Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.30, NO.5, Oct. 2020 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2020.30.5.859

자원제약적 임베디드 환경에서 종단간 보안을 지원하는 수정된 MQTT-SN*

남 혜 민,[1†] 박 창 섭[2‡]
[1,2]단국대학교 (대학원생, 교수)

Modified MQTT-SN Protocol for End-to-End Security in a Constrained Embedded Environment*

Hye-min Nam,[1†] Chang-seop Park[2‡]
[1,2]Dankook University (Graduate student, Professor)

요 약

MQTT-SN(Message Queuing Telemetry Transport - Sensor Network) 프로토콜은 센서 기반의 IoT(Internet of Things)환경에서 사용되는 메시지 전송 프로토콜이다. 이 MQTT-SN 프로토콜은 메시지 전송 중간에 중개자(Broker)를 둔 발행-구독 모델(Publish-Subscribe Model)로 각 IoT 장치들이 메시지를 전달 할 때 반드시 중개자를 통해 메시지를 주고 받는 모델이다. 하지만 MQTT-SN 프로토콜은 메시지 보안, 상호 인증, 접근 제어, 중개자 보안등을 만족하는 보안 관련된 기능을 제공하고 있지 않다. 이에 따라 최근 다양한 보안 문제가 발생하고 있으며, 보안이 필요한 상황이 대두되고 있다. 본 논문에서는 MQTT-SN의 보안 요구사항을 다시 한번 살펴보고, 이 프로토콜이 적용되는 IoT의 자원이 제한된 환경에서의 제약 조건을 만족하면서 보안을 향상시키는 수정된 프로토콜을 제안한다. 제안 프로토콜은 기존과 다르게, 보안 필드와 인증 서버가 추가되었으며 이를 통해 보안 요구사항을 만족시키도록 한다. 더불어 제안된 프로토콜을 실제 구현 및 테스트하고 에너지 소모 관점에서 제안된 프로토콜이 실제 사용이 가능한지 평가하도록 한다.

ABSTRACT

The MQTT-SN (Message Queuing Telemetry Transport-Sensor Network) protocol is a message transmission protocol used in a sensor-based Internet of Things (IoT) environment. This MQTT-SN protocol is a publish-subscribe model with a broker in the middle of message transmission, and each IoT device sends and receives messages through an intermediary when delivering messages. However, the MQTT-SN protocol does not provide security-related functions such as message security, mutual authentication, access control, and broker security. Accordingly, various security problems have recently occurred, and a situation in which security is required has emerged. In this paper, we review the security requirements of MQTT-SN once again, and propose a modified protocol that improves security while satisfying the constraints in the environment where the resource of IoT to which this protocol is applied is limited. Unlike the existing protocol, the security field and authentication server have been added to satisfy the security requirements. In addition, the proposed protocol is actually implemented and tested, and the proposed protocol is evaluated for practical use in terms of energy consumption.

Keywords: Internet of Things, MQTT-SN, End-to-End Security

Received(06. 29. 2020), Modified(09. 01. 2020), Accepted(10. 07. 2020)

* 본 논문은 2019년도 정부(과기정통부)의 재원으로 한국연구재단의 지원을 받아 수행된 중견연구자지원사업 성과임 (NRF-2019R1A2C1002185)

† 주저자, nhm9410@naver.com
‡ 교신저자, csp0@dankook.ac.kr(Corresponding author)

I. 서 론

1.1 개요

정보 통신 기술의 발달로 인해 다양한 사물들이 연결된 사물인터넷(Internet of Things, IoT) 개념이 등장하였다. 사물인터넷은 TV, 세탁기 등 가전제품은 물론, 자동차, 열차, 항공기와 같은 운송수단, 의료 장비, 헬스 케어 시스템과 같은 건강관리 서비스까지 다양한 분야에서 사용되고 있다[1].

이러한 사물인터넷을 가능하게 하는 여러 기술 중에서 핵심이 되는 기술이 바로 사물통신(Machine to Machine, M2M)이다. 사물통신은 기계, 센서, 컴퓨터 등 다양한 장치들이 유무선 통신 기술을 이용해 서로 정보를 교환하는 것을 의미한다. 사물통신을 통해 개별 장치들의 기능이나 성능을 개선시켜 주고 개별 장치들이 제공하지 못했던 새로운 지능형 서비스를 제공할 수 있게 되었다[2].

사물통신이 유무선 통신 기술을 이용하여 서비스를 제공하는 만큼 사용하는 통신 상의 보안이 중요해졌다. 사물인터넷이 가전제품 및 건강관리 서비스와 같이 개인의 프라이버시(Privacy)를 다루고 있을 뿐만 아니라, 자동차, 열차, 항공기와 같이 안전이 보장되어야 되는 곳에서도 사용되기 때문이다.

현재 사물통신으로 주로 사용되는 프로토콜에는 MQTT (Message Queuing Telemetry Transport)[3]가 있다. MQTT는 데이터를 생성하는 발행자(Publisher)와 데이터를 소비하는 구독자(Subscriber)가 중개자(Broker)를 통해 메시지를 전달하는 발행-구독 모델(Publish-subscribe Pattern)의 프로토콜이다. MQTT에서는 제한된 처리 및 스토리지 자원을 가진 임베디드 장치에 최적화하여 MQTT-SN(Sensor Network)을 제시하였다. MQTT-SN은 헤더 및 페이로드 구조가 MQTT에 비해 간단하고 짧은 특징을 가지고 있다. MQTT와 MQTT-SN의 프로토콜은 그림 1과 같이 진행되며 표 1은 MQTT-SN에서 주로 사용되는 제어 패킷에 대한 설명을 작성한 것이다.

문제는 MQTT와 MQTT-SN에는 상호 인증 및 메시지 무결성과 같은 보안 관련 기능이 부족하다는 것이다. MQTT 표준에서 지원하는 보안 기능은 사전 공유 비밀번호를 기반으로하는 인증 뿐이며 이는 브로커와 장치간에 보안 채널이 제공되지 않기 때문에 다양한 보안 문제가 발생할 수 있다[4].

