Message Queuing Telemetry Transport (MQTT) is a de facto standard for various Internet of Things (IoT) and industrial IoT applications. The data produced by a publisher are delivered to several subscribers via an MQTT broker. However, the MQTT standard lacks security-related functionalities, such as mutual authentication, access control, control message security, and end-to-end security. According to a recent report, many security threats and flaws associated with MQTT have been identified worldwide. This paper proposes security architecture and protocols to bootstrap MQTT security in the wireless sensor network. Security bootstrapping for MQTT includes security credential generation and distribution; registration protocol for joining MQTT entities such as publishers, subscribers, and brokers to the security controller; and rekeying protocol for group membership management. Special attention is given to the end-to-end security between the publishers and subscribers because the data from the publishers should not be corrupted by, and exposed to, the compromised broker. Both security analysis and performance evaluation show that our proposed security architecture and protocols for secure MQTT can be a viable solution to enhance MQTT security.

INDEX TERMS Access control, end-to-end security, group membership, MQTT-SN, IoT, security architecture, security bootstrapping, security credential.

I. INTRODUCTION Machine-to machine (M2M) protocols are key technology that significantly contribute to the fast development of Internet of Things (IoT) and industrial IoT (IIoT). Of the various existing M2M protocols, the most prevalent ones include constrained application protocol (CoAP) [1] on the client–server model and message queuing telemetry transport (MQTT) [2] on the publish–subscribe model. M2M technology based on CoAP or MQTT is found in various sectors [3–7], such as manufacturing, public administration, environment, building automation, and healthcare. CoAP provides a direct channel between a client and a server for data/command delivery, while MQTT provides an indirect channel (via a broker) between the publishers (producing data) and subscribers (consuming data). In MQTT, when receiving the data produced by publishers, the broker classifies them by topics and sends them to the subscribers who subscribe to the topics.

In particular, MQTT-SN (sensor network) [8] is optimized for the resource-constrained devices with limited processing and storage resources, so that the MQTT-SN header and payload structures are simple and short in length relative to MQTT.

Even though MQTT has become an OASIS standard, it lacks basic security-related functionalities such as access control, mutual authentication, and control message security. The only security functionality supported by the MQTT standard is unilateral authentication based on the pre-shared password. However, as no secure channel is provided between each device (publisher/subscriber) and the MQTT broker, various security issues [9], [10] have recently been raised. To address them, current implementations for MQTT [11]–[13] support transport layer security (TLS) to create a secure channel between each device and the MQTT broker. However, TLS does not provide fine-grained access control.

In this study, we investigate an adversarial model: adversaries might compromise the MQTT broker, which has not been considered in the previous works on the MQTT-SN security. Under such an adversarial model, the end-to-end
security between the publisher and the subscriber should be required. Associated with end-to-end security, two security models for the MQTT broker are defined: malicious broker model and honest-but-curious broker model. The broker on both models can intercept the data exchanged between the publisher and subscriber, while the broker can additionally deviate from the normal MQTT-SN execution on the malicious broker model. As sensitive information is stored in the MQTT broker along with the security credentials of both publishers and subscribers, a security framework for a variety of secure MQTT-SN applications is indispensable, particularly when the MQTT-SN service is provided using the MQTT broker owned by the third-party operator.

The contribution of this paper is as follows: first, having two security models for the MQTT broker in mind, we design a security architecture for MQTT-SN, which is a fundamental security framework to support both fine-grained access control and the end-to-end security. Second, we introduce the concept of topic certificate to establish a direct secure channel between a publisher and a set of subscribers, based on which the end-to-end security can be attained. Third, a couple of security protocols are designed to embed the basic security-related functionalities (such as mutual authentication and the control message security) into the standard MQTT-SN control messages, without depending on the TLS. Further, we identified various DoS attacks that can be suppressed by providing the control message security. Fourth, considering that the subscribers are group members interested in a certain topic, the rekeying mechanism is designed to revoke subscribers from the group whenever a group membership is changed. Finally, for performance evaluation, we have implemented our security mechanism proposed for MQTT-SN on the Contiki OS and Cooja simulator. It is shown that our proposed security mechanism added for secure MQTT-SN does not induce a big overhead to hinder a normal MQTT-SN execution.

In Section II, several security issues related to MQTT-SN, as well as previous works, are reviewed. The security architecture and protocols for MQTT-SN are introduced in Section III and IV, and the rekeying mechanism for topic membership change is proposed in Section V. Security analysis and performance evaluation are given in Sections VI and VII, respectively.

II. MQTT-SN SECURITY AND RELATED WORK
A. MQTT-SN AND SECURITY REQUIREMENTS
Each publisher can publish their data on a topic to the MQTT broker that maintains a prescribed set of topics (Topic1, Topic2,...). Subsequently, it publishes it to any subscriber subscribing to the topic as shown in Fig. 1. There are many MQTT-SN control messages such as connect, publish, subscribe, and pingreq for both publication and subscription, and the connection establishment and maintenance.

There are several security issues to be addressed for secure MQTT-SN: mutual authentication, access control, control message security, and end-to-end security. First, a mutual authentication between a device (publisher/subscriber) and a broker should be performed when the device attempts to connect with the broker. Additional to device authentication, broker authentication is indispensable because the device might be lured into a bogus broker.

Second, as each device can publish and subscribe on a prescribed set of topics, fine-grained access control should be enforced for each device. Even if the device is successfully authenticated to the broker, its attempt to publish/subscribe to the topics that are not authorized should be blocked. Hence, it is required for the broker to maintain a type of access control list (ACL) or to query the privilege of the device to the external server.

Third, the confidentiality and integrity of the data sent through the publish control message should be guaranteed. Depending on the security requirement of the application, the confidentiality service can be optional. Hence, two security associations should be established between the publisher and the broker, the subscriber and the broker, which means that the encrypted data sent from the publisher are decrypted and re-encrypted before being forwarded to the subscriber.

Fourth, each of the MQTT-SN control messages contains several control fields to handle the normal MQTT-SN publish–subscribe operations. When an adversary maliciously modifies some of them, an abnormal MQTT-SN operation can be induced, resulting in various DoS attacks. A couple of examples are given in Section VI.B.

Finally, if the broker is compromised, the data stored inside it can be revealed to an adversary. Furthermore, the security credentials shared with the devices can be exposed to the adversary. Hence, if the end-to-end security between the publishers and subscribers can be attained, the effect of attacking the broker can be minimized.

B. PREVIOUS STUDIES ON MQTT-SN SECURITY
Most of the previous studies on MQTT/MQTT-SN security focused on MQTT security, where both the publishers and subscribers were assumed not to be resource-constrained. They can be categorized as follows.

First, an intuitive solution is to apply TLS to secure the communication channel between the clients and the broker.
As the resource-constrained devices publishing the topic data cannot connect directly with the broker when TLS is applied, an intermediate node can be positioned between them [15], [16]. Namely, the intermediate node collects the sensing data from the resource-constrained devices. However, no security mechanisms have been suggested to secure the interface between the resource-constrained nodes and the intermediate node.

Second, as TLS is a general-purpose security protocol comprising several stages, it is not efficient for a specific application protocol such as MQTT. Depending on a password-based authenticated key exchange protocol, called AugPAKE [17], two schemes, AugMQTT [18] and MQTT-Auth [19], have been proposed for securing the topic data from an unauthorized eavesdropper. Two session keys are established by executing AugPAKE between a publisher and a broker: a broker and a subscriber. Thus, the broker should decrypt the encrypted data sent from the publisher, and then re-encrypt and send it to the subscriber, which means that the topic data are stored unencrypted on the broker side. Furthermore, they are not suitable for MQTT-SN because of their heavy computational load and message exchanges.

Third, considering the constrained property of MQTT-SN, a couple of schemes [20], [21] using the lightweight cryptography has been proposed for mutual authentication between a device and a broker, and for authenticating the topic data. Assuming each device and a broker obtain the other’s elliptic curve public key beforehand, a session key can be derived between them for securing the subsequent publish messages [20]. On the other hand, [21] has proposed a hash-based authentication scheme for the end-to-end integrity of the topic data. Given a range of possible values for a certain topic data (e.g. temperature) from a publisher, the publisher computes a HMAC-value based on the topic data and a symmetric key shared with the subscribers. When receiving it, the subscriber computes all the possible HMAC-values and compares them with it to find the topic data value. However, it is vulnerable to the replay attack.

Finally, for establishing an end-to-end security between the publishers and subscribers, several security mechanisms [22], [23] have been proposed based on attribute-based encryption (ABE) [24], where the publisher encrypts data based on a set of conditions in terms of the access policy, and subsequently, the subscribers can decrypt the ciphertext if it satisfies the access policy. On the other hand, [25] has proposed a security mechanism, assuming that the MQTT-SN gateway is positioned between the publisher and the broker. Based on the pre-shared secret key for end-to-end security between the publisher and the subscriber, the topic data is encrypted and authenticated using the AEAD (Authenticated Encryption with Associated Data)-CHACHA 20-POLY1305.

