**IoT 하드웨어 보안 모듈을 이용한 secure FoTA 시스템 구축**김선우 박형모 이호재 조민제  
동국대학교 컴퓨터공학부  
swkim@dongguk.edu, gudah1478@dongguk.edu, yslhj93@gmail.com, godifwar@donggukl.edu   
**Designing Secure FoTA System with IoT Hardware Security Module**Sunwoo Kim Hyungmo Park Hojae Lee Minjei Cho

**요약**

전 세계적으로 Internet of Things(IoT)가 발전하고 있다. IoT device들의 배포가 늘어나면서 그에 대한 유지보수가 점점 복잡해지고 있다. 또한 IoT device들이 실생활에 적용되어 가면서 그에 대한 보안은 끊임없는 문제가 되어 가고 있다. 스마트폰 모바일 기기에는 유지 보수를 쉽게 하기 위해 FoTA(Firmware over the air)가 적용되어 있다. Open Mobile Alliance(OMA)에서는 FoTA에 대한 기준점을 제시하고 있으며, 그에 대한 Framework도 배포 중이다. 하지만 제시하고 있는 자료에는 보안성 수준이 낮으며 아직 미흡한 점이 많이 나타나고 있다. 본 논문에서는 OMA에서 제시한 FoTA 방식과는 다르게 하드웨어 보안 모듈 및 API를 이용하여 Secure FoTA를 제시한다. 데이터에 대한 handling/Recovery, Integrity 확인 등을 제안하며 Application security를 적용하여 다양한 네트워크 환경에서 보안성이 있는 펌웨어 업데이트를 할 수 있게 한다.

**1. 서 론**

현재 자동차에 무선 통신 기능을 탑재하여 외부 인프라와 연동을 통해 편의성을 향상시킨 Connected Car 기술이 주목을 받고 있다. 그러나, 이러한 기술에는 문제점이 존재한다. Jeep사의 Cherokee를 해킹하여 핸들, 브레이크를 원격으로 조종하는 것을 시연해 기술잡지에 기재되어 140만대의 차량이 리콜된 사례가 있다. 또한, 중국의 인터넷 기업 ‘텐센트’에서 테슬라 모델 S를 약 19km 떨어진 곳에서 원격으로 해킹해 차량 통제권을 완전히 탈취한 사례도 존재한다.

위와 같은 보안 사례들을 살펴보았을 때, Connected Car의 발전을 위해서는 다음과 같은 문제를 해결해야 한다. IoT device는 일반 컴퓨터와는 달리 하드웨어 성능이 낮으며 운영체제에서 지원하는 보안 기술을 사용하는데에 어려움이 있다. 그러므로 추가적인 보안 기술을 이용해 보안성을 유지를 해야한다. 본 논문에서 이용할 기술은 FoTA이며 이 기술을 통해서 펌웨어 업데이트를 실행할 것이다. 하지만 FoTA에서 보안에 관한 표준이 아직 자리를 잡고 있지 못하고 있다. 인터넷에 연결된 디바이스가 업데이트 도중, Hijacking의 위협은 존재하며 안전한 통신을 하기 위한 방안이 있어야 한다.

본 논문에서 소개할 내용은 IoT 하드웨어 보안 모듈을 이용한 FoTA이다. Device가 FoTA Server를 통해 업데이트를 하게 될 것이다. 이 때, 하드웨어 보안 모듈을 이용해 펌웨어 위변조 방지 기능을 수행하며 펌웨어 업데이트가 안정적으로 이루어질 수 있게 만들어 준다.

**2. 관련 연구**

Secure FoTA 관련 기술에 대한 Architecture는 Secure FoTA object for IoT[1].에 제시되었다. 이 논문에 제시하는 방식은 다음과 같다.

1. Update할 FoTA 데이터를 분할 후 CoAP(Constrained Application Protocol )[2]를 이용하여 전송한다.
2. FoTA로 전송할 데이터를 암호화한 FOSE object를 보내며 Client에서 매번 acknowledgement를 보내게된다.
3. Acknowledgement를 받아야 다음 파일을 전송하게 된다.
4. 보내는 파일의 size를 이용하여 받은지 확인을 한다.

