# DASH 7 보안 기술

이동건* · 김호원*

## 1. 서 론

IT 839 전략이 시작되었던 2004년부터 최근까지 국내에서는 RFID와 센서 네트워크, 그리고 USN(Ubiquitous Sensor Network) 기술에 대해 산업체와 학계, 연구소에서 활발한 연구 개발을 수행해왔다. 최근 국내에서는 이들 분야에 대한 연구 활력이 다소 떨어진 것처럼 보이지만 여전히 이 분야는 산업체에서 중요한 이슈인 것만은 확실한 것 같다.

이 분야에서 특히 많은 응용 시장을 확보하고 있는 기술로는 수동형 RFID 기술(UHF RFID 기술, EPC Class 1 Gen2, ISO 18000-6)과 ZigBee 기술이 있다. 이 기술은 각각 RFID 태그와 WPAN(Wireless Personal Area Network) 분야에서 어느 정도 시장을 확보하면서 안정적인 성장을 하는 것으로 보이지만, 여러 가지 단점을 가지는 것도 사실이다. 예를 들어, 수동형 RFID 태그는 리더에서 수신된 RF 에너지에 기반하여 동작하므로 기능 및 메모리 용량에 있어서 태생적 한계를 가질 수 밖에 없으며, ZigBee 기술(특히 IEEE 802.15.4)은 컨테이너와 같은 응용 환경에서는 통신의 안정성 및 신뢰성에 있어서 단점을 가지는 것으로 알려져 있다. 컨테이너 환경처럼 많은 통신 방해 요소를 가지는 물류 응용 환경에서는 RFID의 기능적 장점과 ZigBee의 장거리 통신/네트워킹 장점을 갖는 기술이 필요하다. 능동형 RFID 기술이 이와 같은 두 가지 특성을 모두 가지는 것으로 볼 수 있다. 최근 능동형 RFID 기술 분야에서는 기존의 ISO18000-7을 개선한 DASH 7에 대한 표준화 및 기술 개발이 활발히 진행되고 있다. DASH 7에서는 넓은 coverage와 저전력 특성, 높은 신뢰성 등 여러 가지 좋은 특성을 가지지만 특히 기존의 RFID 기술에서 취약한 특성인 보안 특성이 좋다는 장점도 있다.

DASH 7의 보안 특성은 표준에서 RFID 태그와 리더(Interrogator) 사기의 상호 인증(mutual authentication) 기법 암호화된 데이터 전송 기법, 프레임 암호화 기법, 데이터 접근 기법, 보안 서비스로 정의 된다. 현재 각 보안 특성에 따라 구체적인 표준화 작업이 진행되고 있다. 특히 상호 인증 기법과 암호화된 데이터 통신 기법, 프레임 암호화 기법은 암호 알고리즘과 보안 프로토콜 등이 정해진 상황이다. 이에 본 논문에서는 DASH 7의 주요 보안 기술을 살펴보고 이에 대한 개발 사례를 살펴보기로 한다.

※ 교신저자(Corresponding Author) : 김호원, 주소 : 부산광역시 금정구 장전동 산30번지 부산대학교 정보컴퓨터공학부, 전화: 051-510-1010, FAX: 051-517-2431, E-mail : howonkim@gmail.com
* 부산대학교 정보컴퓨터공학부

## 2. DASH 7 보안 기술 표준

### 2.1 DASH 7 상호 인증 프로토콜 개요

DASH 7 보안 기술 중에서 상호 인증 기법과 암호화된 데이터 통신 기법, 프레임 암호화 기법은 구체적으로 정의 되어 있다. 아래 그림은 DASH 7 표준에서 정의된 태그와 리더(interrogator) 사이의 상호 인증 프로토콜을 보여주고 있다. 먼저 Message 1은 리더가 태그를 인증하기 위해 Challenge 값을 태그에 전송한다. 태그는 이에 대한 응답으로 Response 값을 interrogator에 전송하는 것과 더불어, 태그가 interrogator를 인증하기 위해서 Challenge 값을 전송한다(Message 2). 여기서 Message1을 Secure challenge start 메시지라고 하며 Message 2를 Secure challenge end 메시지로 볼 수 있다. Interrogator는 이에 대한 응답을 태그에 전송해야 한다. Tag는 Interrogator에게 응답 값을 받았다는 ACK 메시지를 보낸다(Message 4). 그림 1의 프로토콜은 두 개체가 상호 인증을 위한 전형적인 프로토콜로서Interrogator는 태그를 인증하고 태그는 Interrogator를 인증하는 상호 인증을 위한 것이다. 참고로 태그와 Interrogator가 전송하는 Challenge 값은 공격자에 의한 재생 공격(Replay attack)이 일어나는 것을 막기 위해 난수(random number) 특성을 가지는 nonce 값을 사용해야 한다.