**III. PROPOSED SECURITY ARCHITECTURE FOR MQTT-SN**

**A. NETWORK/SERVICE MODEL AND DESIGN PRINCIPLES**

We adopt the network/service model of Fig. 2 consisting of several application service providers (ASPs) and broker service providers (BSPs). Each ASP installs its own publishers and subscribers to a variety of customers such as homes, utility companies, or factories. On the other hand, there are two types of BSPs: the one is a private BSP such that it is owned by a specific ASP. The other is a third-party BSP with which each ASP makes a contract for MQTT broker service. In this paper, a third-party BSP is assumed to be employed by the ASP.

For our proposed security architecture, each ASP maintains an entity, “MQTT Security Management Server (MSMS)”, which comprises two components: access controller (AC) and certificate authority (CA). The CA issues both device certificate and topic certificate to each resource-constrained device (publisher/subscriber), while the AC defines and enforces a fine-grained access control for devices. The MSMS coordinates with the devices and the broker at each stage shown in Fig. 3 for security bootstrapping.

During the “Initialization” stage, the device certificate is generated and installed into each device including the broker. Based on it, both mutual authentication and session key exchange can be performed between each device and the broker for a secure MQTT-SN. The “Registration” stage is for distributing the access control list (ACL) to the broker (Broker Registration) and the topic certificate to each device (Device Registration). Then, the “Secure MQTT-SN” stage starts for secure publication and subscription, which will be detailed in Section IV. When the topic membership change occurs among a set of subscribers authorized to access a specific topic, both the topic key update and ACL update are performed during the “ReKeying” stage, which will be also more detailed in Section V.

Let \( P = \{ P_1, P_2, \ldots \} \) and \( S = \{ S_1, S_2, \ldots \} \) be sets of authorized publishers and subscribers, respectively, and \( T = \{ \text{Topic}_1, \text{Topic}_2, \ldots \} \) be a set of valid topic names to be enrolled at the broker. The AC prepares the access control list (ACL) according to its pre-determined access policy.

**Definition 1 (Access Control List (ACL)):** Let \( P(\text{Topic}_i) \subseteq P \) and \( S(\text{Topic}_i) \subseteq S \) be sets of publishers and subscribers authorized to access \( \text{Topic}_i \). Then, \( ACL = \{ ACL_i | \text{Topic}_i \in T \} \), where \( ACL_i = \{ \text{Topic}_i, P(\text{Topic}_i), S(\text{Topic}_i) \} \).

For example, \( ACL_2 = \{ \text{Topic}_2, P(\text{Topic}_2), S(\text{Topic}_2) = \{ S_1, S_2 \} \} \) denotes that the data published on \( \text{Topic}_2 \) by \( P_1 \) are delivered to \( S_1 \) and \( S_2 \), as shown in Fig. 4.
During the Broker Registration stage, the AC passes $ACL = \{ACL_t | Topic_t \in T \}$ to the broker, based on which the broker can enforce fine-grained access control. Hence, $P_x$ cannot publish its data on $Topic_t$ if $P_x \notin P(\text{Topic}_t)$, while $S_y$ cannot subscribe to $Topic_t$ if $S_y \notin S(\text{Topic}_t)$. In this paper, it is assumed that each publisher can publish its data to only one topic name, while each subscriber can subscribe to more than one topic names. However, more than one publisher can publish their data to a single topic name. It is also assumed that $|P(\text{Topic}_t)| \gg |S(\text{Topic}_t)|$. Considering the application scenario such as industrial automation systems, they are the reasonable assumptions.

**Assumption:** Let $T(P_i) \subseteq T$ and $T(S_j) \subseteq T$ be sets of topic names, which $P_i$ and $S_j$ can access, respectively. Then,
- $|T(P_i)| = 1$, $|T(S_j)| \geq 1$ and $|P(\text{Topic}_t)| \geq 1$.
- $|P(\text{Topic}_t)| \gg |S(\text{Topic}_t)|$.

Table 1 shows the notations used throughout this paper.

### B. INITIALIZATION FOR DEVICE CERTIFICATE ISSUANCE

A lifecycle of the resource-constrained devices comprises the following stages (manufacturing/deployment/operational).

During the deployment stage, the security credentials for the publishers, subscribers, and the broker are installed into them. The security credential for each device consists of both device certificate and topic certificate, each of which is for the device identification and end-to-end security, respectively, when the secure MQTT-SN is initiated.

Both certificates are constructed based on the Elliptic Curve Qu-Vanstone (ECQV) certificate [26], [27], which is particularly well-suited for application environments where resources such as bandwidth, computing power, and storage are limited. It is much smaller (tens of bytes) than the X.509 certificate.

**Definition 2 (EC Domain Parameters $(p, a, b, G, n)$):** An elliptic curve $E(F_p)$ over a finite field $F_p$, where $p$ is a prime, is a set of points satisfying $y^2 \equiv x^3 + ax + b \pmod{p}$. The EC-domain parameters are denoted by $(p, a, b, G, n)$, where $n$ is a large prime divisor of the order of the curve $E(F_p)$ and $G$ is a base-point generator with the order of $n$.

As shown in Procedure 1, the AC performs an ECQV-certificate-issuance protocol with the CA on behalf of...
X (\(= P_i, S_j, Broker(B)\)), as a result of which a device certificate of X (\(Cert_X\)) and the device-private-key (\(d_X\)) are generated.

Procedure 1 (Device Certificate Issuance):
- on input (X)
- call genDeviceCert (\(\cdot\)) *defined in Appendix 1*/
- output (\(d_X, Cert_X\))

Suppose a pair of the private and public keys of the CA are given: (\(d_{CA}, D_{CA}\)) such that \(D_{CA} = d_{CA} \cdot G\). An ECQV certificate comprises a device’s identity and some reconstruction public data, which together with the CA’s public key are used to reconstruct the device-public-key. That is, the device-public-key is not explicitly contained in the implicit certificate \(Cert_X\). The device-public-key of X can be computed by any entity, starting from \(D_{CA}\) and \(Cert_X\), as in Procedure 2.

Procedure 2 (Public Key Derivation Function):
- on input \((D_{CA}, Cert_X)\)
- call genPub(\(\cdot\)) *defined in Appendix 2*/
- output (\(D_X\))

The security credential for X, \([d_X, Cert_X, D_{CA}]\), is pre-installed into each device \((X = P_i, S_j, B)\), and the CA also generates \([d_{iAC}, Cert_{AC}, D_{CA}]\) for the AC. As a concrete example, Fig. 5-(A) shows the result of the initialization.

![FIGURE 5. Result of initialization and registration (example).](image)

C. REGISTRATION FOR TOPIC CERTIFICATE ISSUANCE
A topic certificate for each \(Topic_i \in T\) is defined for attaining end-to-end security between \(P(Topic_i)\) and \(S(Topic_i)\). The AC also performs another ECQV-certificate-issuance protocol with the CA on behalf of X \((= P_i, S_j)\), as shown in Procedure 3 whose basic framework is the same as that shown in Procedure 1. The only difference between them is that \(Topic_i\) is used as input instead of X. Especially, the topic certificate for \(Topic_i\) should be updated whenever a change in the topic membership for \(Topic_i\) occurs \((l = 0, 1, 2, \ldots)\), where \(l = 0\) denotes an initial topic certificate issuance.

Procedure 3 (Topic Certificate Issuance):
- on input (\(Topic_i\))
- call genTopicCert(\(\cdot\)) *defined in Appendix 3*/
- output (\(d_i^{(l)}, Cert_i^{(l)}\))

Given \([D_{CA}, Cert_i^{(l)}], Topic_i\), the topic-public-key of \(Topic_i\) can also be computed, using \(genPub(D_{CA}, Cert_i^{(l)}, Topic_i) = D_i^{(l)}\) of Procedure 2. An initial topic certificate for \(Topic_i\) is \(Cert_i^{(0)}\), and the associated topic-private-key and topic-public-key are \(d_i^{(0)}\) and \(D_i^{(0)}\), respectively.

To distribute \([Topic_i, Cert_i^{(0)}]\) and \([Topic_i, d_i^{(0)}]\) to \(P(Topic_i)\) and \(S(Topic_i)\), respectively, the AC performs a Device Registration protocol with each of them. On the other hand, the AC also performs a Broker Registration protocol with the broker to distribute the ACL. As shown in Procedure 4, the Device/Broker Registration is for the AC to deliver the topic certificate securely to X \((= P_i, S_j)\), together with Info_{B} \(= (B, IP_{B}, Port_{B}, Cert_{B})\), where IP_{B} and Port_{B} denote the broker’s IP address and port number, respectively.

Procedure 4 (Device/Broker Registration):
- send ACL = \{ACL_{l}, Topic_{i} \in T\} to \(B\), where ACL_{l} = \{Topic_{i}, P(Topic_{i}), S(Topic_{i})\}.
- For each \(Topic_{i} \in T\) (when \(l = 0\)), send:
  - \([Topic_{i}, Cert_{i}^{(i)}, Info_{B}]\) to each \(X \in P(Topic_{i})\).
  - \([Topic_{i}, d_{i}^{(i)}, Info_{B}]\) to each \(X \in S(Topic_{i})\).

As a concrete example, Fig. 5-(B) shows the result of the registration.