Open Mobile Alliance - Device Management (OMA-DA)[3]에서 Firmware Update Management Object(FUMO) 표준화를 제시하고 있다. FUMO protocol은 PUSH, PULL mechanism을 이용하여 package data를 생성하며 업데이트를 하지만 따로 보안에 대한 처리를 해주고 있지 않다. 그리고 데이터에 대한 integrity 확인, handling/recovery 등 제시하고 있지 않으며 또한 분할을 이용한 업데이트에 대한 기준이 있지 않다.

최근에는 발전하고 있는 IoT 환경에 맞추기 위해 네트워크 보안보다는 어플리케이션 보안에 중점을 두고 있다. GSM Association, Open Mobile Terminal Platform(OMTP)에서는 모든 모바일 환경에서는 애플리케이션 보안을 우선 순위로 권장하고 있다[4].

Secure FoTA object에서 사용하고 있는 환경은 제한적인 부분이 존재한다. 먼저 CoAP의 환경에서 파일을 전송하고 있으며 Device에서는 매번 ack 신호를 보내고 있고 별도의 파일에 대한 위변조를 검사하고 있지 않다. 본 논문에서는 Device와 통신을 최소화하며 IoT device에 부담을 줄이고 오픈한 네트워크에서 사용해도 파일의 위변조 변경을 방지하기 위한 방법을 제시하며 OMA-DA에서 제안하고 있는 표준화는 원하는 수준이 아니므로 추가적으로 Hashing을 통한 integrity 확인, 데이터 전송에서 handling 및 오류시 error recovery의 방법을 제안한다.

**3장 시스템 구성**

**3.1 시스템 정의**

Secure FoTA 시스템은 Wi-Fi와 같은 무선 네트워크를 이용해 Cloud server와 연결하여 차량의 ECU 펌웨어를 업데이트를 수행하도록 한다. 여기서 Cloud server는 차량 제조사가 업로드한 업데이트 파일을 분할하고 순차적으로 차량에게 전송하는 역할을 수행한다. 그리고 차량의 ECU는 Wi-Fi를 이용해 Cloud server가 전송하는 분할된 파일들을 다운로드한 후, 통합하여 업데이트를 진행한다.

그림 1은 secure FoTA 시스템의 구성을 도식화한 것이다.

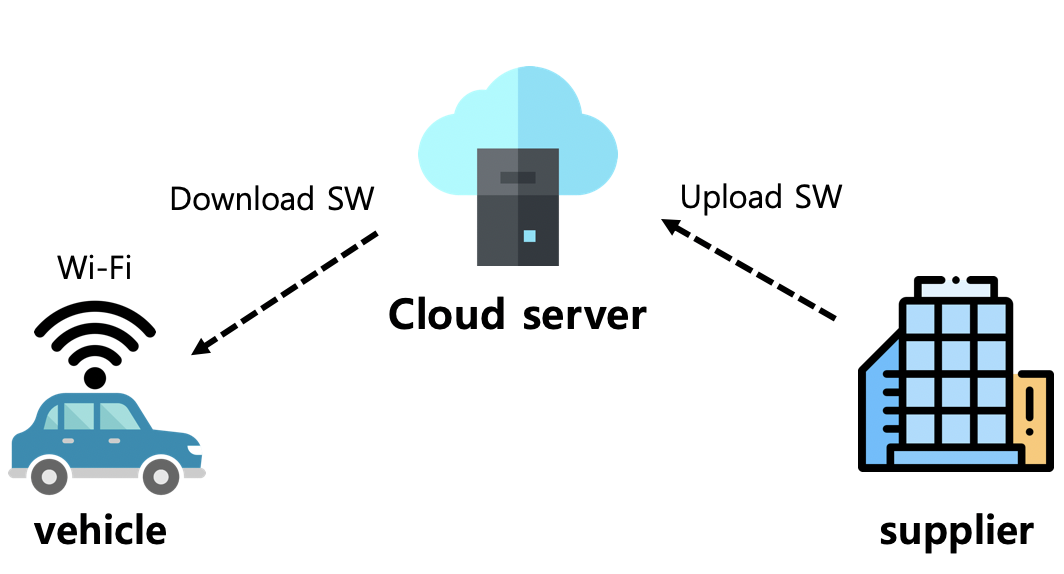


그림 1. secure FoTA 시스템 구성도

secure FoTA 시스템은 크게 3가지 구성요소가 존재한다.

