DASH7 환경에서 안전한 메시지 전송을 위한 CoAP 기반 보안 프로토콜 설계

최슬기*, 송경환*, 송진희**, 전문석*
*숭실대학교 컴퓨터학과
**신한대학교 IT융합공학부
e-mail:seulgi.choi@ssu.ac.kr, mainevent34@ssu.ac.kr, jhsong@shinhan.ac.kr, mjun@ssu.ac.kr

A Design of Security Protocol Based on CoAP for Secure Message Transmission in DASH7 Environment

Seulgi Choi*, Kyung-Hwan Song*, Jin-Hee Song**, Moon-Seog Jun*
*Dept of Computer Science, Soongsil University
**School of IT Convergence Engineering, Shinhan University

요 약

IoT 환경에서는 자원 제약적인 디바이스가 보안이 취약한 무선 네트워크에 연결될 가능성이 존재한다. 초저전력 디바이스를 지원하는 무선 네트워크 표준 DASH7 Mode 2는 보안 분야 표준화가 진행 중이며 다양한 취약점에 대한 해결 방안을 모색 중이다. 따라서 본 논문에서는 DASH7 환경에서 IoT 표준 프로토콜인 CoAP를 이용하여 안전한 메시지 전송을 위한 프로토콜을 제안하였다. 또한 데이터 위변조, 재전송 및 가장 공격에 대한 안전성에 대하여 분석하였다.

1. 서론

IoT(Internet of Things)에서는 자원 제약적인 디바이스들이 DASH7(Developers Alliance for Standards Harmonization of ISO 18000-7), ZigBee, BLE(Bluetooth Low Energy), Z-WAVE, Insteon 등의 표준을 따르는 다양한 무선 네트워크에 연결되어 통신한다. 무선 네트워크는 도청, 데이터 위변조 등 보안 취약점이 존재하기 때문에 무선 네트워크 표준에서 보안을 고려하는 것은 필수적인 요소이다. 하지만 DASH7은 표준화가 진행 중이기 때문에 아직 보안 측면에서의 취약점에 대한 해결방안을 모색 중이다. 기존의 DASH7 Mode 1의 보안 프로토콜을 적용하는 방안이 있지만 DASH7 Mode 2가 시스템 구조, 전송 속도, 디바이스 등 여러 측면에서 DASH7 Mode 1과 다르기 때문에 환경의 변화에 따른 보안 기술을 적용하는데 어려움을 겪고 있다.

CoAP(The Constrained Application Protocol)는 DASH7처럼 자원 제약적인 디바이스를 지원하는 표준 통신 프로토콜이다. UDP 기반으로 가벼우면서 신뢰성을 보장하고 적은 메모리 자원을 고려하여 4byte의 축약된 헤더와 옵션을 가지므로 초저전력 디바이스를 지원하는 DASH7 환경에 적절하다[1].

따라서 본 논문은 IoT 환경을 고려한 초저전력 데이터 기술을 지원하는 무선 네트워크 표준인 DASH7 환경에서 안전한 메시지 전송을 위한 CoAP 기반의 보안 프로토콜을 제안한다.

본 논문은 다음과 같이 구성된다. 2장 관련 연구에서는 DASH7과 CoAP, 3장은 제안 프로토콜, 4장은 성능분석, 5장 결론은 향 후 연구 방향을 기술한다.

2. 관련 연구

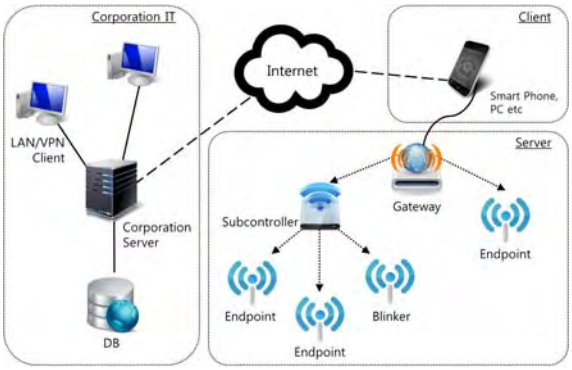
2.1. DASH7 Mode 2

DASH7은 ISO/IEC 18000-7의 상호로 433MHz 주파수 대역을 사용하는 초저전력 무선 데이터 기술이며 능동형 RFID의 대표적인 표준이다. 기존의 DASH7은 태그 ID 수집과 같은 저용량의 데이터를 전송하며 27.7kbps의 낮은 통신 속도를 지원한다. 또 태그 신호 전송 및 wake-up 전송 시간을 업체마다 달리하여 상호운영성에 문제를 가진다. DASH7 Alliance는 이를 보완하기 위하여 DASH7 Mode 2를 제안하였으며 이는 능동형 RFID 뿐만 아니라 IoT 환경도 고려하고 있다. 하지만 DASH7 Mode 2는 아직 ISO/IEC 18000-7의 제7절 Extended Mode 부분에 LR-WPAN(Low Rate Wireless Personal Area Network)으로 추가됐을 뿐 국제 표준 개정에서 아직 다루어지고 있지 않다[2][3].