Both Device and Broker Registration protocols are performed only once to create an initial security association between X \((= P_i, S_j/B)\) and the AC. Both protocols can be protected with the datagram TLS (DTLS) [28]. Among the several security modes supported by DTLS, the PSK mode is the most efficient, so the device certificates can be employed to derive the PSK, as shown in Appendix 4.

IV. PROPOSED SECURE MQTT-SN PROTOCOL
The original MQTT-SN protocol introduced in Section II.A is modified to embed the security features: mutual authentication, access control, control message security, and end-to-end security for the topic data, without employing the TLS/DTLS.

A. OUTER SECURE CHANNEL FOR MQTT-SN
CONTROL MESSAGES
After Initialization and Registration stages, each publisher \((X = P_i)/\text{subscriber} \((X = S_j)\) creates an outer secure channel with the broker \((B)\) through exchanging both \textit{connect} and \textit{connack} control messages, as in Fig. 6. Namely, an authenticated key exchange (Fig. 7) is performed for mutual authentication and session key derivation. Procedure 5 shows how the session key \((SK_X)\) can be derived based on \textit{Elliptic Curve Diffie-Hellman} (ECDH) key agreement. Then, subsequent
control messages can be securely exchanged through the outer secure channel.

Procedure 5 (Session Key Derivation): A session key $SK_X$ is generated to create an outer secure channel between $X$ (= $P_i$/$S_i$) and $B$, given $\{[d_X, Cert_X, D_{CA}, \ldots, Info_X = (\ldots, Cert_B)]\}$ and $\{[d_B, Cert_B, D_{CA}], ACL\}$ maintained by $X$ and $B$, respectively.

$$SK_X = kdf(d_X \cdot D_B || TS_X) = kdf(d_B \cdot D_X || TS_X),$$

where $D_X = genPub(D_{CA}, Cert_X, X)$ and $TS_X$ is a timestamp value generated by $X$.

In Fig. 7, the **connect** control message protected with $MIC(SK_X)$ is sent to the broker, where “…” denotes the original message fields of standard **connect** control message and $MIC(SK_X)$ denotes a message integrity code computed with $SK_X$ for all preceding fields in the message. When receiving it, the broker also derives the session key $SK_X$ and verifies if $TS_X$ is fresh ($TS_X > TS_X^*$) and $MIC(SK_X)$ is valid. $TS_X^*$ is a previous timestamp value generated by $X$, which is maintained by the broker. If the verification is successful, a session state for $X$ is created at the broker and the broker responds with the **connack** control message protected with $MIC(SK_X)$. The successful verification of $MIC(SK_X)$ by $X$ means the outer secure channel between them is created. From now on, the MQTT-SN control messages such as **publish**, **subscribe**, and **pingreq** can be securely exchanged.

**B. INNER SECURE CHANNEL FOR END-TO-END SECURITY**

Suppose $P_i$ publishes data on $Topic_i$, which will be delivered to $S_j ∈ S(Topic_i)$ via the broker ($B$). For the end-to-end security between the publisher and subscriber, the data should be encrypted.

Definition 3 (End-to-End Security): The data to be published should be exposed only to the authorized subscribers. So, even the broker should not be accessible to the data. The source authentication should be also provided such that each authorized subscriber can verify which authorized publisher generates the data.

A topic key is defined to encrypt the data symmetrically, which can be derived from the topic certificate and the device certificate of the authorized publisher. It can be derived as in Procedure 6.

Procedure 6 (Topic Key Derivation): A topic key is defined for the end-to-end security between a publisher $P_i ∈ P(Topic_i)$ and a group of subscribers $S(Topic_i)$. Given $\{[d_i, Cert_i, D_{CA}], [Topic_i, Cert_i^{(l)}]\}$ and $\{[d_j, Cert_j^{(l)}]\}$ maintained by $P_i$ and $S(Topic_i)$, respectively. The topic key for $Topic_i$ is also derived based on Elliptic Curve Diffie-Hellman (ECDH) key agreement:

$$K_{it}^{(l)} = kdf(d_i \cdot D_i^{(l)}) = kdf(d_i^{(l)} \cdot D_i),$$

where $D_i = genPub(D_{CA}, Cert_i^{(l)}, Topic_i)$ and $D_i = genPub(D_{CA}, Cert_i, P_i)$.

In Fig. 8, two topic keys for securing data on $Topic_i$ from $P_1$ and $P_2$. For example, as shown in Fig. 8, both $P_1$ and $P_2$ can publish a data on $Topic_i$, while $S_3$, $S_4$, and $S_5$ can subscribe to $Topic_9$ and receive the data. Hence, the publishers know the topic certificate $Cert_i^{(l)}$ and the subscribers know the corresponding topic-private-key $d_i^{(l)}$. The topic key $K_{19}^{(l)} = kdf(d_1 \cdot D_1^{(l)}) = kdf(d_1^{(l)} \cdot D_1)$ is employed to secure the data on $Topic_9$ published by $P_1$. On the other hand, the data on $Topic_9$ published by $P_2$ is secured by the topic key $K_{29}^{(l)} = kdf(d_2 \cdot D_2^{(l)}) = kdf(d_2^{(l)} \cdot D_2)$. Hence, the source authentication can be attained.

The data in the **publish** ($Topic_i$, data, …, $MIC(SK_i)$) control message is encrypted as follows:

**publish** ($Topic_i$, secured_data, …, $MIC(SK_i)$), where secured_data = $\{P_i, Cert_i, TS_i, (data)K_{it}^{(l)}, MIC(K_{it}^{(l)})\}$

The data is symmetrically encrypted with $K_{it}^{(l)}$ and $MIC(K_{it}^{(l)})$ is a message integrity code computed with $K_{it}^{(l)}$ for $P_i$, $Cert_i$, $TS_i$, and $(data)K_{it}^{(l)}$, which means that an *inner*
secure channel in Fig. 6 is created for “data” to be delivered from $P_i$ to $S_j \in S(\text{Topic}_i)$.

When receiving the publish control message, the broker first checks if $P_i \in P(\text{Topic}_i)$ based on $ACL_t = (\text{Topic}_i, P(\text{Topic}_i), S(\text{Topic}_i))$, and verifies if $MIC(SK_i)$ is valid. If both verifications are successful, the broker stores secured_data until it is requested by the subscribers in $S(\text{Topic}_i)$. Then, the broker sends the publish control message to $S_j \in S(\text{Topic}_i)$.

### Publish (Topic$_i$, secured_data, …, MIC(SK$_j$))

After checking $MIC(SK_j)$, each subscriber in $S(\text{Topic}_i)$ derives the topic key $K_{it}^{(1)}$ using $d_{t}^{(1)}$ and $D_i = genPub(D_{CA, Cert, P_i})$, based on which $MIC(K_{it}^{(1)})$ can be verified and “data” can be obtained.

### V. REKEYING PROTOCOL FOR TOPIC MEMBERSHIP CHANGE

#### A. TOPIC CERTIFICATE, TOPIC KEY, AND TOPIC MEMBERSHIP

$ACL = \{ACL_t \mid Topic_t \in T\}$, where $ACL_t = (\text{Topic}_i, P(\text{Topic}_i), S(\text{Topic}_i))$, is managed by the AC and passed to the broker. Whenever a topic membership change in $S(\text{Topic}_i)$ occurs, namely from $S(\text{Topic}_i)$ to $S^*(\text{Topic}_i)$, the AC updates $ACL_t$ and notifies the broker about it. Furthermore, the topic key derived from the topic’s private and public keys are updated. As the topic-private-key is shared among $S(\text{Topic}_i)$, the AC generates and distributes a new topic-private-key to each subscriber in $S^*(\text{Topic}_i)$. In addition, a new topic-public-key (topic certificate) is also distributed to each publisher in $P(\text{Topic}_i)$.

![FIGURE 9. Rekeying protocol for topic membership change.](image-url)

Figure 9 shows an example rekeying protocol when a subscriber no longer subscribes to a topic. Suppose $ACL_0 = (\text{Topic}_i, P(\text{Topic}_i)) = \{P_1, P_2\}$, $S(\text{Topic}_i) = \{S_3, S_4, S_5\}$, namely the data published on Topic$_i$ by $P_1$ and $P_2$ are delivered to $S_3$, $S_4$, and $S_5$. The AC generates a new topic certificate $Cert_0^{(0)}$ and the topic-private-key $d_0^{(0)}$ for Topic$_i$ for the first time ($l = 0$), as in $\circ$ of Fig. 9. Then, $\{\text{Topic}_i, Cert_0^{(0)}\}$ is sent to $P_1$ and $P_2$, while $\{\text{Topic}_i, d_0^{(0)}\}$ is sent to $S_3$, $S_4$, and $S_5$ during the Device Registration stage.

Suppose $S_5$ leaves from a group of subscribers $S(\text{Topic}_0)$. Then, the AC generates a new topic certificate $Cert_0^{(1)}$ and the associated topic-private-key $d_0^{(1)}$, which should be securely delivered to $\{P_1, P_2\}$ and $S^*(\text{Topic}_0) = \{S_3, S_4\}$, respectively, as in $\odot$ of Fig. 9.