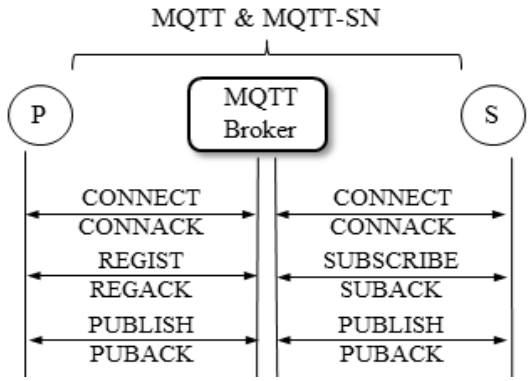

Fig. 1. MQTT-SN Process

Table 1. Some of MQTT-SN Control Packets

Control Packet	Information
CONNECT	Request connection to Server
CONNACK	Connect Acknowledgement
REGISTER	register their topics
REGACK	Register Acknowledgement
SUBSCRIBE	Subscribe to topics
SUBACK	Subscribe acknowledgement
PUBLISH	Publish message
PUBACK	Publish acknowledgement

이에따라 현재 MQTT에서는 보안 문제를 해결하기 위해서 TLS(Transport Layer Security)를 지원함으로써 각 디바이스와 MQTT 브로커 사이에 보안 채널을 생성하도록 하였다. 하지만, MQTT에서 사용되는 TLS의 경우, 높은 암호 강도를 위하여 많은 연산을 요구하고 있다. MQTT-SN[5]는 기존 MQTT보다 제한적인 자원을 가진 기기에서 사용되므로 기존 MQTT에서 적용하는 보안 서비스를 그대로 적용 하는 것은 어렵다. 따라서 성능이 상대적으로 낮은 기기를 대상으로 하는 MQTT-SN의 환경에 맞는 보안을 적용 할 필요가 있다.

본 논문에서는 MQTT-SN의 보안 요구사항을 확인하고 제한된 자원을 가진 기기에서도 보안 채널을 생성하는 것을 목표로 한다. 이를 달성하기위해 기존 MQTT-SN의 제어 메시지에 보안을 위한 필드를 추가하고 이를 이용하여 보안 채널을 생성하는 수정된 MQTT-SN을 제안한다.

논문은 다음과 같이 구성된다. 1장에서는 논문의 개요를 소개하고 MQTT-SN 프로토콜의 보안 요구사항을 살펴본다. 2장에서는 관련 연구에 관한 내용을 기술한다. 3장에서는 사물인터넷의 제한된 환경

을 확인하고 이를 고려한 수정된 MQTT-SN 프로토콜을 제안하고 이를 설명한다. 4장에서는 제안한 프로토콜을 구현하고 이를 평가하도록 한다. 5장에서는 결론을 맺도록 한다.

1.2 보안 요구사항

기존 MQTT-SN에는 보안 기능이 부족하다. 그림 2와 같이 전송되는 메시지를 훔쳐보거나 변조하는 공격이 가능하며, 중간에서 메시지 관리하는 중개자를 장악하거나, 중개자인 척 속이는 공격, 악의적인 노드가 네트워크에 참여하여 정보를 탈취 및 서비스 거부 공격이 가능하다. 이런 위협들에 대응하기 위해서 해결해야하는 몇 가지 보안 요구사항이 있다.

첫째는 메시지에 대한 보안이다. 장치(발행자/구독자)와 중개자 사이의 전송되는 메시지의 기밀성과 무결성이 보장되어야 한다. 전송되는 메시지는 MQTT -SN 발행-구독 조작을 처리하기 위한 제어 필드와 정보가 담겨있는 데이터 필드로 구분된다. 무결성은 두 필드 모두에 적용되고 기밀성의 경우 필요시에만, 데이터 필드에 적용된다.

둘째로 상호 인증이다. 장치와 중개자 간의 상호 인증이 진행되어야 한다. 장치가 가짜 중개자로 유인될 수 있으므로 중개자에 대한 인증이 필요하며 중개자 역시 악의적인 장치가 네트워크에 참여하는 것을 방지하기 위해 장치에 대한 인증이 수행되어야 한다.

셋째는 접근 제어다. 각 장치가 규정 된 토픽에 대하여 발행하고 구독 할 수 있으므로 각 장치에 대해 세분화 된 접근 제어가 시행되어야 한다. 장치가 중개자에 성공적으로 인증되더라도 권한이 없는 토픽에 대한 발행 및 구독 시도는 차단되어야 한다. 따라서 중개자가 접근 제어 목록(Access Contol List)을 유지 보수하거나 외부 서버에서 장치 권한을 조회하는 등의 접근 제어가 시행되어야 한다.

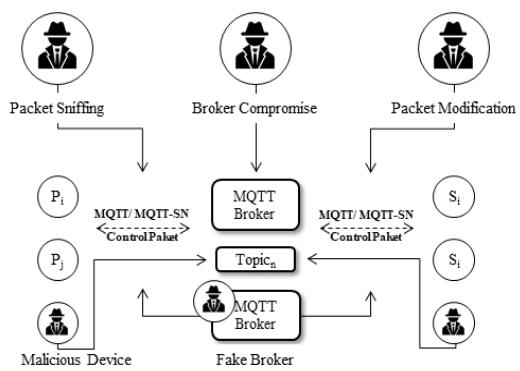

Fig. 2. MQTT Threat Model

마지막으로 중개자 보안이다. 중개자가 손상되거나 공격자에 의해 장악 된 경우, 중개자에 저장된 데이터를 보호해야 된다. 저장된 데이터는 장치와 공유된 보안 자격 증명 혹은 MQTT의 QoS(Quality of Service)에 따른 메시지가 저장 될 수 있다. 이때 발행자와 구독자 사이의 엔드-투-엔드 보안이 달성 된다면, 공격자가 브로커에 저장된 데이터를 알게 되어도 공격하는 효과가 최소화 시킬 수 있다.

II. 관련 연구

MQTT/MQTT-SN 보안에 대한 이전의 연구는 다음과 같이 진행되었으며 대부분의 연구에서 MQTT에 좀 더 중점을 두어 연구가 진행되었다.

첫째는 장치(발행자, 구독자)와 중개자 사이의 통신 채널을 보호하는 것이다. C. Lesjak 등[6]은 산업용 장치 환경에서 MQTT를 사용할 때 보안을 위하여 TLS[7]의 적용을 제안하였다. 이를통해 장치 인증과 라우팅 정보의 기밀성 및 무결성을 보장 할 수 있었다. 하지만 TLS는 장치 간 직접적인 보안채널이 아닌 중개자를 중간 노드로 데이터를 전달하게 된다. 때문에 중개자는 장치로부터 전달되는 데이터를 수집 할 수 있으며 취약점이 될 수 있다.