또한 표준에 의하면 Message 1과 Message 2는 태그 혹은 Interrogator에 의해 시작될 수 있다고 기술되어 있다. 즉, 이는 태그와 Interrogator 누구도 먼저 상대 entity 인증을 위한 절차를 시작할 수 있음을 의미한다. 또한, Message 1,2,3는 모두 추가 정보를 포함할 수 있어서 추가 정보 전송, 보안성 강화를 위한 추가 조치(예를 들어, time stamp 값 추가 등)를 할 수 있다.


그림 1 DASH 7 태그와 리더간의 상호 인증 프로토콜

안전하게 키 값을 분배하기 위해선 그림 1에서 정의된 태그와 Interrogator 간에 상호 인증을 수행한 후, Interrogator가 암호화 된 키 값을 태그에 전송한다.


그림 2 태그 키 값 업데이트 프로토콜

아래 메시지 형식은 상호 인증을 위해 태그와 Interrogator간에 전송되는 메시지 형식으로서 Challenge 값이나 Key 인식 정보, 초기 설정 값 등 다양한 정보를 전송할 때 사용되는 메시지 형식이다.


그림 3 태그와 Interrogator 간의 메시지 전송 형식

### 2.2 DASH 7 보안 프로토콜

DASH 7에서는 AES 대칭키 암호 알고리즘이나 SHA, HummingBird 알고리즘을 사용하여 상

호 인증 프로토콜을 실현 할 수 있으며 이에 대한 방법이 표준에서 기술되어 있다. 아래 그림은 AES 대칭키 암호 알고리즘과 SHA 해쉬 함수 기반 사전 배분 키 값을 기반으로 상호 인증을 수행하는 인증 프로토콜을 보여주고 있다.

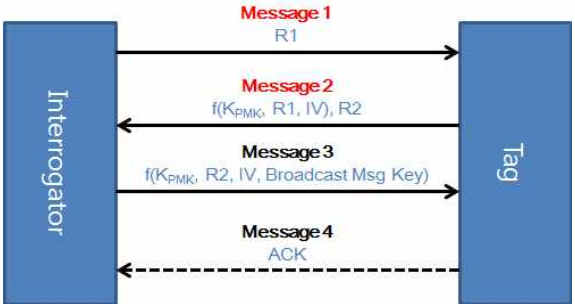


그림 4 사전 배분 키 값 기반 상호 인증 프로토콜

먼저 Interrogator는 난수 R1 값(Message 1)을 태그에 전송하면 Tag는 사전에 상호 배분된 128 비트 마스터 키 값을 사용하여 자신이 수신한 난수 R1 값과 초기 값(IV)을 암호화하여 응답한다. 이때 태그가 생성한 난수 R2 값을 Interrogator에 전송한다. R2 값은 태그가 Interrogator를 인증하기 위한 값이다. Interrogator는 태그와 사전 배분/공유된 키 값인 $K_{PMK}$값을 갖고 있기 때문에 태그로부터 자신(Interrogator)이 보낸 R1 값이 $K_{PMK}$값으로 암호화되어 응답이 정상적으로 왔음을 확인함으로서 태그가 정당한 태그임을 확인할 수 있다.

참고로 태그로부터 Interrogator에 전송된 태그 값은 16 바이트(128 비트) 값으로서 R1 값과 사전 분배 마스터 키 값($K_{PMK}$), R2의 길이와 동일하다.