- FoTA server

기기 및 계정 관리 기능. 펌웨어 버전을 최신으로 유지 하기위한 업로드 기능이 필요하다. 또한, 업데이트가 필요한 Device에게 최신 버전의 업데이트 파일을 전송하는 기능이 필요하다.

- Packager

제조사에서 업데이트 하고자 하는 업데이트 파일을 작은 크기의 단위로 분할하는 기능, 분할된 파일들을 암호화 하는 기능이 필요하다.

- Device(Unpacker)

무선 네트워크를 통해 FoTA Server와 통신하며 업데이트가 필요할 경우, 최신 업데이트 파일들을 다운로드 하고 복호화한 후 파일들을 합치는 기능이 필요하다.

**3.2 개발 환경**

- Raspberry Pi

차량의 ECU 역할을 수행하며 OS는 Raspbian을 사용한다.

- Cloud Server(AWS)

Amazon 사에서 제공하는 클라우드 컴퓨팅 서비스로서, 본 논문에서는 메인 서버를 AWS에서 구축할 것이다. OS는 ubuntu를 사용한다.

-Linux

대표적인 오픈 소스 C 프로그래밍 운영체제이다. ECU에서 사용하는 Raspbian과 서버에서 사용하는 Ubuntu 모두 Linux 기반 운영체제이다.

-C++

절차지향 프로그래밍 언어인 C언어에 객체지향 프로그래밍을 지원하기 위해 만들어졌다. ECU에서 구동할 Device 프로그램 구현 시 사용할 것이다.

-Apache

Web server의 한 종류로써, 본 논문에서는 Apache를 이용해서 제조사가 업데이트 파일을 업로드하고 Device 프로그램이 다운로드 할 수 있는 메인 서버를 구현할 것이다.

- CIOT HUSSM GPIO module

Raspberry Pi 호환 하드웨어 보안 모듈로써 다양한 대칭키 알고리즘 및 암호화 알고리즘을 지원한다. Device에서 보안 기술을 사용할 경우, 이 모듈을 통해 하드웨어적으로 보안 처리를 수행한다. 그림 2는 모듈의 그림을 나타낸다.