둘째는 발행자와 구독자 사이의 종단 간 보안을 위한 암호화이다. M. Singh 등[8]은 발행자와 구독자 사이의 종단 간 보안을 위해, 속성 기반 암호화(ABE) [9]에 기초하여 보안 메커니즘을 제안하였다. 발행자는 ABE를 사용하여 데이터를 암호화한 후 브로커로 보내고 이를 구독자에게 전달한다. 유효한 구독자는 접근 정책을 만족시키는 키를 사용하여 데이터를 해독한다. 이 때, 중개자는 메시지에 접근할 수 있는 권한이 있지만 데이터를 해독할는 수 없다. 이를 통해 종단간 보안이 실현된다. 하지만, MQTT 메시지 제어 필드에 관한 무결성이 보장되지 않아 악의적인 장치에 의한 서비스 거부 공격 및 제어 메시지 변조 공격에 취약하다.

셋째는 권한 부여 메커니즘을 통한 상호 인증 및 데이터 인증이다. A. Niruntasukrat 등[10]은 OAuth 프레임 워크를 기반으로 MQTT 플랫폼 환경에 적합한 권한 부여 메커니즘을 구현하고 새로운 인증 매커니즘을 제안하였다. 이에 따라 사용자와 인

증 서버사이에는 도청 공격, 중간자 공격, 재생 공격, 노드 캡처 공격 방지 및 악의적인 인증 서버에 대한 보호, 서비스 거부 공격으로부터 보호가 가능하다. 하지만, 해당 메커니즘에서는 장치의 제한된 자원을 수용하기 위해 암호화를 사용하지 못하였다. 따라서 메시지의 기밀성을 해결하는 것을 다음 목표로 하고 있다.

넷째는 비밀번호 기반의 인증된 키 교환 프로토콜인 AugPAKE[11]에 따라 전달되는 데이터를 보호하는 방안도 제안되었다. S. Shin 등[12]은 새로운 AugMQTT 프로토콜을 제안하였다. 해당 프로토콜은 장치와 중개자 간에 AugPAKE를 실행한다. 실행 후 두 개의 세션 키(장치와 중개자)가 설정되며 각각의 세션 키를 이용하여 중개자는 발행자가 보낸 암호화 된 데이터를 해독 한 다음 다시 암호화하여 구독자에게 보내게 된다. 즉, 데이터는 중개자 측에서 암호화되지 않은 상태로 저장된다는 취약점이 있다. 또한 계산 로드 및 메시지 교환이 많기 때문에 제한된 환경에서 사용하기에 어려움이 있다.

이외에도 종단 간 무결성을 위한 해시 기반 인증 체계도 제안되었다. Dinculeană, Dan 등[13]은 발행자와 구독자 간 공유 된 대칭 키를 기반으로 해시 기반 메시지 인증 코드(HMAC)를 계산하고 이를 비교하여 데이터 값을 확인하는 메커니즘을 제안하였다. 공유 된 키를 통해 메시지 변조 공격에 대한 무결성은 보장 받을 수 있으나, 상대적으로 메시지 재생 공격에는 취약하다는 단점이 있다.

III. 제안 프로토콜

새롭게 제안 프로토콜은 MQTT-SN의 보안 요구사항을 충족하는 것을 가장 큰 목표로 한다. 이를 위해 기존 프로토콜에 2가지 사항을 추가하였다.

첫째는 보안필드의 추가이다. 보안필드는 보안 요구 사항 중에서 개체 인증, 메시지 무결성 및 기밀성을 보장하기 위해 추가하였다. 두 번째는 인증 서버의 추가이다. 기존 MQTT-SN 프로토콜에 존재하는 3 분류의 참가자(발행자, 구독자, 중개자) 외에 인증 서버를 추가하였다. 추가된 인증 서버는 보안필드와 함께 개체 인증을 보장하는 데에 사용된다. 더불어 제안 프로토콜은 기존 프로토콜의 메커니즘을 준수하며 IoT 환경에서의 제약된 사항을 고려하였다.

다음으로는 제안 프로토콜에서 제약 조건을 살펴보고 자세한 제안 프로토콜을 소개하도록 한다.

3.1 제안 프로토콜 제약조건

제안 프로토콜은 IoT 환경에서 사물통신으로 사용되는 MQTT-SN 프로토콜을 수정하여 구성하였다. 해당 프로토콜이 자원의 제약이 있는 IoT 환경에서 이용됨에 따라 생기는 두 가지 제약조건이 존재한다.

3.1.1 IoT 기기의 암호연산의 한계

제안하는 프로토콜에서는 각 노드의 인증을 기반으로 보안 채널을 구성하고자 한다. 하지만, IoT 기기의 경우에는 기기의 전반적인 성능이 PC 환경에 비하여 매우 낮다. IoT 기기의 제한된 메모리, 대역폭, 연산 능력 안에서 제안 프로토콜을 만족시키기 위해서 적절한 암호 알고리즘을 선택해야한다. 본 논문에서는 이를 해결하기 위해서 타원 곡선 암호(Elliptic curve cryptography)를 이용한 묵시적 인증서-ECQV (Elliptic Curve Qu-Vanstone)를 사용하였다. ECQV는 묵시적 인증서로 일반 명시적 인증서와는 다르게 신원 정보, 공개키 복원 데이터로 구성된다. 묵시적 인증서는 구성된 공개키 복원 데이터와 인증기관의 공개키를 이용하여 인증서로부터 공개키를 복원한다. 이때 인증서로부터 공개키를 복원이 정상적으로 이루어지는지에 따라서 묵시적으로 인증서의 위변조가 없음을 판단하며 별도의 공개키에 대한 검증은 진행하지 않는다. 만약 묵시적 인증서의 내용이 위조되었다면 인증서로부터 복원된 공개키는 인증서 소유자의 개인키와 대응되지 않는다.[14] 이러한 ECQV는 ECC를 기반으로 동작되므로 동일한 암호화 강도의 다른 암호화 방식에 비해 키 길이가 짧고 계산 속도가 빠르다.[15] 따라서 메모리, 대역폭과 같이 자원이 제한된 IoT 환경에서의 사용에 적합하다.