위 프로토콜에서 Broadcast Msg Key는 Ks 값으로 암호화된다(Message 3). 즉, Interrogator가 태그에 전송하는 Message 3은 Ks로 암호화된 메시지 값을 다른 정보(R2, IV)와 함께 $K_{PMK}$로 암호화하여 태그로 전송함을 알 수 있다. 여기서 세션 키 값 $K_S$는 다음과 같이 유도된다.

$$K_S = SHA\text{-}1(R1 \parallel R2 \parallel K_{PMK})$$

다음 그림은 Rotor 기반 암호 알고리즘인 HummingBird 암호 알고리즘을 사용하여 상호 인증을 수행하는 모습을 보여주고 있다. Interrogator는 태그에 상호 인증을 위한 시작 신호만 전송하면 태그는 Interrogator에 사전 분배된 키 값으로 자신이 원하는 IV값과 이 값을 $K_{PMK}$로 암호화 한 값을 보낸다(Message 2). Interrogator 측에서는 태그가 자신과 사전 분배되어 있는 $K_{PMK}$로 암호화 되어 있는지를 확인함으로서 태그를 인증할 수 있다.

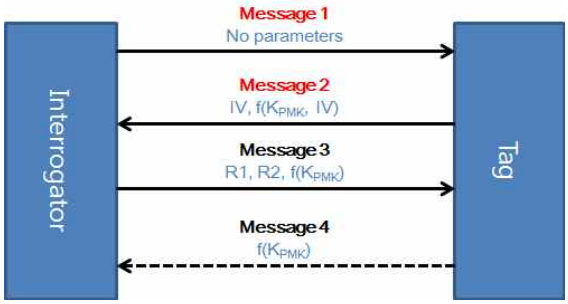


그림 5 사전 배분 키 값 및 HummingBird 기반 상호 인증 프로토콜

Message 3은 Interrogator가 자신을 태그에 인증하기 위한 것이다. 그리고 Interrogator가 태그에 전송하는 R1 값과 R2 값은 태그내의 HummingBird 암호 알고리즘에서 사용하는 난수 값이다. 한편, DASH 7 보안 표준에서는 사전 분배된 키 값을 사용하여 다수의 태그를 상호 인증하는 프로토콜도 정의 되어 있다. 이때 기존의 ISO/IEC 18000-7:2009 통신 프로토콜과 호환성을 가지도록 정의 된다.

DASH 7 보안 표준에서는 공개키 암호 알고리즘

을 사용한 상호 인증 프로토콜도 정의 되어 있다. 비록 mandatory는 아니지만 공개키 암호를 사용할 경우 키 업데이트 등, 키 관리가 쉬워지며 기존의 PKI(Public Key Infrastructure)를 사용할 수 있기 때문에 많은 장점을 가진다. 특히, 자원 제약성이 높은 태그에서는 타원곡선 암호 알고리즘(Elliptic Curve Cryptosystem)을 사용할 수 있기 때문에 많은 장점(빠른 연산 속도 짧은 키 길이)을 가진다. 아래 그림은 Interrogator의 개인키 값으로 서명한 메시지를 태그에 전송하는 Message 1과 이에 대한 응답 Message 2를 보여주는 그림이다. Message 1에는 Interrogator가 자신의 인증서($R_{certificate}$)을 태그에 전송함을 알 수 있다. Interrogator가 자신의 인증서를 태그에 전송하는 이유는 인증서 내부의 자신의 공개키 값이 들어있기 때문이다. 태그는 수신한 Interrogator의 공개키 값을 사용하여 Diffie-Hellman 식 키 분배나 Interrogator가 서명한 값을 검증할 수 있다.


그림 6 태그와 Interrogator 사이의 인증서 교환 프로토콜

Message 2는 태그 자신의 인증서($T_{certificate}$)를 자신의 개인키 값으로 서명하여 Interrogator에 전송하는 모습을 보여주고 있다. 즉, Interrogator는 태그의 공개키를 서명되어 수신함을 의미한다.