(그림 1)은 DASH7 Mode 2의 시스템 구조를 나타낸다. DASH7 Mode 2에서는 서버 측에서 데이터를 수집하여 클라이언트와 웹을 통하여 외부 기업 서버로 이 데이터를 전송한다. 서버 측에서 데이터를 수집하기 위한 Endpoint는 RFID 태그, 센서 노드와 비슷한 역할을 하며 대부분 저전력(sleep) 상태에 있지만 wake-up 상태가 되면 RF 응답을 수행한다. 이와 반대로 클라이언트는 항상 wake-up 상태로 Gateway에게 전달 받은 Endpoint의 데

이터를 인터넷을 통하여 기업 서버(즉, 해당 데이터를 요청하고 관리하는 서버)로 전달한다[2].

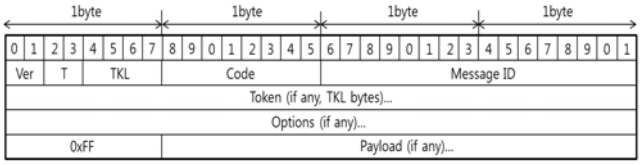
Gateway와 Subcontroller는 비슷한 역할을 하지만 Subcontroller는 클라이언트와 연결될 수 없고 저전력 동작 수행이 가능한 Endpoint와 연결되어 디바이스에 상태에 따라 저전력 동작을 수행하는 차이점이 있다[2].



(그림 1) DASH7 Mode 2 시스템 구조

2.2. CoAP

CoAP는 컴퓨팅 능력, 전력 소모 등에서 제한된 자원을 갖는 센서로 구성되는 IoT 환경에 적절한 통신 프로토콜이다. (그림 2)는 CoAP 프로토콜의 메시지 포맷을 나타낸다[5][6].

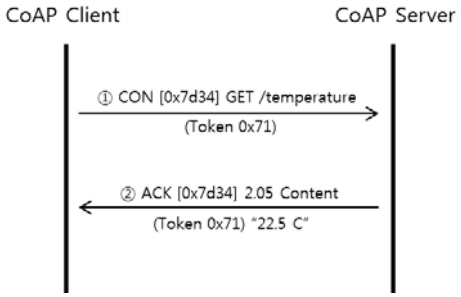


(그림 2) CoAP 프로토콜 메시지 포맷

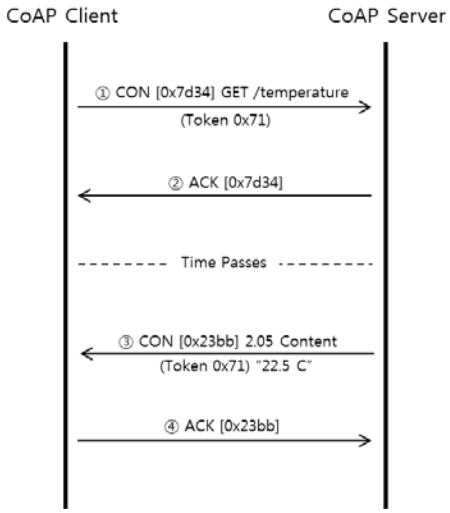
CoAP 프로토콜 헤더에는 버전(Ver), 타입(T), 토큰 길이(TKL), 코드(Code), 메시지 ID(Message ID)가 있다. 타입은 메시지 타입을 표현하는 것으로 Confirmable(0), Non-confirmable(1), Acknowledgement(2) 등으로 표현될 수 있다. 코드는 8비트의 값으로 앞에 3비트는 클래스(Class)를 나타내며 범위는 0~7로 각각 요청(0), 응답 성공(2), 클라이언트 에러 응답(4), 서버 에러 응답(5)를 나타낸다[1][4][5].