3.1.2 IoT 기기의 전송 네트워크 패킷 단편화

모든 메시지는 그 길이가 길어지면 1회에 모든 내용을 보낼 수 없게 되어 단편화(Fragmentation)가 일어나게 된다. IoT 환경에서는 메시지 전송에 사용하는 전력이 전체 소비 전력 중에 많은 양을 차지한다. 따라서 전력 소비의 최적화를 위해 메시지의 단편화는 지양된다. 따라서 프로토콜을 수정하여 보안 필드를 삽입하려 할 때 단편화가 발생하지 않는

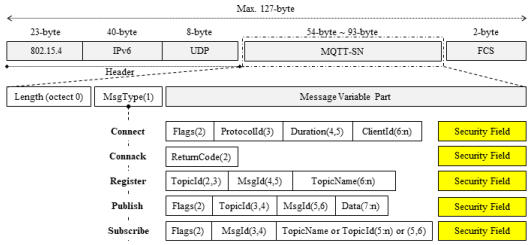

Fig. 3. MQTT-SN Packet structure

수준으로 보안 필드를 추가해야한다. 그림 3을 살펴보면 MQTT-SN의 길이는 최대 93 바이트까지 가능하다. 이는 IPHC[16]를 이용하여 IPv6 헤더를 압축하고, NHC[17]를 이용하여 UDP 헤더를 최대로 압축하였을때 가능한 수치이다. 더불어 MQTT 메시지의 타입에 따라서 사용 가능한 보안 필드의 길이가 상이하다. 이러한 점을 고려하여 적절한 암호 알고리즘의 선택 및 헤더의 압축이 적용되어야 한다.

3.2 제안 프로토콜

3.2.1 제안 프로토콜 프로세스

제안 프로토콜은 초기화, 연결, 등록, 실행의 4 단계의 프로세스로 구성된다.

첫째, 초기화 단계에서는 장치가 인증 서버와 통신을 통해 장치 인증서를 생성한다. 장치 인증서를 생성하기 위한 메커니즘은 그림 4과 같다. 우선 장치에서 타원곡선 암호의 공개키, 개인키 쌍을 생성한다. 이후 공개키와 함께 장치의 ID를 인증서버에 전송한다. 인증서버는 전달 받은 공개키와 ID를 이용하여 ECQV 인증서를 생성한다. 이후 생성된 인증서와 서명을 장치에 전송한다. 장치는 ECQV 인증서로부터 공개키를 복원하는 작업을 진행한다. 공개키가 인증서로부터 복원되면 인증서와 공개키는 묵시적으로 검증 되었다고 할 수 있다. 장치 인증서를 만드는 작업은 발행자, 구독자, 중개자 모두 진행한다.

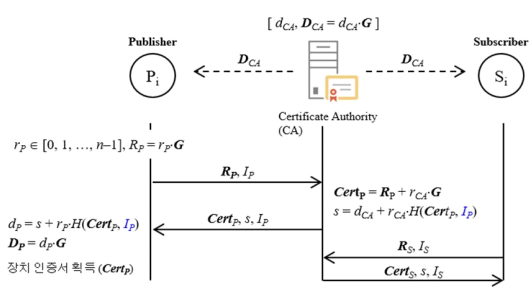

Fig. 4. Initialization phase, Generation of Device Certificate

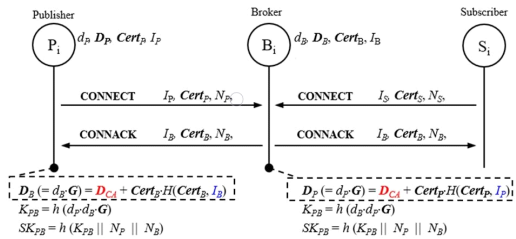

Fig. 5. Connection phase, Mutual Authentication & Session Key Generation

둘째, 연결 단계에서는 초기화 단계에서 발급된 인증서와 공개키, 개인키쌍, 장치 ID를 사용하여 각 장치와 브로커 간에 상호 인증 및 세션 키 교환을 위한 통신을 수행한다. 실제 연결 단계에서부터 기존 MQTT-SN의 패킷에 보안 필드를 추가하여 그림 5와 같이 상호 인증 및 세션키 교환을 위한 보안 인자들이 전달된다. 각 노드(장치 및 중개자)는 전달받은 인증서로부터 공개키를 추출하는 작업을 진행한다. 초기화 단계에서 인증 서버의 공개키(D_{CA})를 전달받았기 때문에 최초 초기화 단계를 수행한 정상 장치라면 공개키를 추출함으로써 묵시적 검증을 수행할 수 있다. 이후 추출된 공개키로 ECDH 키교환을 통해 마스터 키를 생성한 다음, 상호 전달받은 임의의 숫자(N_P,N_B,N_S)를 인자로 암호학적 해시 함수를 사용하여 세션키를 도출한다.

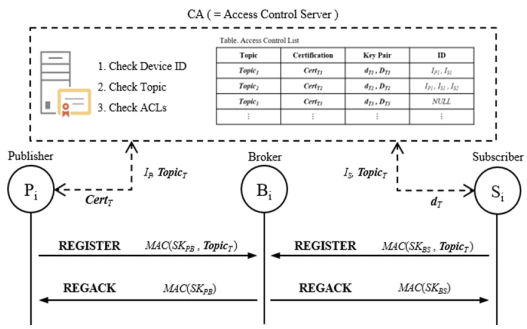

Fig. 6. Registration phase, Request & Response for Topic Certificate

셋째, 등록 단계에서는 각 장치들이 인증 서버에 원하는 토픽을 전송하고 해당 토픽에 대하여 인증 서

버로부터 발행자는 토픽 인증서를 구독자는 토픽 개인키를 발급 받는 작업이 수행 된다. 분배된 토픽 인증서와 토픽 개인키는 종단 간 보안을 위하여 메시지 기밀성을 유지하는 키를 만드는데 사용된다. 이때, 토픽 인증서와 토픽 키 쌍은 그림 4와 같은 방식으로 사전에 미리 생성된 것으로 가정한다.

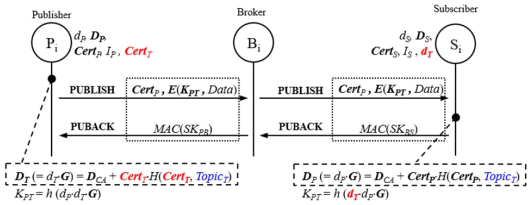

Fig. 7. Acting phase, Messaging Publish & Puback

마지막으로 실행 단계에서는 장치 인증서로 개체 인증과 메시지 무결성을, 토픽 인증서로 기밀성을 보장하는 방식으로 발행 및 구독 메시지가 전송된다.