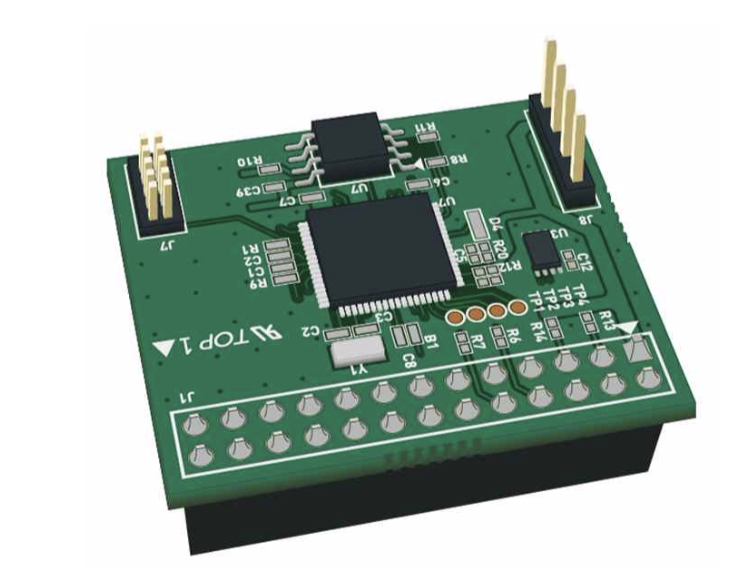


그림 2. CIOT HUSSM GPIO의 그림

**3.3 하드웨어 보안 모듈**

본 논문에서 사용할 보안 모듈은 CIOT HUSSM GPIO module이다. Raspberry Pi와 같은 많은 IoT device들은 GPIO를 제공하고 있으며 본 논문에서 사용할 보안 모듈은 40-pin GPIO 장착이 가능하다. 또한, 별도의 프로세서(Cortex-M0) 및 메모리를 갖고 있다. 그렇기 때문에 Device에 보안 적용 시, 부담을 최소화 할 수 있다.

대부분의 IoT device는 32bit 이하의 운영체제를 갖고 있다. IoT-developer-survey-2019[5]에서는 Linux가 대부분을 차지하고 있다. Embedded Linux중에서도 Ubuntu, Raspbian, Debian 순으로 나타나고 그 이하의 운영체제가 존재하지 않는 IoT device가 전체의 11%를 차지하고 있다. 이처럼 많은 IoT device들은 운영체제의 bit가 작으므로 보안성이 떨어지게 되며 현재 사용하고 있는 보안 알고리즘을 사용할 수 없게 된다. 여기서 하드웨어 모듈을 사용하게 되면 Cross-platform뿐만 아니라 32-bit 운영체제가 지원하는 보안 알고리즘을 사용이 가능해진다.

**3.4 암호 알고리즘**

TCP 통신을 할 경우, 일반적으로 네트워크 통신 과정에서 데이터 오류를 checksum 기능을 이용해 보호한다.[6] 그러나, 해커가 위변조 펌웨어를 전송하게 되면 네트워크 통신 과정에 문제가 없다고 가정하고 디바이스는 위변조 사실을 알 수 없다. 이러한 문제를 해결하기 위해 본 논문에서는 블록 암호인 ARIA 암호화 알고리즘과 안전한 키를 분배하기 위해 ECC 공개 키 알고리즘을 사용하였고, 데이터가 서버에서 보낸 데이터와 일치한 지 확인하기 위해 해시 알고리즘 SHA-256을 사용하였다.

ARIA : 한국에서 개발된 블록 암호 알고리즘으로 128,192,256비트를 지원하며 경량 환경 및 하드웨어에 최적화 되어있어 이 알고리즘을 채택하였다.[7]

ECC 공개키 알고리즘 : 타원곡선 암호에 기반한 공개키 알고리즘으로 일반적으로 사용하는 RSA보다 더 짧은 비트로 비슷한 수준의 안정성을 제공한다.[8] 표 1을 보면 ECC 160비트가 RSA 1024비트와 비슷한 안전도를 제공하는 것을 알 수 있다. 컴퓨팅 파워에 제한이 있는 IoT 환경에서 상대적으로 더 작은 비트를 사용하는 ECC 방식을 채택하였다.

SHA-256 : 해시 알고리즘의 한 종류로써 MD5, SHA-1 와 비교했을 때, 더 좋은 성능을 보이므로 본 논문에서는 SHA-256을 채택하였다.

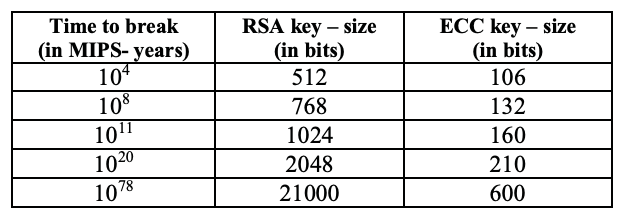


표 1. RSA와 ECC의 키 길이 별 안전도 비교

secure FoTA에서 업데이트 파일을 암호화하는 과정은 그림 3과 같다.