(그림 3)에서 CoAP 서버는 CoAP 클라이언트가 요청한 데이터 및 서비스를 ACK 메시지에 포함하여 전송한다. 이를 통해 프로토콜 처리와 메시지 수에 대한 오버헤드를 줄일 수 있다. 또 CON 메시지 또는 ACK 메시지를 전송할 때 '0x7d34'라는 메시지 ID 값을 사용한다. CoAP는 EXCHANGE_LIFETIME(기본 값 247초) 안에서 동일한 메시지 ID를 가지는 메시지를 전송할 수 없도록 하여 메시지 중복을 방지한다[1][5].

(그림 4)에서 CoAP 클라이언트가 CON 메시지를 전송했을 때 CoAP 서버가 슬립 상태이면 빈 메시지로 응답한다. 이것은 CoAP 클라이언트가 ACK_TIMEOUT(기본 값 2초)이 지나고 CON 메시지를 재전송하는 것을 방지하기 위함이다[1][5].



(그림 3) GET 요청에 따른 Piggy-Backed 응답



(그림 4) GET 요청에 따른 별도의 응답

3. CoAP 기반의 보안 프로토콜

3.1. 제안 프로토콜 개요

본 논문에서는 DASH7 환경에서 안전한 메시지 전송을 위한 CoAP 기반 보안 프로토콜은 제안한다. 제안 프로토콜은 다음을 가정으로 한다.

- 활동하는 모든 노드들은 ACK_TIMEOUT, EXCHANGE_LIFETIME의 기본 값을 공유한다.
- CS와 Endpoint는 DB에 SN_i, CID, DID, K_{cs-ep} 값을 공유한다.
- 각 노드는 현재 세션에서 수신된 모든 메시지를 DB에 저장한다.
- GW는 편의상 Smart Phone, PC 등 디바이스 역할을 한다고 가정한다.
- $STATE_{ep}$는 "SLEEP"과 "WAKE UP"의 값을 가진다.
- 재전송 프로토콜은 예외 상황이기 때문에 ACK_TIMEOUT, EXCHANGE_LIFETIME을 고려하지 않는다.

<표 1> 제안 프로토콜의 표기법

표기	설명
MID_{ep}, MID_{cs}	CS와 Endpoint가 전송할 CON과 ACK 메시지에서 사용되는 메시지 ID
K_{cs-ep}	CS와 Endpoint 사이에 공유된 비밀 키
SN_i	i 번째 세션 번호
CID, DID	CS와 Endpoint의 식별 번호
$REQTK_{ep}$	Endpoint에게 전송될 CON 메시지에 포함되는 토큰 값
$RequestInfo_{ep}$	CS가 Endpoint에게 요청하는 데이터
EXCHANGE_LIFETIME	CS가 CON 메시지를 전송하고 Endpoint의 ACK 메시지를 수신할 때까지의 시간
$RESTK_{ep}$	Endpoint의 ACK 메시지에 포함되는 토큰 값
$ResponseInfo_{ep}$	Endpoint가 CS에게 전송하려는 데이터
ACK_TIMEOUT	Endpoint가 ACK 메시지를 전송하고 CS가 수신할 때까지의 시간
$STATE_{ep}$	Endpoint의 상태

<표 2> 제안 프로토콜의 연산 과정

매개 변수	설명
MID_{ep}	DB에서 가져온 SN_i, CID, DID을 XOR 연산한 뒤, K_{cs-ep}로 암호화
$REQTK_{ep}$	$RequestInfo_{ep}$를 해시함
$RESTK_{ep}$	$RequestInfo_{ep}$와 $ResponseInfo_{ep}$를 연접한 후, 그 결과를 해시함
MID_{cs}	SN_{i+1}와 DID, CID를 XOR 연산한 뒤, K_{cs-ep}로 암호화