우선 발행자와 중개자, 구독자와 중개자 사이에서 메시지 무결성이다. 이미 연결 단계에서 상호 인증을 통한 세션키(SK_{PB},SK_{BS})를 공유하였다. 실행 단계에서는 세션키를 사용하여 전체 메시지에 대한 메시지 인증 코드를 생성한다. 이를 통해 장치와 중개자 간 메시지에 대한 무결성을 보장 할 수 있다.

다음은 발행자와 구독자 간의 종단간 메시지 기밀성을 위한 데이터 암호화이다. 암호화를 위한 공유키 생성 작업은 그림 7과 같이 진행된다. 발행자는 등록 단계에서 발급받은 토픽 인증서로부터 토픽 공개키를 추출하는 작업을 수행한다. 추출된 공개키와 발행자의 개인키로 ECDH 키 교환을 수행한 후, 해당 키를 해시 함수에 적용 한 뒤 공유 키를 생성한다. 발행자는 공유 키로 데이터를 암호화하고, 발행자의 인증서를 포함하여 발행 메시지를 전송한다. 구독자는 전달받은 발행자의 인증서로부터 발행자 공개키를 추출한다. 이후 등록 단계에서 전달받은 토픽 개인키를 발행자 공개키와 ECDH 키 교환 수행한다. 이후 생성된 키를 해시 함수에 적용하여 공유키를 생성한다. 구독자는 이 키는 데이터의 복호화에 사용한다.

3.2.2 제안 프로토콜 보안 분석

본 논문의 1.2절의 보안 요구사항에서 MQTT-SN에 메시지 보안, 상호 인증, 접근 제어, 중개자 보안이 요구되었다. 제안된 프로토콜은 다음의 방법으로 보안 요구사항을 충족하도록 하였다.

첫째로 메시지 보안의 경우, 장치와 중개자 간의 공유키를 도출하고 이를 사용하여 메시지 인증 코드를 생성하였다. 메시지 인증 코드를 통해 메시지의 무결성을 보장하였다. 더불어 발행자와 구독자 간 토픽을 이용한 공유키를 도출하고 이를 사용하여 데이터를 암호화함으로써 메시지의 기밀성을 보장하였다.

둘째로 상호 인증의 경우, 프로세스의 초기화 단계와 연결 단계에서 ECQV 인증서로부터 공개키를 도출하여 공유 키를 만드는 과정을 통해 묵시적으로 검증이 진행되어 상호 인증을 보장하였다.

셋째로 접근 제어의 경우, 추가된 인증서버가 프로세스의 초기화 단계에서 접근 제어 목록을 생성 및 관리하고, 등록 단계에서 장치의 접근을 제어하였다. 이를 통해 접근 제어를 보장하여 등록되지 않은 장치의 접근으로부터 네트워크를 보호하도록 하였다.

마지막으로 중개자 보안의 경우, 발행자와 구독자 간의 공유 키를 사용한 데이터 암호화로 종단간 보안을 통해 보장하였다. 실질적인 중개자에 대한 보안은 아닐 수 있지만 종단간 보안이 달성 되었기 때문에 중개자가 저장한 데이터에 대해서 공격자의 공격 효과를 최소화 시키도록 보장 할 수 있다.

3.2.3 Scyhter 도구를 이용한 안정성 검증 분석

3.2.2 절의 다음의 네 가지의 보안 분석을 안정성 자동 검증 도구인 Scyther[18]를 사용하여 진행하였다. 다음 그림 8부터 그림 11은 프로토콜의 각 단계별 Scyther 수행결과이며 본 절에서는 각 단계별 검증에 관한 내용을 설명한다.

Fig. 8. Initialization Phase, Scyther

첫 번째로 그림 8은 초기화 단계에서 안정성 분석을 진행한 결과이다. 해당 단계에서는 인증 서버와 장치간 ECQV 연산을 통하여 장치 인증서를 얻는다. 이후 장치 인증서로부터 다음 단계에서 사용할

개인키, 공개키 쌍을 도출하는 것이 목표이다.

따라서 초기화 단계에서는 각 장치가 장치 인증서로부터 도출한 개인키, 공개키 쌍에 안정성을 검증하는 것을 목표로 분석을 진행하였다.

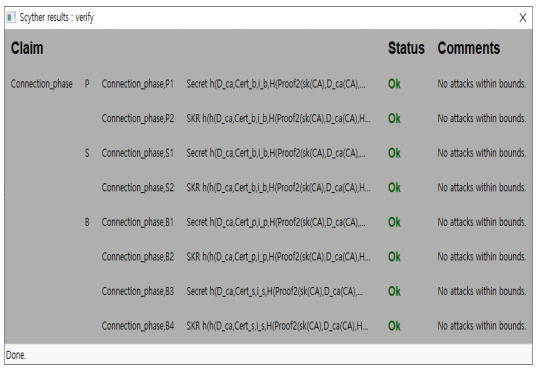

Fig. 9. Connection Phase, Scyther

다음 그림 9는 연결 단계에서 안정성 분석을 진행한 결과이다. 해당 단계에서는 장치 인증서에서 도출한 개인키, 공개키 쌍을 이용하여 발행자와 중개자간, 중개자와 구독자 간에 서로 공유할 키를 만드는 것을 목표로 한다.

따라서 연결 단계에서는 발행자와 중개자간, 중개자와 구독자 간에 생성된 공유키와 공유키로부터 파생된 세션 키에 대한 안정성 검증을 목표로 분석을 진행하였다.

다음 그림 10은 등록 단계에서 안정성 분석을 진행한 결과이다. 해당 단계에서는 서버로부터 접근제어 목록을 통해 토픽에 대한 인증서와 개인키를 발급받고, 연결 단계에서 생성한 공유키 및 세션키를 사용하여 발행자와 구독자가 중개자에게 장치의 토픽을 등록하는 단계이다.