다음 그림은 공개키 암호를 사용하여 태그와 Interrogator사이의 상호 인증 프로토콜을 보여주고 있다. 먼저 Message 1은 Interrogator가 태그에 난수 R1값을 Interrogator가 서명하여 보내는 메시지이다. 태그는 이에 대한 응답으로서 자신이 생성한 난수 R2 값과 수신한 R1값을 Interrogator의 공개키로 암호화한다(Message 2).

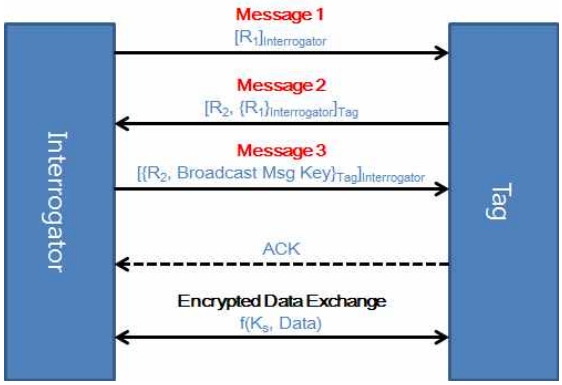


그림 7 공개키 암호 알고리즘을 사용한 상호 인증 프로토콜

Message 2의 값이 Tag의 개인키 값으로 서명되었기 때문에 Interrogator는 자신이 통신하고자하는 특정 태그에 해당함을 확신할 수 있다. Interrogator는 Message3을 통해 태그에게 자신을 인증함과 동시에 브로드캐스트 키 값을 태그에 전송할 수도 있다.

## 3. DASH7 RFID 태그 보안 기술

### 3.1 DASH7 RFID용 암호 기술

DASH7 표준에서는 Interrogator와 Tag 사이의 상호 인증을 위해 대칭키 암호 방식을 사용하는 3가지 상호 인증 방법과 공개키 암호 방식을 사용하는 2가지 상호 인증 방법을 정의하고 있다. 대칭키 암호 방식에는 AES, DES, 그리고 HummingBird 알고리즘을 이용한 방식이 정의되어 있으며, 공개키 암호 방식에는 RSA와 ECC 방식을 사용한 방법이 정의되어 있다. 이외에 추가

적으로 SHA 알고리즘이 인증 과정에서 해쉬 생성을 위해 사용된다.

또한 패킷의 보호를 위해 기밀성을 위한 암호와 기법과 무결성을 보장하기 위한 메시지 인증 기법이 정의되고 있다. 표준에서는 암호화를 지원 하지 않는 모드를 포함하여 총 5가지의 대칭키 방식의 Protection Suite을 정의하고 있는데, 여기에서 정의된 암호화에 사용되는 대칭키 알고리즘으로 AES와 HummingBird가 있다. 이 중 AES의 경우 CBC모드와 Counter 모드를 사용하도록 정의하고 있다. 메시지 인증에는 SHA 알고리즘을 사용하는 방법과 AES-CBC-MAC을 사용하는 방법, 그리고 HummingBird 알고리즘에서 제공하는 MAC 생성 기법을 사용하는 방법 등을 정의하고 있다. 공개키 방식의 경우 RSA와 ECC를 정의하고 있다.

### 3.2 DASH7 암호 모듈 사례

이번 절에서는 DASH7을 대상으로 연구된 암호 모듈 구현 사례를 살펴보고자 한다. DASH7 표준에서 mandatory 로 요구하는 AES 알고리즘의 경우 2001년 표준으로 제정된 알고리즘으로써 십여 년간 소프트웨어/하드웨어로의 구현에 관한 연구가 많이 진행되어 있다. 특히 최근에는 경량의 마이크로 프로세서에도 AES 가속 하드웨어를 패키지에 포함하는 사례가 늘고 있다. TI에서는 능동형 DASH7 Tag에 사용될 수 있는 RF 모듈을 포함한 프로세서에 AES 암/복호 가속 하드웨어를 탑재하기도 하였다[2]. 하지만, AES-CBC-MAC이나 Counter 모드와 같은 Mode of Operation을 가속 하드웨어로 제공하고 있지 않기에, 이를 한정된 마이크로프로세서 자원으로 효율적으로 구현하고자 하는 시도도 있었다[3].