3.2. 제안 프로토콜의 세부 실행 과정

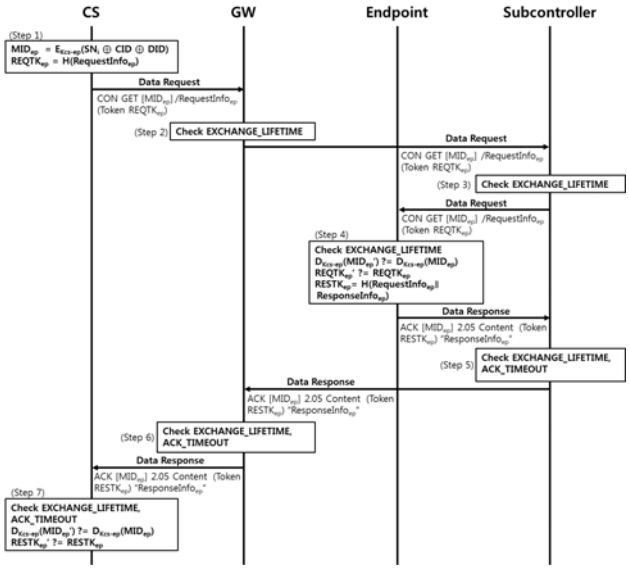
3.2.1. 메시지 전송 프로토콜

• Step 1(CS→GW): CS는 Endpoint로 전송할 CON 메시지의 MID_{ep}와 $REQTK_{ep}$를 생성한다. 먼저 MID_{ep}는 현재 세션에서만 유효한 세션 번호 SN_i을 사용하여 MID_{ep} 값의 탈취에 의한 재전송 공격을 방지한다.

그리고 MID_{ep}를 생성할 때 CS와 Endpoint가 안전하게 공유하고 있는 K_{cs-ep}, CID, DID 값을 사용하기 때문에 CS와 Endpoint가 상호 인증할 수 있도록 해준다.

$REQTK_{ep}$는 공격자가 CON 메시지를 탈취하여 CS가 요청하지 않은 $RequestInfo_{ep}$를 Endpoint로 전송하는 것을 방지하기 위한 $RequestInfo_{ep}$의 무결성 검증 값이다.

CON GET [MID_{ep}] /$RequestInfo_{ep}$ (Token $REQTK_{ep}$)



(그림 5) 메시지 전송 프로토콜

• Step 2(GW→Subcontroller): GW는 CON 메시지의 EXCHANGE_LIFETIME를 확인한다. 만약 이 시간 안에 GW의 DB에 저장된 수신 메시지 중 동일한 메시지 ID를 가지는 CON 메시지가 있다면 GW는 메시지 전송을 중단한다.

• Step 3(Subcontroller→Endpoint): Step 2와 동일한 과정을 수행한다.

• Step 4(Endpoint→Subcontroller): Endpoint는 MID_{ep}와 $REQTK_{ep}$를 생성하고 수신 값과 비교한다. 이 값이 같으면 Endpoint는 CS로 전송할 ACK 메시지에 포함되는 $RESTK_{ep}$를 생성한다.

$RESTK_{ep}$는 공격자가 ACK 메시지를 탈취하여 올바르지 않은 $ResponseInfo_{ep}$를 CS에게 전송하는 것을 방지하기 위한 $ResponseInfo_{ep}$의 무결성 검증 값이다.

또 CS가 $RESTK_{ep}$에 포함된 $RequestInfo_{ep}$를 통하여 합법적인 Endpoint로부터 ACK 메시지가 전송되었다는 것을 확인하면서 Endpoint의 인증도 수행할 수 있게 한다.

ACK [MID_{ep}] 2.05 Content (Token $RESTK_{ep}$)
"$ResponseInfo_{ep}$"

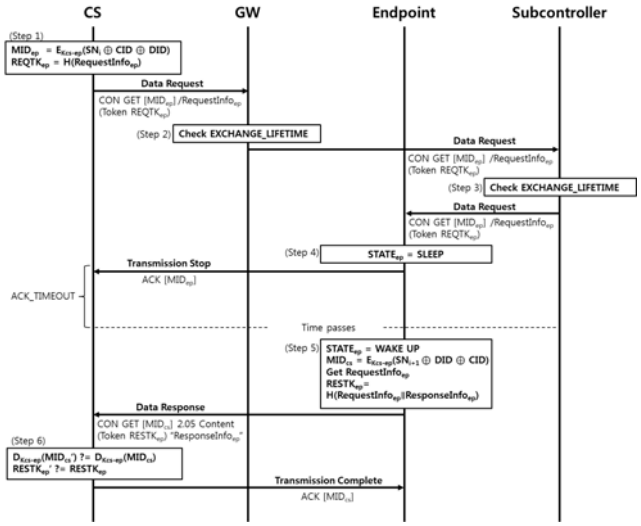
• Step 5(Subcontroller→GW): Subcontroller는 ACK 메시지의 EXCHANGE_LIFETIME과 ACK_TIMEOUT을 확인한다. 만약 ACK_TIMEOUT이 만료되었으면 Subcontroller는 메시지 전송을 중단한다.

• Step 6(GW→CS): Step 5와 동일한 과정을 수행한다.

• Step 7(CS): CS는 MID_{ep}와 $RESTK_{ep}$를 생성한

다. 이 값을 이용하여 수신한 MID_{ep}와 $RESTK_{ep}$를 검증한다.