따라서 등록 단계에서는 접근제어목록을 통해 발행자와 구독자가 등록하고자 하는 토픽에 대해서 권한을 가지고 있는지 인증하는 작업이 진행된다. 다만, Scyther 도구에서는 접근제어목록에 관한 안정성 분석에는 어려움이 있어, 개체 간 공유된 고유키 및 세션키를 통한 상호 인증에 대한 안정성 검증을 목표로 진행하였다.

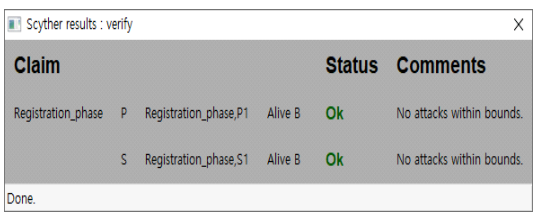

Fig. 10. Registration Phase, Scyther

다음 그림 11은 실행 단계에서 안정성 분석을 진행한 결과이다. 해당 단계에서는 등록 단계에서 획득한 토픽 인증서와 토픽 개인키를 통해 발행자와 구독자 간에 공유키를 도출하고, 해당 공유키를 통해 발행자와 구독자 간에 종단간 보안을 위한 메시지 암호화를 수행하는 것을 목표로한다.

따라서 실행 단계에서는 토픽 인증서와 토픽 개인키를 통해 발행자와 구독자 간에 생성된 공유키에 대한 안정성 검증을 수행하고, 발행자가 발행하는 메시지가 종단 간 보안이 수행되는지 대한 안정성 검증을 목표로 진행하였다.

해당 안정성 분석을 진행한 결과, 종단 간 전달되는 메시지에 대한 안정성 검사는 이상 없이 진행되었다. 하지만, 각 장치 및 중개자에 대한 인증에 대한 안정성 분석의 결과에서는 발행자가 구독자에 대한 인증을 만족하지 못하였다.

이는 발행자가 발행 메시지를 전송하고 구독자가 해당 메시지를 전달 받았는지에 대한 확인은 불가능함을 의미한다. 이를 해결하기 위해서는 MQTT-SN에서 제공하는 QoS(Quality of Service)의 단계를 높여 구독자가 메시지를 받았는지에 대한 응답을 받는 단계를 추가 구성하면 해결 할 수 있다.

하지만, QoS 단계의 증가로 인한 응답 대기시간의 증가와 전송되는 메시지의 수의 증가로 인한 추가 전력 소비를 고려하였을 때, 종단 간 전송되는 데이

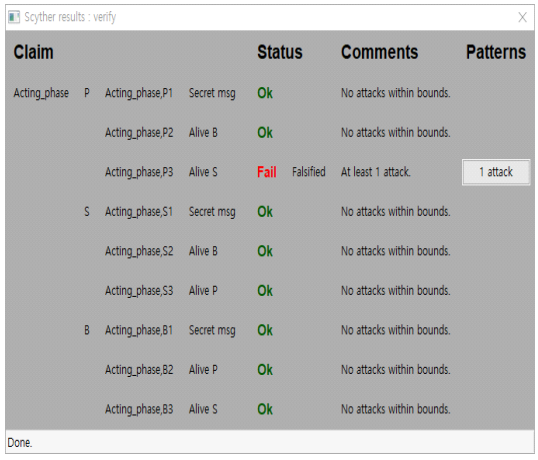

Fig. 11. Acting Phase, Scyther

터의 안정성이 보장된다면, 발행자가 구독자에 대한 인증은 필수가 아닌 선택의 문제라고 판단하였다.

IV. 구현 및 성능 평가

제안 프로토콜은 IoT 기기에서 작동된다. 표 2는 한국정보통신기술협회(TTA)에서 기기의 성능을 기준으로 IoT 기기 등급을 분류한 것이다[19]. 본 논문에서 제안하는 프로토콜은 3.1절의 IoT 환경의 제약조건을 고려하여 표 2의 등급 1의 성능을 갖춘 기기를 기준으로 제안한 프로토콜을 구현하였다. 이는 암호 알고리즘과 경량 프로토콜을 적용하기 위해서 최소 등급 1의 성능이 필요하기 때문이다. 추가적으로 제안 프로토콜의 초기화 단계에서 진행되는 인증서버와 장치의 장치 인증서 생성은 사전에 이루어졌다고 가정하여 실험을 진행하였고 이를 반영하기 위해 5초간의 딜레이 시간을 부여하였다.

Table 2. Classification according to the capabilities of IoT devices

	Characteristic	CPU Data size Code size	OS
Class 0	- Communication with the gateways and proxies - Only minimal communication ability - Pre-configured with configuration files	〈〈 10Mhz 〈〈 10KB 〈〈 100KB	Firm ware
Class 1	- Communication without gateway - Uses a lightweight protocol such as CoAP - Need to limit memory, code,power consumption	~100Mhz ~ 50KB ~ 250KB	Tiny OS, RIOT, Contiki
Class 2	- Supports existing protocol - Advantageous to use a lightweight protocol	~500Mhz ~ 250KB ~ 1MB	Embedd ed Linux
Class 3	- Existing protocol can be used - Power constraint exists	〉〉 1Ghz 〉〉 1MB 〉〉 5MB	Android iOS Tizen

4.1 구현 환경

제안 프로토콜은 그림 12와 같이 가상머신 상에서 쿠자 시뮬레이터(Cooja Simulator)[20]를 사용하여 구현하였다. 시뮬레이터에서 발행자를 위스모트(Wismote)로 생성하고 6LBR 라우터를 통해 중개

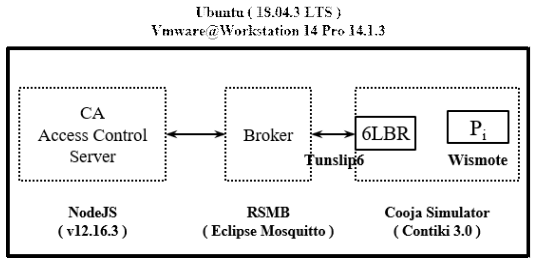

Fig. 12. Testbed for Proposed MQTT-SN

Table 3. Implementation Environment

	Device		PC
VM	Cooja simulator 〔 Wismote 〕		VM ware
CPU	Up to 18-MHz System Clock		Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz
RAM	Flash	256 KB	3 GB
	RAM	16 KB	
OS	Contiki OS		Ubuntu 18.04.2 LTS

자와 통신하였다. 중개자는 이클립스(Eclipse)의 오픈 소스 프로젝트인 Eclipse Mosquitto에서 제공하는 RSMB(Really Small Message Broker)를 사용하였다[21]. 인증 서버는 NodeJS를 사용하여 구현하였다. 각 구현에 사용된 장치와 환경에 관한 자세한 세부 능력은 표 3을 통해 확인 가능하다.