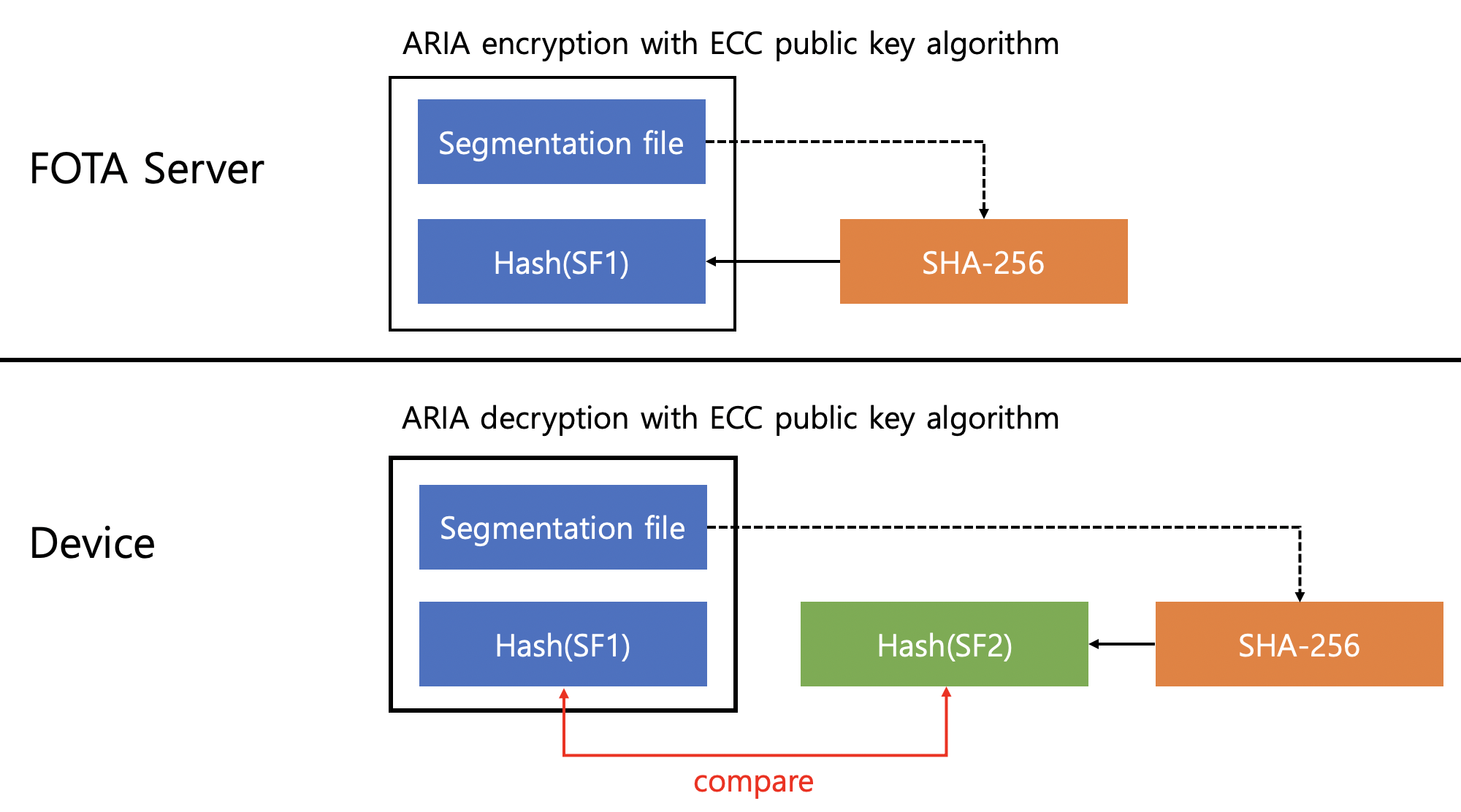
****

그림 3. 보안 알고리즘 블록 다이어그램

FoTA server에서 분할된 파일을 SHA-256을 이용해 해시 값을 만들고 파일과 해시 값을 ARIA를 이용해 암호화한다. Device는 전송받은 암호화 된 데이터를 복호화하여 분할된 파일과 해시 값을 받고, 분할된 파일을 Device 내에서 한번 더 해시 값을 구하고 FoTA server에서 구한 해시 값과 비교하고 일치하면 안전한 데이터를 받았다고 판단하고, 그렇지 않으면 재전송 한다.