3.2.2. 재전송 프로토콜



(그림 6) 재전송 프로토콜

• Step 1, 2, 3 : 이 과정은 (그림 5) 메시지 전송 프로토콜과 동일한 과정을 수행한다.

• Step 4(Endpoint→CS): SLEEP 상태인 Endpoint는 CS로 빈 ACK 메시지를 전송한다.

ACK [MID_{ep}]

• Step 5(Endpoint→CS): ACK_TIMEOUT이 지나고 일정한 시간이 흐르면 Endpoint와 Subcontroller는 다시 WAKE UP 상태가 된다. WAKE UP 상태인 Endpoint는 CS로 전송할 CON 메시지의 MID_{cs}와 $RESTK_{ep}$를 생성한다.

CON GET [MID_{cs}] 2.05 Content (Token $RESTK_{ep}$)
"$ResponseInfo_{ep}$"

• Step 6(CS→Endpoint): CS는 MID_{cs}와 $RESTK_{ep}$를 생성한다. 이 값을 수신한 MID_{cs}, $RESTK_{ep}$와 비교하여 검증이 완료되면 빈 ACK 메시지를 전송한다.

ACK [MID_{cs}]

4. 안전성 분석

4.1. 요청 및 응답 데이터 위변조

제안 프로토콜은 CON과 ACK 메시지를 수신한 CS와 Endpoint가 $REQTK_{ep}$와 $RESTK_{ep}$를 검증함으로써 공격자로부터 $RequestInfo_{ep}$와 $ResponseInfo_{ep}$가 변경되는 것을 방지할 수 있다.

4.2. 재전송 공격

제안 프로토콜은 각 노드에서 메시지 매개변수 EXCHANGE_LIFETIME을 확인함으로써 해당 시간 안에 동일한 메시지 ID를 가지는 메시지를 전송할 수 없다. 또 메시지 ID는 세션 번호를 기반으로 하여 생성하여 해당 세션에서만 유효하므로 외부 공격자의 재전송 공격에 안전하다.

4.3. 가장 공격

제안 프로토콜은 CS와 Endpoint가 사전에 DID, CID, K_{cs-ep}를 안전하게 공유하고 이 값들이 직접적으로 노출되는 구간이 없다. 따라서 공격자가 이 값을 이용하여 MID_{ep}와 MID_{cs}를 생성할 수 없어 가장 공격에 안전하다.

5. 결론

IoT 환경은 리소스, 배터리 자원 등에서 제약을 가지는 디바이스들이 무선 네트워크를 통하여 통신한다. 이러한 무선 네트워크에서의 통신은 여러 가지 보안 취약점이 있기 때문에 해결 방안은 필수적이다. 하지만 DASH7 Mode 2는 아직 보안 표준화 진행 단계로 보안에 대한 별다른 대책이 없는 실정이다.

따라서 본 논문에서는 IoT에서의 자원 제약적인 디바이스에 적합하며 신뢰성을 보장하는 통신 프로토콜 CoAP를 DASH7 환경에 적용하여 안전한 메시지 전송 프로토콜을 제안하였다. 안전성 분석을 통하여 제안 프로토콜이 요청 및 응답 데이터 위변조, 재전송 공격, 가장 공격에서 안전하다는 것을 확인하였다.

향 후 종단 간 신뢰된 통신 노드 검증을 위해 CS와 Endpoint 간의 상호 인증과 통신 환경 시뮬레이션을 통한 효율성 분석에 관한 연구를 진행할 예정이다.

참고문헌

[1] Z. Shelby, K. Hartke, C. Bormann, "The Constrained Application Protocol (CoAP)", IETF, RFC 7252, 2014.
[2] 황효성, "DASH7 기반 멀티홉 컨테이너 보안장치(CSD) 설계 및 구현", 부산대학교 석사학위논문, 2013.
[3] 이두원, "DASH7 Technology의 표준화 동향", 한국멀티미디어학회지, Vol. 14, No. 2, pp.27-32, 2013.
[4] 김문권, 김도현, "상호 호환성 검증을 위한 IoT 기반의 CoAP 프로토콜 구현 및 실험", 한국인터넷방송통신학회 논문지, Vol. 14, No. 4, pp.7-12, Aug. 31, 2014.
[5] 정민근, "ZigBee 네트워크 상에서 CoAP 사용을 위한 시스템 설계", 대구대학교 석사학위논문, 2014.