#### B. REKEYING TOPIC CERTIFICATE AND TOPIC-PRIVATE KEY

To update the topic-public-key for Topic$_0$ from $D_0^{(0)}$ to $D_0^{(1)}$, the AC multicasts ($P_i \leftarrow AC$) an updateReq message to each $P_i$ in $P(\text{Topic}_0)$, as in Protocol 1. To guarantee the integrity of the multicast message, a group key is defined.

**Definition 4 (Group Key For Multicast Message):** A group key $(GK_t)$ is defined for each Topic$_i \in T$ to secure the updateReq message sent from the AC. It should be shared in advance among the publishers in $P(\text{Topic}_i)$.

When receiving it, each $P_i$ checks if $MIC(GK_0)$ is valid, where $GK_0$ is a group key shared among $P(\text{Topic}_i)$. If the verification is successful, each $P_i$ responds with an updateRsp message protected with $MIC(GSK_i)$ and derives $D_1^{(1)} = genPub(D_{CA, Cert_1^{(1)}, \text{Topic}_0})$, where $GSK_i$ has been derived to secure the Device Registration in Procedure 4. The AC should keep multicasting the updateReq message until it receives the updateRsp messages from all publishers in $P(\text{Topic}_0)$. It will be shown in Section VII that the number of multicast retransmissions under an unreliable UDP communication is not high even when the number of publishers in $P(\text{Topic}_0)$ is large.

**Protocol 1 (Topic-Public-Key Update Protocol):**

1. $P_i \leftarrow AC$: updateReq $(\text{Topic}_0, Cert_0^{(1)}, MIC(GK_0))$
2. $P_i \rightarrow AC$: updateRsp $(Cert_1^{(1)}, MIC(GSK_i))$

To update the topic-private-key for Topic$_0$ from $d_0^{(0)}$ to $d_0^{(1)}$, the AC sends it in unicast to each subscriber $S_j$ in $S^*(\text{Topic}_0) = \{S_3, S_4\}$, as in Protocol 2. To secure the update protocol, a secure channel is created between the AC and $S_j$ based on DTLS with the PSK mode, where the PSK is $PSK_j$ derived to secure the Device Registration of Procedure 4. As it is assumed that $|P(\text{Topic}_0)| \gg |S(\text{Topic}_0)|$ in Section III.A, it is feasible for the AC to establish a separate DTLS session with each subscriber in $S^*(\text{Topic}_0)$.

**Protocol 2 (Topic-Private-Key Update Protocol):**

1. $S_j \leftarrow AC$: DTLS handshake with $PSK_j$
2. $\leftarrow$ secure channel is created
3. $S_j \leftarrow AC$: updateReq$(\text{Topic}_0, d_0^{(1)})$
4. $S_j \rightarrow AC$: updateRsp $(\text{Code} = \text{“accepted”})$

### VI. SECURITY ANALYSIS AND DISCUSSIONS

#### A. ADVERSARIAL MODEL FOR SECURE MQTT-SN

In addition to devices (publishers/subscribers) and MQTT broker, the MSMS (AC/CA) is an entity that also interacts with them, and it is introduced for the proposed security architecture and associated protocols. However, as the MSMS is only involved with the (device/topic) certificate issuance
and update operations, it does not participate in the proposed Secure MQTT-SN protocol.

An adversarial model for the proposed Secure MQTT-SN is based on the following assumptions: it is assumed that the adversary can control the communication path between the devices and the broker. It is also assumed that the MSMS is not compromised. However, the device might be compromised, such that it can also attempt to forge other device certificates based on its device certificate. Especially, the device leaving from a group of subscribers subscribing to a specific topic can attempt to access the topic data.

Considering that the broker might be compromised, two adversarial models for the broker are defined: honest-but-curious broker and a malicious broker. The broker on both models can intercept the data exchanged between the publisher and subscriber, while the broker can additionally deviate from the normal MQTT-SN execution on the malicious broke $r$ model, which will be more detailed in Section VI.D.

**B. DOS ATTACK AND COUNTERMEASURE**

If the integrity of each MQTT control message is not guaranteed, various DoS attacks can be mounted against the MQTT publication and subscription protocols. As examples in this subsection, “keep alive” timer and “topic name/topic identifier” of MQTT are investigated to explain a possible DoS attack that can be mounted if the integrity of connect and subscribe control messages is not guaranteed.

First, even if a 2-byte “keep alive” timer field is not shown in the connect control message as in Fig. 7, it is the maximum time interval, measured in seconds, permitted to elapse between the point at which the device finishes transmitting one MQTT control message and that at which it starts sending the next. It is the responsibility of the device to ensure that the interval between MQTT control messages being sent does not exceed the “keep alive” value. In the absence of sending any other MQTT control message, the device must send a pingreq control message before the “keep alive” value expires. Suppose that the “keep alive” timer field of the connect control message is set to 30, which means that the device should send a pingreq control message if no messages are exchanged between the device and the broker within 30 s. When the “keep alive” timer value is maliciously modified from 30 to 0, a connection between them is never established.

Second, subscribe and suback control messages contain a topic name $Topic_i$ and a topic identifier $Tid_i$, respectively, for using a compact 2-byte topic identifier instead of a long human-readable topic name. For example, if $Tid_i$ corresponding to $Topic_i$ (device$_g$’s temperature) is changed to $Tid_j$ corresponding to $Topic_j$ (device$_g$’s temperature), the subscriber happens to receive the wrong data from the unintended publisher.

**C. SECURITY OF DEVICE AND TOPIC CERTIFICATES**

The proposed device and topic certificates are constructed based on the ECQV certificate, and the basic security of the ECQV certificate has been given in [14]. As these certificates are securely generated between the CA and AC inside MSMS, the certificates ($Cert_X$, $Cert_Y^{(1)}$) and the implicit signatures ($w_X$, $w_Y^{(1)}$) in Appendix 1 and 3 are not exposed even to the authorized devices.

However, when issuing a device certificate $Cert_X$ ($X = P_i$, $S_j$, $B$), both AC and CA should use a distinct random number ($r_X$, $r_{AC}$) for each device certificate. Otherwise, the private key of CA, $d_{CA}$, can be exposed if more than two devices collide. Suppose $d_1$, $d_2$, and $d_3$ are the device-private-keys of $S_1$, $S_2$, and $S_3$, respectively, such that $d_j = d_{CA} + r_{CA} \cdot H(Cert_X||S_j)+ r_j \cdot H(Cert_X||S_j)$ for $j = 1, 2, 3$. If $r_{CA}$ is fixed and $r_1 = r_2 = r_3$, then they can derive $d_{CA}$ from the above equations. This restriction also holds when the topic certificate is generated. Furthermore, it is not feasible for an adversary, or even $X$ ($= P_i$, $S_j$, $B$), to derive a new device certificate ($Cert_X$, $X^*$) from its own device certificate ($Cert_X$, $X$), namely to replace $X$ by $X^*$, without changing $Cert_X$ and $d_j$. Hence, the adversary cannot disguise another entity. We show this in Proposition 1.

**Proposition 1:** Given ($Cert_X$, $X$), it is infeasible to replace $X$ by $X^*$ ($\neq X$), such that $d_X = d_{CA} + r_{CA} \cdot H(Cert_X||X) + r_X \cdot H(Cert_X||X) = d_{CA} + r_{CA} \cdot H(Cert_X||X^*) + r_X \cdot H(Cert_X||X^*)$, if $H(\cdot)$ is a second-preimage resistant hash function.

**Proof:** Let $A = Cert_X||X$, $B \in \{0, 1\}^*$ and $B_1 = Cert_X||X^*$. Because $H(\cdot)$ is a second-preimage resistant hash function, it is not feasible to find any $A$ such that $H(A) = H(B)$ and $A \neq B$, given $A$. Namely, $Pr[(A, B)||H(A) = H(B) \text{ and } A \neq B] \leq negl(s)$ for a negligible function $negl(\cdot)$ and a security parameter $s = |H(\cdot)|$. It is evident that $Pr[(A, B_1)||H(A) = H(B_1) \text{ and } A \neq B_1] \leq Pr[(A, B)||H(A) = H(B) \text{ and } A \neq B] \leq negl(s)$; that is, it is not feasible to find such a specific $X^*$.

Similarly, it is also infeasible to derive a new topic certificate ($Cert^{(1)}$, $Topic^{(1)}_i$) from a given topic certificate ($Cert^{(1)}$, $Topic_i$), namely, to replace $Topic_i$ by $Topic^{(1)}$ without changing $Cert^{(1)}$ and $d^{(1)}_i$. Its proof is omitted because it is the same as that of Proposition 1.

**D. HONEST-BUT-CURIOS AND MALICIOUS BROKER**

In this subsection, two security models for the broker are defined: an honest-but-curious broker and a malicious broker. The honest-but-curious broker behaves normally according to the MQTT protocol but attempts to acquire the topic data for its own purpose. In this case, it is enough for $P_i$ and $S_j \in S(Topic_i)$ to generate and process secured_data $= [P_i$, $Cert_i$, $T_{Sec}(data)K_{w_i}^{(1)}$, $MIC(K_{w_i}^{(1)})]$, respectively, to attain end-to-end security. As the only information the broker maintains is $\{d_{CA}, Cert_B, D_{CA}, ACL\}$, where $ACL_d = (Topic_i, P(Topic_i), S(Topic_i))$, the broker cannot derive data from secured_data.