**3.5 통신 프로토콜**

분할, 암호화된 파일들을 안전하게 통신하기 위해서는 특수한 프로토콜이 필요하다. 그림 4는 본 논문에서 제시하는 데이터 통신 프로토콜을 나타내는 시퀀스 다이어그램이다.

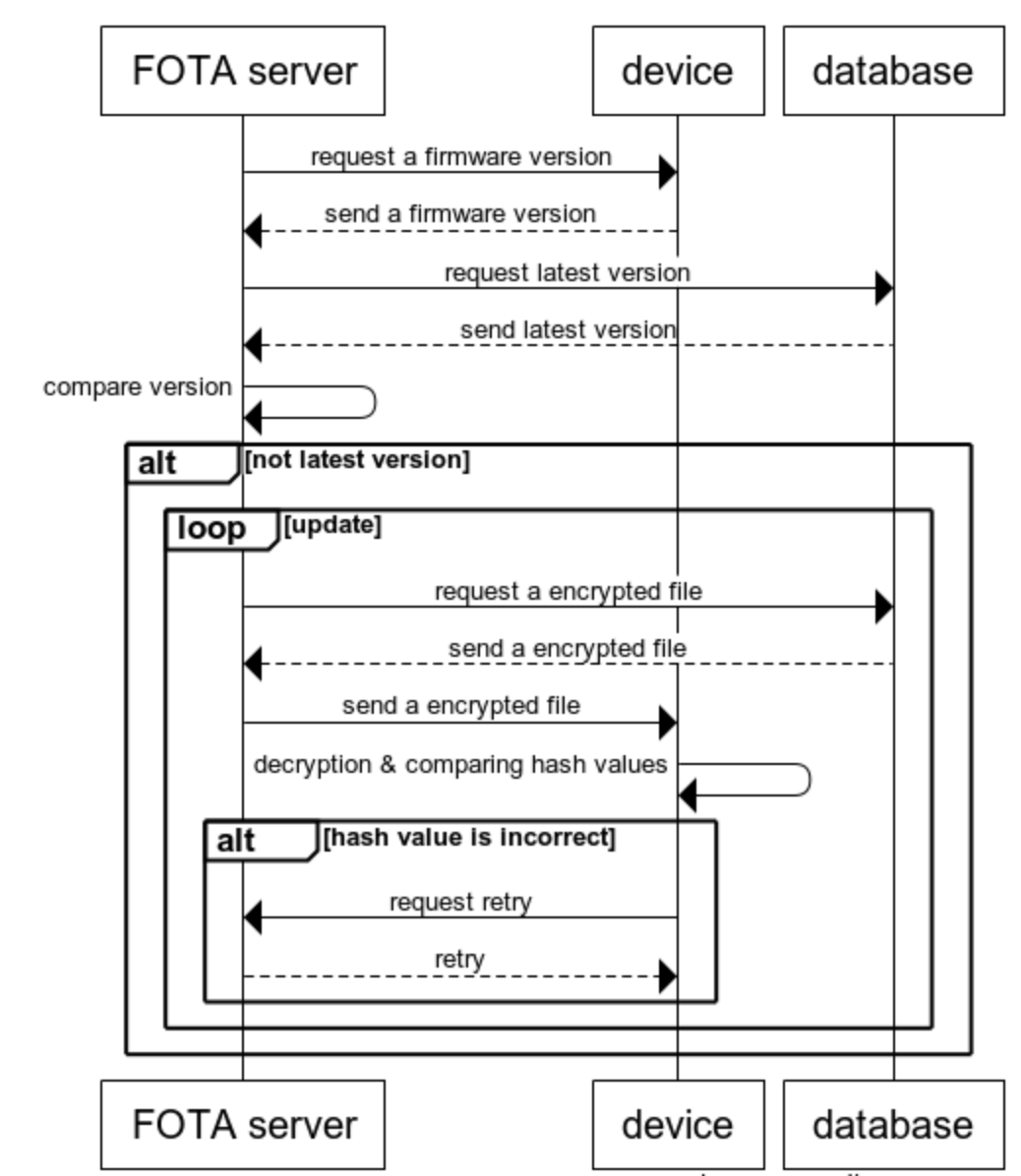


그림 4. 데이터 통신 프로토콜

우선, FoTA server가 Device에게 펌웨어 버전을 요청하면 Device가 펌웨어 버전을 전송한다. FoTA server는 최신 버전과 비교하고 만약, 최신 버전이 아니라면 위와 같은 방식으로 통신을 진행한다. 암호화된 파일들을 순차적으로 전송한 후, Device가 복호화된 파일을 통해 hash value를 구하고 FoTA server에서 전송한 hash value과 비교한다. 만약 일치하면 다음 파일을 전송한다. 일치하지 않으면 해당 파일을 재전송한다.

**4장 결론**

본 논문에서 제시한 시스템을 이용해 무선 네트워크 환경에서도 IoT Device의 업데이트를 안전하게 수행할 수 있을 것이다. IoT Device의 늘어나는 숫자와 발전할 가능성을 봤을 때, 앞으로 해결할 문제가 많다. FoTA에 관한 표준화가 아직 확실히 잡혀져 있지 않아 최적화 에러 처리와 같은 문제가 존재한다. IoT Device의 환경에서, Wi-Fi 처럼 빠른 네트워크를 사용하지 못할 수 있으므로 그에 맞는 handling이 필요하다. 이러한 문제점들을 해결하기 위해 위와 같은 시스템을 제안한다. 본 논문에서 사용하는 암호화 알고리즘 SHA-256은 현재까지는 word list brute force attack이 아니면 공격이 힘들다. 또한, 하나의 비트의 차이로 인해 Hashing 값이 달라지게 되므로 파일에 대한 integrity 확인이 가능하다. 이처럼 OMA-DA에서 지정하지 않는 점들을 개선해주며 IoT Device의 컴퓨팅 한계로 나타나는 보안성 문제들을 극복할 수 있다.

지금까지는 단순히 웹 서버를 Cloud computing 자원을 사용하는 것에 그치기 때문에 다수의 Device에 효율적으로 서비스를 제공하기 위해 docker 컨테이너로 서버를 구현할 필요성이 있다. 또한, 최근 connected Car의 통신 방식은 Wi-Fi가 아닌 WPLAN 기술들을 사용한다. 그러므로 추세에 맞춰 cat.m1과 같은 LPWAN 환경에서도 본 논문의 시스템을 적용해야 한다.