공개키 방식의 경우 RSA와 ECC 두 가지 방법을 표준에서 제시하고 있지만, 실 구현에서는 RSA와 ECC 모두 연산에 오랜 시간이 소모되며, Tag의 경우 특성상 저렴해야 함에도 불구하고, 공개키 방식이 요구하는 프로그램에 필요한 메모리의 용량이나, 키를 저장하기 위한 공간 등을 만족시키기가 어려울 가능성이 많다. 하지만 ECC의 경우 비슷한 레벨의 안전성을 제공함에도 RSA에 비해서 키 길이가 짧으며, 최근에는 프로그램 사이즈 역시 마이크로 컨트롤러에 구현될 수 있을 만큼 작게 하는 기법들이 많이 연구되고 있기 때문에, 공개키 방식이 사용된다면, RSA 보다는 ECC가 사용될 가능성이 더 높다.

ECC의 경우 DASH7의 자원 제약 조건과 가장 유사한 환경인 802.15.4 기반의 센서네트워크를 기반으로 ECC 및 관련 프로토콜을 구현한 사례인 TinyECC가 있다[4]. 해당 연구에서는 SECG(Standards for Efficient Cryptography Group)에서 정의한 128, 160, 192 bit 파라메터를 가지는 타원 곡선과 ECDSA, ECDH등의 프로토콜을 구현하였다. ECC 구현시 Sliding Window 기법이나 Projective 좌표계를 사용함으로써 성능 향상을 꾀하였으며, 서명 검증시에도 Shamir Trick을 사용하여, 연산 속도를 향상 하고자 했다. 이러한 모든 최적화 기법을 사용할 경우, ECDSA 서명과 검증 과정에서 가장 보편적으로 사용되는 TMote 플랫폼을 사용할 경우 2초 이상의 시간이 소요되며, 모든 최적화 기법을 사용하지 않을 경우 서명은 20초, 검증은 40초 이상의 시간이 소요된다. ECDSA 서명의 경우 최적화 기법을 사용하지 않을 시 8KByte 이상의 ROM과 160Byte의 RAM을, ECDH의 경우 7KByte에 육박하는 ROM과 158Byte의 RAM을 요구한다. 최적화 기법 적용시 메모리 요구량은 더 늘어나게 되는데,

ECDSA의 경우 13KByte 이상의 ROM과 1.5KByte의 RAM, ECDH의 경우 11KByte 이상의 ROM과 1.8KByte의 RAM을 요구한다. 자원 제약적인 DASH7 Tag의 조건을 고려하였을 때, ECDSA 기법이 소프트웨어로 구현될 경우 본래 Tag 가 수행해야 하는 소프트웨어를 구현하는데 사용될 메모리 공간에 제약이 더 커질 수가 있다. 프로그램 공간과 수행 시간을 고려하였을 때, ECC 알고리즘 및 ECDSA와 같은 알고리즘을 별도의 하드웨어로 구현하는 방법을 고려해볼 수 있다.

SECG에서는 ECC 파라메터를 정의함에 있어 소수를 기반으로 $GF(p)$ 상에서 정의되는 곡선과 이진체를 기반으로 $GF(2^m)$ 상에서 정의되는 곡선을 제공하고 있다[5]. 일반적으로 하드웨어 구현의 경우 $GF(p)$보다는 $GF(2^m)$ 상에서 구현하는 것이 더 효율적인 것으로 알려져 있다. 표준에서는 1024비트의 RSA와 유사한 안전도를 제공하는 160비트 파라메터의 ECC 곡선을 사용할 것을 권고하고 있으며, 따라서 ECC가 하드웨어로 구현된다면, sect163r1및 sect163r2를 구현하는 것이 가장 현실적이 될 것으로 고려된다.