Meanwhile, the malicious broker is completely controlled by an adversary such that it deviates from the normal execution of the MQTT protocol and behaves according to the adversary’s command. Hence, in addition to acquiring the
topic data, the broker can override the original ACL and relay the topic data from a specific publisher to a specific group of subscribers for its own purpose. In this case, a minor modification to our security mechanism proposed in Section IV is required to attain end-to-end security. Proposition 2 shows the justification of such a modification.

Proposition 2: Under the malicious broker model, secured_data for Topic\textsubscript{T} generated by \( P_i \notin P(\text{Topic}_T) \) can be maliciously delivered to, and accepted by, \( S(\text{Topic}_T) \).

Proof: Given a topic private/public key pair \( (d_i^{(1)}, D_i^{(1)}) \) for \( \text{Topic}_T \) and a device private/public key pair \( (d_l, D_l) \) of \( P_l \), if \( P_l \in P(\text{Topic}_T) \), then \( P_l \) can generate secured_data = \([P_l, \text{Cert}_l, \text{TS}_l, (\text{data})K^1_{lT}, \text{MIC}(K^1_{lT})]\) acceptable by \( S(\text{Topic}_T) \), where \( K^1_{lT} = \text{kdf}(d_l \cdot D_i^{(1)}) = \text{kdf}(d_l^{(1)} \cdot D_i) \). Under the honest-but-curious broker model, if \( P_l \notin P(\text{Topic}_T) \), then secured_data = \([P_l, \text{Cert}_l, \text{TS}_l, (\text{data})K^1_{lT}, \text{MIC}(K^1_{lT})]\) generated by \( P_l \) cannot be delivered to \( S(\text{Topic}_T) \) due to the ACL, where \( K^1_{lT} = \text{kdf}(d_l \cdot D_i^{(1)}) = \text{kdf}(d_l^{(1)} \cdot D_i) \) and \( D_i^{(1)} \) is a public information. However, under the malicious broker model, secured_data generated by \( P_l \) can be delivered to \( S(\text{Topic}_T) \). When receiving it, \( S_l \in S(\text{Topic}_T) \) derives \( K^1_{lT} = \text{kdf}(d_l \cdot D_i^{(1)}) = \text{kdf}(d_l^{(1)} \cdot D_i) \) from \( [P_l, \text{Cert}_l] \) and obtains data from secured_data. A main reason it is feasible is that \( S_l \) cannot check whether \( P_l \) is authorized to publish the topic data for \( \text{Topic}_T \).

As shown in Proposition 2, it is necessary for \( S_l \) to determine if \( P_l \) is eligible to publish such topic data. Procedure 1 in Section III.B and Appendix 1 show how to generate the device certificate of \( P_l \). It is modified as follows; given \( \text{Cert}_l = R_l + r_{CA} \cdot G = R_l + R_{CA} \cdot W_l = d_{CA} + d_{CA} \cdot H(\text{Cert}_l || P_l || \text{Topic}_T) \), \( d_l = W_l + r_l \cdot H(\text{Cert}_l || P_l || \text{Topic}_T) \), and \( D_l = d_l \cdot G \). Hence, to generate the device-public-key, “\( P_l || \text{Topic}_T \)” should be used instead of “\( P_l \)”: namely, \( D_l = \text{genPub}(D_{CA}, \text{Cert}_l, P_l || \text{Topic}_T) \). Through such a modification, secured_data = \([P_l, \text{Cert}_l, \text{TS}_l, (\text{data})K^1_{lT}, \text{MIC}(K^1_{lT})] \) generated by \( P_l \) cannot be accepted by \( S_l \in S(\text{Topic}_T) \), because \( K^1_{lT} = H(d_l \cdot D_i^{(1)}) \) derived by \( S_l \) is different from \( K^1_{lT} = H(d_l \cdot D_i^{(1)}) \) derived by \( P_l \), where \( D_i^{(1)} = \text{genPub}(D_{CA}, \text{Cert}_l, P_l || \text{Topic}_T) \) and \( d_l \cdot G = D_i^{(1)} \).

E. FINE-GRAINED ACCESS CONTROL FOR TOPIC DATA

As shown in the previous subsection, it is not feasible for any authenticated publisher \( P_i \notin P(\text{Topic}_T) \) to publish a topic data for \( \text{Topic}_T \), even under the malicious broker model. Meanwhile, any subscriber \( S_k \notin S(\text{Topic}_T) \) cannot have access to secured_data from \( P_i \) for \( S(\text{Topic}_T) \) either, because the broker does not send it to \( S_k \). Moreover, under the malicious broker model, it cannot be decrypted by \( S_k \) because the topic key \( K^1_{iT} \) is not derivable by \( S_k \). Furthermore, if a subscriber \( S_j \) is revoked for \( \text{Topic}_T \), secured_data for \( S(\text{Topic}_T) \) is no longer accessible to \( S_j \), as shown in Proposition 3.

Proposition 3: Once a subscriber \( S_j \) is revoked for \( \text{Topic}_T \), secured_data for \( S(\text{Topic}_T) \) is no longer accessible. (Proof in Appendix 5)

VII. PERFORMANCE EVALUATION

A. NETWORK TOPOLOGY AND MQTT-SN MESSAGE STRUCTURE

MQTT-SN is implemented within the wireless sensor network. The IETF prescribes a wireless sensor network as a low-power wireless personal area network (LoWPAN), and IEEE 802.15.4 [29] is a de facto standard for MAC/PHY layer of the LoWPAN. Meanwhile, 6LoWPAN [30] resolves an asymmetry between the LoWPAN and IPv6 Internet’s processing capabilities so that the resource-constrained devices for sensing and actuation on LoWPAN can connect with the IPv6 Internet hosts through the 6LoWPAN border router (6LBR). Figure 10 shows the deployment of several MQTT-SN components in LoWPAN.

As shown in Figure 11, the MQTT-SN protocol is implemented within the wireless sensor network. A secure MQTT-SN message comprises an MQTT-SN control message structure with several headers. It is assumed that the LoWPAN is a static wireless sensor network and that the MQTT-SN client corresponds to the class 1/2 node [33]. For performance analysis and comparison, we have implemented the Secure Publication protocol based on the Contiki OS and the Cooja simulator. Figure 11 shows an MQTT-SN (control) message structure adapted to include our proposed security functionality, which is encapsulated with several headers (IEEE 802.15.4 header/trailer, IPv6 header, and UDP header). The MQTT-SN message comprises an MQTT-SN header and an MQTT-SN payload, where the MQTT-SN payload comprises various fields depending on the control message type (\textit{connect, connack, publish, puback, ...}). The “Security Fields” in the dotted grey-colored box are added for our proposed security mechanism.
When an IPv6 packet with the MQTT-SN message travels within the LoWPAN, it is fragmented if an IEEE 802.15.4 frame is longer than 127 bytes, which induces severe latency in processing the IPv6 packet. However, when both IPHC and NHC compressions [31] are applied, the IPv6 header can be reduced from 40-byte up to 7-byte, and the UDP header can be reduced from 8-byte up to 2-byte so that more space can be reserved for the MQTT-SN payload for application without fragmentation. The security fields \{Cert/ Cert\}, \{TS/ TS\}, \{MIC(SK)/MIC(SK)\}, and secured_data = \{P, Cert_i, TS_i, (data)K_i^{(1)}, MIC(K_i^{(1)})\} require 20-byte, 8-byte, 16-byte, and 72-byte, respectively, assuming secp160r1 [32] for \{Cert/ Cert\}, 8-byte \pi (=ClientID), 8-byte \TS, and AES-CTR-128 for encryption of 16-byte data and AES-CBC-MAC-128 for MIC. Hence, our proposed security mechanism does not fragment. 

**B. CRYPTOGRAPHIC OPERATIONS FOR SECURE MQTT-SN** 

Table 2 shows the number of cryptographic operations for each of the six MQTT-SN control messages during the Secure Publication protocol. EC denotes an EC scalar multiplication for ECDH computation or EC public key computation genPub(.). HMAC denotes HMAC-SHA-1 for key derivation kdf(\cdot), while CBC denotes AES-CBC-MAC-128 for MIC computation. AES denotes AES-CTR-128 for symmetric encryption.

| MQTT Messages | Client (Publisher) | Encryption Primitives |
|---------------|--------------------|-----------------------|
| connect       | (2* 1* 1 0)        | genPub(\cdot) secp160r1 for ECDH |
| connack       | (0 0 1 0)          | AES-CTR-128 SHA-1 AES-CBC-MAC-128 |
| register      | (0 0 1 0)          | HMAC-SHA-1 |
| regack        | (0 0 1 0)          | |
| publish       | (2* 1* 2 1)        | |
| puback        | (0 0 1 0)          | |

* denotes only one computation

One advantage of AES over lightweight cryptographic ciphers such as PRESENT or HIGHT is that the state-of-the-art motes (e.g. TI cc2538) support the hardware implementation of AES/SHA/ECC and the Coija simulator also provides the mote type of cc2538. Although HMAC based on SHA is slower than CBC-MAC, HMAC-SHA-1 is used for the key derivation function recommended by NIST Special Publication 800-56C Revision 2 (Recommendation for Key-Generation Methods in Key-Establishment Schemes).