**참고 문헌**

[1] : Krishna Doddapaneni∗, Ravi Lakkundi†, Suhas Rao†, Sujay Gururaj Kulkarni†, Bhargav Bhat†, “Secure FoTA object for IoT”, in 2017 IEEE 42nd Conference on Local Computer Networks Workshops

in 978-1-4799-4774-4/14/$31.00 ©2014 IEEE

[2] : C. B. Z. Shelby, K. Hartke, “The Constrained Application Protocol (CoAP),” RFC 7252, 2014. [Online]. Available: https://rfc-editor.org/ rfc/rfc7252.txt

[3](2017, March) Open mobile alliance, device management. http:// openmobilealliance.org/iot/.

[4]OMTP Specs (2019, March, 8) <http://www.omtp.org/OMTP_Application_Security_Framework_v2_2.pdf>

[5]<https://iot.eclipse.org/resources/iot-developer-survey/iot-developer-survey-2019.pdf>

[6] : Jon Postel, “TRANSMISSION CONTROL PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION”, RFC 793

[7] : J.Lee, J.Lee, J.Kim, D.Kwon, C.Kim, “A Description of the ARIA Encryption Algorithm”, RFC 5794

[8] : Ohood Althobaiti, Hatim Aboalsamh, “An Enhanced Elliptic Curve Cryptography for Biometric”, on Computing and Convergence Technology(ICCCT), 2012 7th International Conference