4.2 성능 평가

우선 제안 프로토콜이 IoT 환경의 제한된 자원에서 수행 가능한지 확인한다. 이후, 기존의 MQTT-SN과 제안 프로토콜을 에너지 소모 관점에서 비교 분석한다. 구현에 사용된 암호 알고리즘과 20 바이트 데이터에 대한 각 알고리즘의 계산 시간은 표 4와 같다. ECQV의 경우, 타원곡선 secp224r1에서 인증서로부터 공개키 도출에 소요되는 시간이다.

Table 4. Algorithm execution time

Algorithm	Time
AES-CTR	1.861 ms
CBC-MAC	0.915 ms
SHA256	3.448 ms
HMAC-SHA256	13.610 ms
ECQV (secp224r1)	642.730 ms

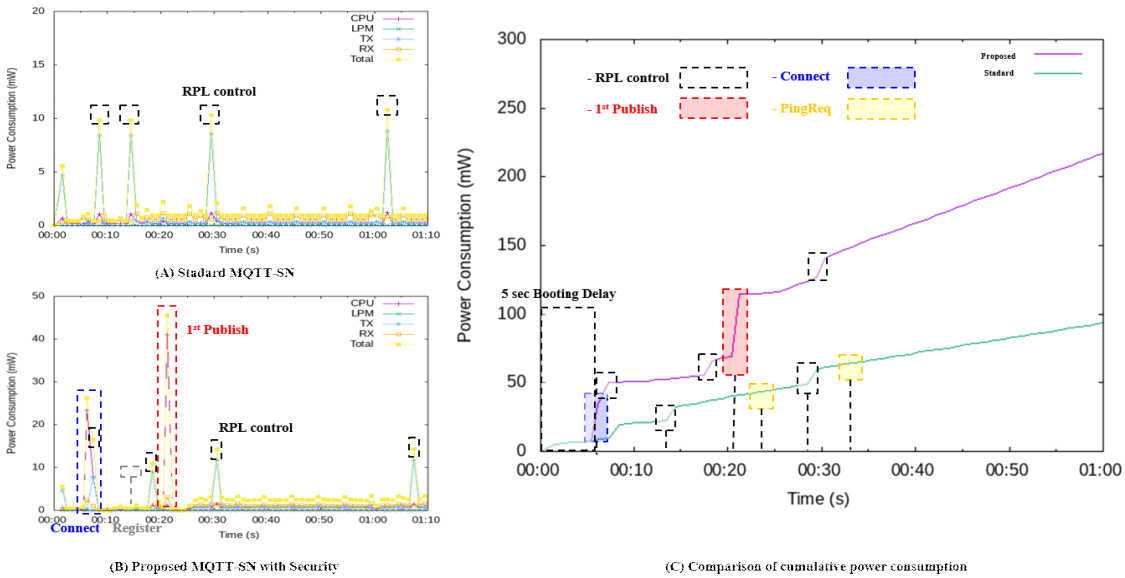

Fig. 13. Performance Evaluaion Graph

Table 5. Control Packet's Security Feild

Packet	Security Field
CONNECT	28 Bytes (Compressed Cert : 28 bytes)
CONNACK	28 Bytes (Compressed Cert : 28 bytes)
REGISTER	40 Bytes (Topic : 8 bytes) (HMAC : 32 bytes)
REGACK	32 Bytes (HMAC : 32 bytes)
PUBLISH	76 Bytes (Compressed Cert : 28 bytes) (HMAC : 32 bytes) (Encrypted Data : 16 bytes)
PUBACK	32 Bytes (HMAC : 32 bytes)

표 5는 최종적으로 제안된 프로토콜에서 보안필드가 몇 바이트를 차지하게 되는지에 대한 길이이며 3.1.2 절의 단편화가 발생하지 않도록 설정하였다.

성능 평가를 위한 전체적인 실험은 다음과 같이 진행된다. 중개자, 인증 서버가 동작 중인 상태에서 발행자와 6LBR 라우터 장치를 실행한다. 발행자는 초기 안정화를 위한 5초간의 지연 이후에, 연결, 등록, 수행 단계를 순차적으로 진행한다. 등록 단계는 1개의 토픽에 대해서만 진행 되며, 수행 단계에서는 1초 간격으로 발행 메시지(Publish)를, 5초 간격으로 핑 요청(PingReq) 메시지를 라우터를 통해 브로커로 전송한다. 실험의 결과는 그림 13과 같다.

그림 13에서 (A)와 (B)는 각각 기존 프로토콜과 제안 프로토콜의 시간당 에너지 소모율을 도표화 한 것이다. (C)의 그래프는 (A)와 (B)의 그래프를 자료로 누적된 전력 소모량을 도표화한 것이다. 그래프 (A)를 통해 기존 프로토콜의 에너지 소모를 확인해 보면, 간헐적으로 전송되는 RPL Control 메시지 전송이 가장 큰 에너지 소모율을 보였다. 하지만, 별다른 암호화 과정 혹은 긴 메시지를 전달하는 경우가 없어 평균적으로 초당 2~3 mW를 소모하였다.

반면, 제안된 프로토콜의 경우, 암호화 연산과 보안 필드의 추가로 기존 프로토콜에 비해 상대적으로 많은 에너지 소모가 진행 되었다. 우선 그래프 (B)를 확인 해보면, 약 6 초 즈음에 높은 에너지 소모를 보인다. 이는 최초 연결 단계에서 Connect 메시지를 생성할 때, ECQV 인증서로 부터 공개키를 도출하기 위한 타원곡선 암호 연산과 세션 키를 만들기 위한 ECDH 키교환 연산이 수행되었기 때문이다.

이후 약 18 초에 등록 메시지를 전달하게 된다. 이때 별다른 암호 연산이 수행되지 않았기 때문에 소모되는 에너지 소모는 낮았다. 더불어 인증서버와 중개자의 통신의 경우, PC 환경에서 수행 되었기 때문에 전송 대기 시간은 무시 할 정도로 매우 낮았다.