During the Secure Publication protocol, connect, connack, register, and regack control messages are exchanged only once. However, both publish and puback control messages are exchanged multiple times until the connection between the publisher and the broker terminates. Especially, EC and HMAC of publish control message are computed only once, even though multiple publish control messages are sent to the broker.

Table 3 shows the average execution times (for 10 runs) of the cryptographic primitives employed for the secure MQTT-SN messages, where the message length and key length are set to 20 bytes and 16 bytes, respectively.

**C. PERFORMANCE OF SECURE PUBLICATION PROTOCOL**

In this subsection, the performance analysis on the proposed Secure Publication protocol, together with a comparison with that of the standard MQTT-SN, is given. As shown in Fig. 6, once the connection is established between a publisher and a broker, one or more datasets on a topic can be sent to the broker until its connection is terminated.

Even though the initial cryptographic computations for creating the security associations among the publisher, broker, and subscriber are introduced, the average computation burden per publish message decreases as the number of publish control messages increases. The power consumption for processing a series of MQTT-SN control messages (connect/connack/register/regack/publish/puback) is estimated using the Coija’s Powertrace. After both connection and registration phases are finished, the publish control messages are kept sending every 5 seconds. The result of estimation for the standard MQTT-SN and the proposed MQTT-SN is shown in Fig. 12, which is the overall energy consumption for the LoWPAN management, MQTT-SN message transmission/reception as well as the security-related operations.

Due to the security-related functionalities added to the proposed MQTT-SN, the overall energy consumption of the proposed one is relatively higher than that of the standard one. Especially in case of the proposed MQTT-SN, a high energy-consumption (dotted, red-colored box in Fig. 12-(b)) is indispensable since the initial ECC operations are required for preparing the connect control message. However, it is required only once during an entire MQTT-SN session. Since each publisher (sensor mote) also participates in the route discovery/reception as well as the security-related operations.

Even though the initial cryptographic computations for creating the security associations among the publisher, broker, and subscriber are introduced, the average computation burden per publish message decreases as the number of publish control messages increases. The power consumption for processing a series of MQTT-SN control messages (connect/connack/register/regack/publish/puback) is estimated using the Coija’s Powertrace. After both connection and registration phases are finished, the publish control messages are kept sending every 5 seconds. The result of estimation for the standard MQTT-SN and the proposed MQTT-SN is shown in Fig. 12, which is the overall energy consumption for the LoWPAN management, MQTT-SN message transmission/reception as well as the security-related operations.

**TABLE 2.** Cryptographic operations for MQTT-SN messages.

| MQTT Messages | Client (Publisher) | Encryption Primitives |
|---------------|--------------------|-----------------------|
| connect       | (2* 1* 1 0)        | genPub(\cdot) secp160r1 for ECDH |
| connack       | (0 0 1 0)          | AES-CTR-128 SHA-1 AES-CBC-MAC-128 |
| register      | (0 0 1 0)          | HMAC-SHA-1 |
| regack        | (0 0 1 0)          | |
| publish       | (2* 1* 2 1)        | |
| puback        | (0 0 1 0)          | |

* denotes only one computation

**TABLE 3.** Average execution times of cryptographic primitives.

| Cryptographic Primitives | Execution Time |
|--------------------------|----------------|
| genPub(\cdot)            | 1109.45 ms     |
| secp160r1 for ECDH       | 216.70 ms      |
| AES-CTR-128              | 1.86 ms        |
| SHA-1                    | 1.46 ms        |
| AES-CBC-MAC-128          | 0.91 ms        |
| HMAC-SHA-1               | 5.64 ms        |

* Execution time on Wiz mote of Coija
control packets is relatively higher than that of the security-related operations of the proposed MQTT-SN.

To estimate the cryptographic cost added to the proposed MQTT-SN, the cumulative energy consumption for preparing the connect/register/publish control messages including the initial ECC operations is compared in Fig. 13. The publish messages are sent every 5 seconds, and the energy consumption for pingreq and RPL control messages are not added to the cumulative energy consumption.

**D. RELIABLE MULTICAST OVER INRELIABLE UDP**

During the Topic-public-key update in Protocol 1, both updateReq and updateRsp messages are exchanged between several publishers in $P(\text{Topic}_i)$ and the AC. A single updateReq message is sent in multicast to the publishers, while the updateRsp message from each publisher is sent in unicast to the AC. As both messages are sent over unreliable UDP, a reliability mechanism (a simple stop-and-wait retransmission) on the application level is required: namely, the AC keeps retransmitting the multicast updateReq message until the updateRsp messages are received from all publishers in $P(\text{Topic}_i)$.

Figure 14 shows two state transition diagrams of the publisher’s and AC’s behavior for our proposed retransmission mechanism. It is assumed that the AC has a list of publishers belonging to a multicast group $P(\text{Topic}_i)$. After sending a multicast updateReq message, the AC enters a Wait state. Whenever receiving a unicast updateRsp message (Rcv_Rsp), it still remains in the Wait state and marks the source address of the message on the publisher list. If the sender of the received updateRsp message is the last one (Rcv_All_Rsp) on the publisher list, it enters a Done state and terminates the Topic-public-key update protocol.

However, if the AC does not receive all messages from all publishers within a pre-defined time interval (Time-out), it enters a ReTx state and retransmits the multicast updateReq message (Send_Req). Each publisher’s behavior for the Topic public-key update protocol is shown in Fig. 14-(b). When receiving a multicast updateReq message (Rcv_Req), the publisher enters an Rsp state and sends a unicast updateRsp message. After sending it, the publisher returns to the Wait state. To see the effect of the packet loss on the retransmission mechanism, the packet loss is simulated as in Procedure 7:
TABLE 4. Comparisons for security and cryptographic operations.

|        | DC | MA | CMS | FGA | E2E | RCB | No | Rekeying | Cryptographic Operations for establishing Security Association |
|--------|----|----|-----|-----|-----|-----|----|----------|---------------------------------------------------------------|
| TLS    | [15] | [16] |     | X   | X   | X   | X |          | TLS-Handshake = broker certificate verification (2048-bit DSA signature) + RSA encryption (2048-bit) + TLS-specific key derivation + 2 MICs (HMAC–SHA-1) |
| PAKE   | [18] | [19] |     | X   | X   | X   | O |          | AugPAKE = 3 modular exponentiations (secp160r1) + 2 Hash (SHA-1) |
| LC     | [20] | [21] |     | X   | X   | X   | X |          | 2 ECC encryptions (secp160r1), EC ElGamal + 1 Hash (SHA-1) + 1 HMAC (HMAC–SHA-1) for message authentication |
| ABE    | [22] | [23] |     |     | X   | O   | X |          | No setup for security association + 1 pairing operation per a topic (GitHub Pairing-based Cryptography Library) |
| E2E    | [25] |     |     | X   | X   | O   | X |          | Authenticated Encryption (AEAD-CHACHA20-POLY1305) |
| Proposed |     |     |     | O   | O   | O   | O |          | 2 ECDHs (secp160r1) + 1 HMAC (HMAC–SHA-1) + 1 Hash (SHA-1) |

LC: lightweight cryptography / DC: topic data confidentiality / MA: mutual authentication / CMS: control message security / FGA: fine-grained access control / E2E: end-to-end security / RCB: resilience to malicious broker / No: No interaction with external server for secure MQTT-SN / Rekeying: Rekeying Support for Membership Change
(△ = unilateral authentication / □ = weak end-to-end security)

Procedure 7 (Packet Loss Simulation):
- Let \( \rho \) be a packet loss rate.
- \( \text{rand} \leftarrow \text{random}(-) \).
- If \( \rho < (\text{rand mod 100})/100 \), then Send-Rsp.

For example, when \( \rho = 0.3 \) and \( \text{rand} = 2312 \), the publisher does not send the updateRsp message and returns to the Wait state, because \( \rho = 0.3 \geq 0.12 \). If no more multicast updateReq message is received within a pre-defined time interval (Timeout), the publisher enters a Done state and terminates the Topic-public-key update protocol.

To evaluate the performance of the retransmission protocol, we implement the retransmission mechanism on the Contiki OS and Cooja simulator. The number \( (M) \) of transmissions from the AC is computed as the size \( (s) \) of the multicast group increases for various packet loss rates \( (\rho = 0.05, 0.10, 0.15, 0.20, \text{and} \ 0.25) \), as shown in Fig. 15.

![FIGURE 15. Number of transmission for the size of the multicast group.](image1)

We simulated the retransmission mechanism 100 times for each case. The topology of the LoWPAN consisting of several publishers is shown in Fig. 16. As expected, the number of transmissions increases with the number of publishers and packet loss rate. In the case of \( \rho = 0.25 \), the average number of transmissions is 3.39 for \( s = 30 \), 3.87 for \( s = 40 \), and 3.96 for \( s = 50 \). Considering that a wireless network with \( \rho = 0.25 \) is a very poor environment, our proposed Topic Public-Key Update protocol, which requires retransmission, does not degrade the overall performance and can be endured because it is executed only when the topic membership is changed.

VIII. COMPARISONS WITH PREVIOUS WORKS

Table 4 shows comparisons with the previous works in terms of the security requirements and the cryptographic operations to establish a security association between an MQTT-SN client and a broker.

The confidentiality of the topic data is the basic security requirement of the MQTT-SN, so that it is supported by the previous schemes including ours.

Mutual authentication between an MQTT-SN client and a broker is supported in [18], [19] and in our proposed scheme. Unilateral authentication is not sufficient, as a man-in-the-middle attack can be mounted between them.
Control message security means the integrity protection of control fields in the MQTT-SN control messages (such as connect, publish, and pingreq messages). Without supporting it, a variety of DoS attacks can be mounted as mentioned in Section VI.B.

Since each MQTT-SN client can publish and subscribe to a prescribed set of topics, fine-grained access control for each MQTT-SN client should be enforced. Even if the MQTT-SN client is successfully authenticated to the broker, its attempt to publish/subscribe to non-authorized topics should be blocked. It can be supported by [19] and our proposed scheme.

To support end-to-end security between a publisher and a subscriber, a security association should be established between them in advance. In the case of [25], there was no mention of how to establish it. On the other hand, the security association between them can be established when the published message is sent to the MQTT-SN subscriber via the broker in the proposed scheme. ABE schemes [22, 23] are not suitable for resource-constrained devices as the cryptographic overhead of the pairing operation increases rapidly as the number of attributes increases, as shown in [34].

The malicious broker deviates from the normal MQTT-SN execution and behaves according to the adversary’s command. The broker can therefore override the original ACL and relay the topic data from a specific publisher to a specific group of subscribers for its own purpose. Our proposed scheme is resilient to such a malicious broker.

During the secure MQTT-SN phase in Fig. 2, it is desirable for the MQTT-SN client to avoid communicating with the external server because such interaction causes additional delay [15], [16], [21].

When the topic membership changes, it is necessary to perform the rekeying protocol even though it is a burden to the resource-constrained IoT-devices. However, considering home, utility companies, and factory IoT applications, there is no frequent change in topic-membership change except for the replacement of the broken and compromised devices. Unlike the previous schemes, our proposed scheme supports the rekeying protocol.

Since the security requirements of each of the previous works are different, there is no meaningful impact on the performance comparison between them. The cryptographic operations required for the MQTT-SN client to establish a security association with the broker (or directly encrypt the topic data) are enumerated in Table 4.

For performance comparison, we attempted to use as close as possible to the real (or virtual) devices employed in each of the previous schemes, as they were not used by simulators such as Cooja. In Table 5, PC, Edge, and Mote denote Ubuntu desktop, Raspberry Pi, and Cooja mote, respectively, where each cryptographic operation was implemented to compare the execution time.

In the case of TLS, the broker performs both key derivation and MIC computation on the PC, while the MQTT-SN client performs the certificate verification, RSA encryption, and MIC computation. On the other hand, both the broker and the MQTT-SN client perform cryptographic operations on the identical platform, namely ABE on the PC, E2E on the Edge, and the proposed one on the Mote.

IX. CONCLUDING REMARKS
MQTT-SN is a de facto standard for various IoT applications. However, motivated by the fact that MQTT-SN has several security issues to be addressed, we proposed a security architecture for secure MQTT-SN and several associated security mechanisms, through which mutual authentication, access control, data security, control message security, and end-to-end security can be supported. Especially, based on two security models for the broker, malicious broker and honest-but-curious broker, the necessity of establishing end-to-end security between the publishers and subscribers was also investigated. Both device and topic certificates constructed using the ECQV certificate could establish a framework to provide end-to-end security, as well as fine-grained access control, for the devices. Both security analysis and performance evaluation showed that our proposed security architecture and mechanisms for secure MQTT-SN can be a viable solution to enhance MQTT-SN security.

APPENDIX
Appendix 1: genDeviceCert()  
- on input (X)  
  - generate $R_X = r_X \cdot G$, where $r_X \in_R [1, n - 1]$.  
  - Run CA(X, $R_X$)  
  - generate $r_{CA} \in_R [1, n - 1]$  
  - return $Cert_X = R_X + r_{CA} \cdot G = R_X + R_{CA}$ and $w_X = d_{CA} + r_{CA} \cdot H(Cert_X || X)$  
- on input: $(Cert_X, w_X)$  
  - output $(d_X, Cert_X)$ such that $d_X = w_X + r_X \cdot H(Cert_X || X)$.

Appendix 2: genPub()  
- on input (DCA, $Cert_X$)  
  - compute $D_{CA} + Cert_X \cdot H(Cert_X || X) = D_X$  
  - output $(D_X)$  
  // Since $d_X = w_X + r_X \cdot H(Cert_X || X)$ and $w_X = d_{CA} + r_{CA} \cdot H(Cert_X || X)$,  
  // $d_X = d_{CA} + r_{CA} \cdot H(Cert_X || X) + r_X \cdot H(Cert_X || X)$ and  
  // $d_X \cdot G = D_{CA} + (R_X + R_{CA}) \cdot H(Cert_X || X)$.  
  // Since $Cert_X = R_X + R_{CA}$, $D_X = D_{CA} + Cert_X \cdot H(Cert_X || X)$.

Appendix 3: genTopicCert()  
for $l = 0, 1, 2, \ldots$  
- on input (Topic_i)  
  - generate $R_{i}^{(l)} = r_{i}^{(l)} \cdot G$, where $r_{i}^{(l)} \in_R [1, n - 1]$.  
  - Run CA(Topic_i, $R_{i}^{(l)}$):  
    - generate $r_{CA}^{(l)} \in_R [1, n - 1]$.  
    - return $Cert_{i}^{(l)} = R_{i}^{(l)} + r_{CA}^{(l)} \cdot G = R_{i}^{(l)} + R_{CA}$ and $w_{i}^{(l)} = d_{CA} + r_{CA}^{(l)} \cdot H(Cert_{i}^{(l)} || Topic_i)$  
  - on input $(Cert_{i}^{(l)}, w_{i}^{(l)})$  
  - output $(d_{i}^{(l)}, Cert_{i}^{(l)})$ for Topic_i such that $d_{i}^{(l)} = w_{i}^{(l)} + r_{i}^{(l)} \cdot H(Cert_{i}^{(l)} || Topic_i)$
Appendix 4: PSK Derivation for DTLS handshake for each X ∈ P ∪ S ∪ B

- X → AC: X, CertX, N_X
- X ← AC: AC, CertAC, NAC

/\ N_X and NAC are nonces generated by X and AC.
/\ X derives D_AC = genPub(D_CA, CertAC, AC).
/\ AC derives D_X = genPub(D_CA, CertX, X).
/\ PSK_X = kdf(d_X • D_AC || N_X || N_AC) = kdf(d_X • \phi(X || N_AC).

X ← AC: DTLS handshake with PSK

/\ DTLS handshake with the PSK mode is performed /\ to create a secure channel between X and AC.

Appendix 5: Proof of Proposition 3

The topic key K^{(t)}_X = H(d_X^{(t)} D_{1}) can be computed from the topic private key d_X^{(t)} for l = 0, 1, 2, .... When the AC receives (Cert^{(t)}, w^{(t)}_X) from the CA, the AC derives and distributes d_X^{(t)} = w^{(t)}_X + t^{(t)} \cdot H(Cert^{(t)} || Topic_t) to the authorized subscribers S_j ∈ S(Topic_t). Suppose S_j is revoked for Topic_t from l = l^* for some l^* ≥ 1. Now, we prove that even the AC cannot derive (Cert^{(l^*)}_j, w^{(l^*)}_j) from (Cert^{(l^*)}_j, w^{(l^*)}_j) received previously from the CA, where l = 0, 1, ... l^*-1.

For a formal proof by reduction in the random oracle model, the following experiment Π_A is defined for an elliptic curve discrete logarithm function ECDL, an adversary Adv_A, and a security parameter s.

Π_A (ECDL, Adv_A, s )

(G,Y) ← Setup^{(s)}.

The adversary Adv_A is given (G, Y) and outputs x.

Π_A (ECDL, Adv_A, s) = 1 if and only if ECDL(G, Y) = x such that Y = x \cdot G.

Definition 4: ECDL is intractable if for all probabilistic polynomial-time (PPT) adversaries Adv_A, there is a negligible function negl such that Pr[ Π_A (ECDL, Adv_A, s ) = 1] ≤ negl(s).

Figure 17 shows an overview of the proof for the following modified proposition: if ECDL is intractable, (Cert^{(l_k)}, w^{(l_k)}_1) is unforgeable in the random oracle model.