다음으로 약 21초 쯤 최초의 발행 메시지를 발행 시기에 매우 높은 에너지 소모를 확인할 수 있다. 이는 프로토콜의 수행 단계에서 발행 메시지를 보내기 전에 ECQV 인증서로부터 공개키를 도출하고, ECDH 키 교환이 수행 되었기 때문이다. 뿐만 아니라, 발행 메시지 이후부터는 종단간 보안을 위해서 데이터를 AES 암호화 하는 작업도 추가로 수행된다. 추가적인 암호 연산이 가장 많은 부분으로 가장 많은 에너지 소모를 하였다.

이후 전달되는 발행 메시지와 핑 요청 메시지는 약 4~5mW로 기존에 비하여 2~3mW 정도 추가 에너지가 소모된다. 이는 기존 프로토콜과 달리 추가된 보안필드에 따른 전송에너지 소모 증가와 암호화 및 메시지 인증 코드를 생성 때문에 상대적으로 시간당 에너지 소모되는 증가된 것을 확인 할 수 있다.

이렇듯 제안된 프로토콜은 기존 프로토콜에 비하여 추가 에너지 소모가 늘었고 그래프(C)를 통해 누적된 차이를 확인 할 수 있다. 초기의 공유키를 만들기 위한 암호연산에 따른 에너지 소모, 메시지의 기밀성 및 무결성을 보장하기 위해 추가된 보안 메커니즘이 기존에 비하여 에너지 소모가 그 이유이다.

V. 결 론

본 논문에서는 기존 MQTT-SN의 보안 요구사항을 확인하고 제한된 환경에서 해당 요구사항을 만족하기 위해서 수정된 프로토콜을 제시하였다. 수정된 프로토콜은 보안 필드와 인증 서버를 추가함으로써 메시지 보안, 상호 인증, 접근 제어, 중개자 보안을 달성하도록 하였다. 제안된 프로토콜은 기존 프로토콜에 비하여 암호 연산과 보안 필드가 추가되었다. 이에 따른 전력 소비가 늘어난 부분도 존재한다. 하지만, 점차 메시지 및 네트워크에 대한 보안이 중요한 상황 및 환경이 등장하는 만큼 제안된 프로토콜은 MQTT-SN의 보안을 향상 시킬 수 있는 하나의 방안이 될 수 있음을 확인 할 수 있다.

References

[1] D. Miorandi, S. Sicari, F. De Pellegrini et al., "Internet of things: Vision applications and research challenges", Ad Hoc Networks, vol. 10, no. 7, pp. 1497-1516, 2012.

[2] G. Wu, S. Talwar, K. Johnsson, N. Himayat and K. D. Johnson, "M2M: From mobile to embedded internet," in IEEE Communications Magazine, vol. 49, no. 4, pp. 36-43, April 2011,

[3] A. Banks and R. Gupta, "MQTT Version 3.1.1," OASIS Standard, Apr. 2014

[4] G. Perrone, M. Vecchio, R. Pecori, and R. Giaffreda, "The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyberattack Carried Out through an Army of IoT Devices," in Proc. Int'l Conf. on Internet of Things, Big Data and Security, pp. 246-253, Porto, Portugal, Apr. 24-26, 2017.

[5] A. Stanford-Clark and H. L. Truong, "MQTT for Sensor Networks (MQTT-SN): Protocol Specification, version 1.2", IBM, 2013.

[6] Lesjak, Christian, et al. "Securing smart maintenance services: Hardware-security and TLS for MQTT." 2015 IEEE 13th international conference on industrial informatics (INDIN). IEEE, 2015.

[7] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018

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

[9] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based Encryption for Fine-grained Access Control of Encrypted Data," in Proc. ACM Conf. on Computer and Communications Security, pp. 89 - 98,

Alexandria, Virginia, USA, Oct. 30 - Nov. 03, 2006.

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

[11] S. H. Shin and K. Kobara, "Efficient Augmented Password-Only Authentication and Key Exchange for IKEv2," IETF RFC 6628, Experimental, June 2012.

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

[13] Dinculeană, D. Cheng, X. "Vulnerabilities and Limitations of MQTT Protocol Used between IoT Devices". Appl. Sci. 2019, 9, 848.

[14] C. Park, "A Secure and Efficient ECQV Implicit Certificate Issuance Protocol for the Internet of Things Applications," in IEEE Sensors Journal, vol. 17, no. 7, pp. 2215-2223, 1 April1, 2017

[15] Kicassl, "ECC Certificate Guide", https://kicassl.com/cstmrsuprt/ntc/searchNtcDetail.sg?page=1& ntcSeq=93&mode=&searchType=subject&searchWord=&searchRowCnt=10

[16] Koren, T., Casner, S., and C. Bormann, "IP Header Compression over PPP", RFC 3544, July 2003

[17] Carlson, R. and L. Winkler, "Guidelines for Next Hop Client (NHC) Developers", RFC 2583, DOI 10.17487/RFC2583, May 1999

[18] C. Cremers, "The scyther tool," www.cs.ox.ac.uk/people/cas.cremers/scyther/ [Online: Accessed on Agust 24, 2020].

[19] TTA, "Classification and Security Requirements based on IoT Device Capabilities (TTAK.KO-12.0298) ", 2016. 12.

[20] A. Dunkels, B. Gronvall and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," 29th Annual IEEE International Conference on Local Computer Networks, Tampa, FL, USA, 2004, pp. 455-462

[21] Eclipse Opensource Project ,Really Small Message Broker, 2014, http://git.eclipse.org/c/mosquitto/org.eclipse.mosquitto.rsmb.git/

〈저자소개〉

남 혜 민 (Hye-min Nam) 학생회원
2019년 2월 : 단국대학교 소프트웨어학과 졸업
2019년 3월~현재: 단국대학교 소프트웨어 보안 석사과정
〈관심분야〉 정보보호, 블록체인, 무선 센서 네트워크 및 암호화 프로토콜

박 창 섭 (Chang-seop Park) 종신회원
1983년 2월: 연세대학교 경제학과 졸업
1987년 2월: Lehigh University 컴퓨터과학과 석사
1990년 2월: Lehigh University 컴퓨터과학과 박사
1990년 3월~현재: 단국대학교 소프트웨어학과 교수
〈관심분야〉 정보보호, 네트워크 보안, 무선인터넷 및 모바일 컴퓨팅 보안