There are two main issues for the formal proof: first, how does Adv_A generate (Cert^{(l_k)}, w^{(l_k)}_1) as a response to Adv_B’s query (R^{(l_k)}, Topic_k) for l = 0, 1, ... l^*-1? This issue can be solved using the simulation based on the random oracle. Second, how does Adv_A extract d_CA such that d_CA = d_CA \cdot G and w^{(l_k)}_1 = d_CA + r^{(l_k)} \cdot H(Cert^{(l_k)} || Topic_k)? This can be done using the “forking lemma,” which has been employed to prove the unforgeability of the Schnorr signature. Adv_A obtains a forged message (R^{(l_k)}, Topic_k, Cert^{(l_k)}_1, w^{(l_k)}_1) from Adv_B, such that Cert^{(l_k)}_1 = R^{(l_k)} + r^{(l_k)} \cdot G and w^{(l_k)}_1 = d_CA + r^{(l_k)} \cdot H(Cert^{(l_k)} || Topic_k).

Using the forking lemma, the following two messages can be obtained, each of which is derived based on an identical random number r^{(l_k)}.

(R^{(l_k)}_1, Topic_{11}, Cert^{(l_k)}_{11}, w^{(l_k)}_1 = d_CA + r^{(l_k)} \cdot H(Cert^{(l_k)}_1 || Topic_{11}))

(R^{(l_k)}_2, Topic_{12}, Cert^{(l_k)}_{12}, w^{(l_k)}_2 = d_CA + r^{(l_k)} \cdot H(Cert^{(l_k)}_2 || Topic_{12}))
Then, $d_{C_A}$ can be computed through the following equations:

$$
W^{(l)}_1 - W^{(l)}_2 =
\frac{r^{(l)}_1}{r^{(l)}_2} [H(Cert^{(l)}_1 || Topic_1) - H(Cert^{(l)}_2 || Topic_2)]
$$

$$
W^{(l)}_2 = (W^{(l)}_1 - W^{(l)}_2) \cdot (H(Cert^{(l)}_1 || Topic_1) - H(Cert^{(l)}_2 || Topic_2))
$$

$$
d_{C_A} = W^{(l)}_1 - r^{(l)}_2 H(Cert^{(l)}_1 || Topic_1)
$$

It has been shown that ECDL can be computed (i.e., $x$ can be computed from $G$ and $Y$) if $(R^{(l)}_A, Topic_2, Cert^{(l)}_2, W^{(l)}_2)$ can be forged. Hence, $(Cert^{(l)}_2, W^{(l)}_2)$ is unforgeable if ECDL is intractable.

REFERENCES

[1] A. Banks and R. Gupta. MQTT Version 3.1. OASIS Standard, Apr. 2014.

[2] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP), document IETF RFC 7252, Jun. 2014.

[3] P. S. Rompas, A. A. Wardana, and Albarda. "Robust flood monitoring platform using message queue telemetry transport protocol," in Proc. Int. Conf. Inf. Technol. Syst. Innov. (ICITS), Bandung, Indonesia, Oct. 2017, pp. 269–276.

[4] K. Hantrakul, S. Sitti, and N. Tantitharanukul. "Parking lot guidance systems using message queue telemetry transport protocol," in Proc. 10th Int. Conf. Inf. Technol. Electr. Eng. (ICITEE), Busan, South Korea, Feb. 2017, pp. 1–5.

[5] C. Lesjak, D. Hein, M. Hofmann, M. Maritsch, A. Aldrian, P. Priller, T. Ebner, T. Ruprechter, and G. Pregartner. "Securing smart maintenance systems," in Proc. IEEE Int. Conf. Consum. Electron. (ICCE), Las Vegas, NV, USA, Jan. 2015, pp. 200–201.

[6] A. Stanford-Clark and H. L. Truong. "MQTT for sensor networks (MQTT-SN): Protocol specification, version 1.2." Int. Bus. Mach. Corp., Armonk, NY, USA, Tech. Rep., 2013.

[7] G. Perrone, M. Vecchii, R. Pecori, and R. Giuffreda. "The day after Mirai: A survey on MQTT security solutions after the largest cyber-attack carried out through an army of IoT devices," in Proc. 2nd Int. Conf. Internet Things, Big Data Secur., Porto, Portugal, Apr. 2017, pp. 246–253.

[8] S. N. Firdous, Z. Bai, C. Valli, and A. Ibrahim. "Modelling and evaluation of malicious attacks against the IoT MQTT protocol," in Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber. Phys. Soc. Comput. (CPSCom), IEEE Smart Data (SmartData), Exeter, U.K., Jun. 2017, pp. 748–755.

[9] Eclipse Foundation. (2018). Eclipse Mosquitto—An Open Source MQTT Broker. [Online]. Available: http://mosquitto.org

[10] Apache ActiveMQ. ActiveMQ—Flexible & Powerful Open Source Multi-Protocol Messaging. [Online]. Available: http://activemq.apache.org/apollo/

[11] D. R. L. Brown, R. Gallant, and S. A. Vanstone. "Provably secure implicit certificate schemes," in Financial Cryptography (Lecture Notes in Computer Science), vol. 2339. Berlin, Germany: Springer-Verlag, 2001, pp. 1–238.

[12] C. Bormann, M. Ersue, and A. Kerenan. Terminology for Constrained-Node Networks, document IETF RFC 7228, May 2014.

[13] B. Girgenti, P. Perazzo, C. Vallati, F. Righetti, G. Dini, and G. Anastasi. "On the feasibility of attribute-based encryption on constrained IoT devices for smart systems," in Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP), Washington, DC, USA, Jun. 2019, pp. 225–232.

[14] M. Calabretta, R. Pecori, M. Vecchii, and L. Veltri. "MQTT-auth: A token-based solution to endow MQTT with authentication and authorization capabilities," J. Commun. Softw. Syst., vol. 14, no. 4, pp. 320–331, 2018.

[15] F. De Rango, G. Potrino, M. Tropea, and P. Fazio. "Energy-aware dynamic Internet of Things security system based on elliptic curve cryptography and message queue telemetry transport protocol for mitigating replay attacks," Pervas. Mobile Comput., vol. 61, Jan. 2020, Art. no. 101105.

[16] D. Dinuculeanu and X. Cheng. "Vulnerabilities and limitations of MQTT protocol used between IoT devices," Appl. Sci., vol. 9, no. 5, Feb. 2019, Art. no. 484.

[17] M. Singh, M. A. Rajan, V. L. Shivraj, and P. Balamuralidhar. "Secure MQTT for Internet of Things," in Proc. Int. Conf. Commun. Syst. Netz. Technol., Gwalior, India, Apr. 2015, pp. 746–751.

[18] A. Rizzardi, S. Sicari, D. Miardoni, and A. Coen-Porisini. "AUPS: An open source AUthenticated Publish/Subscribe system for the Internet of Things," Inf. Syst., vol. 62, pp. 29–41, Dec. 2016.

[19] V. Goyal, O. Pandey, A. Sahai, and B. Waters. "Attribute-based encryption for fine-grained access control of encrypted data," in Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS), Alexandria, VI, USA, 2006, pp. 89–98.

[20] O. Sadio, I. Ngom, and C. Lishou. "Lightweight security scheme for MQTT/MQTT-SN protocol," in Proc. 6th Int. Conf. Internet Things, Syst. Manage. Secur. (HOTSMS), Granada, Spain, Oct. 2019, pp. 119–123.

[21] R. L. Daniel Brown, R. Gallant, and A. Scott Vanstone. "Provably secure implicit certificate schemes," in Financial Cryptography (Lecture Notes in Computer Science), vol. 2339. Berlin, Germany: Springer, Jul. 2002, pp. 156–165.

[22] SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV), Certicom Res., Standard Efficient Cryptogr., Mississauga, ON, Canada, Jan. 2013.

[23] E. Rescorla and N. Modadugu. Datagram Transport Layer Security, document IETF RFC 6347, Jan. 2012.

[24] Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Standard 802.15.4-2011, Standard for Information Technology, Jun. 2011.

[25] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets Over IEEE 802.15.4 Networks, document IETF RFC 4944, Sep. 2007.

[26] J. Hui and T. Thuenn, Compression Format for IPv6 Datagrams Over IEEE 802.15.4-Based Networks, document IETF RFC 6282, Sep. 2011.

[27] SEC 2: Recommended Elliptic Curve Domain Parameters, Version 2.0, Certicom Res., Standards Efficient Cryptogr., Mississauga, ON, Canada, Jan. 2010.

[28] C. Bormann, M. Ersue, and A. Kerenan. Terminology for Constrained-Node Networks, document IETF RFC 7228, May 2014.

[29] B. Girgenti, P. Perazzo, C. Vallati, F. Righetti, G. Dini, and G. Anastasi. "On the feasibility of attribute-based encryption on constrained IoT devices for smart systems," in Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP), Washington, DC, USA, Jun. 2019, pp. 225–232.

CHANG-SEOP PARK (Member, IEEE) received the B.A. degree from Yonsei University, in 1983, and the M.Sc. and Ph.D. degrees in computer science from Lehigh University, Bethlehem, PA, USA, in 1987 and 1990, respectively. He has been with the Department of Software, Dankook University, since 1990. He has been working on the wireless mobile network security during the last ten years. His research interests include network security, cryptographic protocols, and coding theory.

HYE-MIN NAM received the B.Sc. degree in software science from Dankook University, Republic of Korea, in 2019, where he is currently pursuing the master’s degree in computer science. His research interests include blockchain, wireless sensor network security, and cryptographic protocols.