ECC 알고리즘의 주요 연산인 스칼라곱 연산의 경우 반복적인 좌표 더블링 연산과 덧셈 연산으로 구성된다. 두 연산에서는 이진체의 덧셈, 곱셈, 제곱, 역승산 연산등이 필요한데, 이들 연산은 $GF(2^m)$ 상에서 구현될 때 매우 간단하게 구현될 수 있다. $GF(2^m)$ 연산의 경우 덧셈 연산이 각 비트간의 XOR로 간단하여 비트 캐리에 의해 지연 시간이 길어지는 정수 연산에 비해 로직이 단순하고, 많은 수의 비트를 동시에 연산하여 병렬성이 높다는 점에서 장점을 가진다. 또한 리덕션 연산 역시 특정 비트에 대한 XOR 연산으로 간단하게 처리되기 때문에, 곱셈 및 제곱 연산 역시 비교적 단순하게 구현될 수 있다.

ECDSA 알고리즘은 ECC의 스칼라 곱, 포인트 덧셈 연산을 포함하며, 추가적으로 다정도 정수 연산을 포함한다. 이는 ECDSA 연산의 경우 ECC 파라메터 중 베이스 포인트의 order를 이용해 $GF(p)$ 다정도 정수 연산을 수행해야하기 때문이다. ECDSA 서명 및 검증 기법은 FIPS-186-3[6]에 정의되어 있으며, 사용되는 연산은 덧셈, 뺄셈, 곱셈, 역승산 연산 등이다. ECC를 제외하고 ECDSA를 수행하는 부분을 소프트웨어로 구현하더라도 다정도 연산을 구현하는 데 상당한 메모리를 필요로 한다. 따라서, ECDSA 연산 역시 하드웨어로 수행하는 것을 생각해 볼 수 있다.

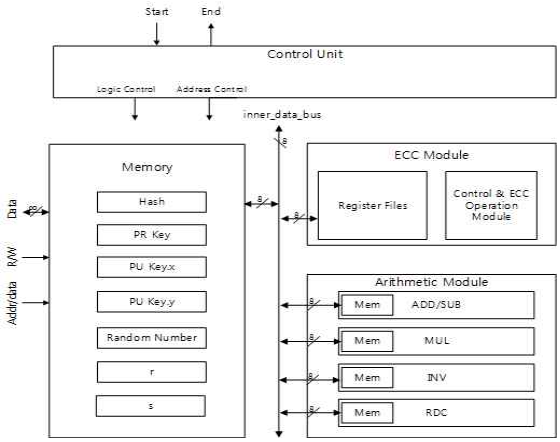


그림 8 ECDSA 하드웨어(FPGA) 구조

그림 8은 ECSDA를 FPGA로 구현한 사례를 보여주고 있다. 사례에서는 Xilinx Spantan 6 계열의 FPGA칩에 ECDSA를 구현하였다. 구현에서는 sect163r1을 구현한 ECC 모듈을 포함하고 있으며, 추가적으로 ECDSA 연산을 위해서 필요한 입출력을 위한 메모리와 다정도 정수 연산을 수행하기 위해 연산 모듈을 포함하고 있다. 상기 구현에서는 ECC를 일반적인 방법으로 Slice를 이용해 구현하고, 최소의 추가 Slice를 이용해 다정도 연

산을 구현하기 위해서 사용하지 않은 Block RAM을 이용해 다정도 연산을 구현하였다. 이는 입출력 및 연산에 필요한 Flip-flop(Slice Register)의 수를 줄이고자 함이며, Slice 수가 적은 FPGA 칩에서도 구현 가능하도록 하기 위함이다.

입출력을 위한 메모리의 경우 서명의 대상이 되는 메시지의 해쉬값을 저장하기 위한 공간과, 서명에 사용되는 개인키를 저장하기 위한 공간, 서명 검증에 사용되는 공개키의 x와 y좌표를 저장하기 위한 공간, 서명 생성시 사용되는 Random Number를 저장하기 위한 공간, 계산된 서명값을 출력하거나, 서명 검증을 위해 서명 값을 입력 받기 위한 공간인 r과 s의 메모리로 구성되어 있다. 각 메모리들은 Dual Port Memory로 구성되어 있어, 외부 메모리 인터페이스로 연결되지 않는 포트의 경우 내부 데이터 버스에 연결되어 ECC 연산 모듈이나 연산 로직에 있는 메모리와 데이터를 주고받는다. 메시지에 대한 해쉬값의 경우 내부에 해쉬 함수를 구현하지 않았는데, 이는 해쉬 입력이 되는 임의의 길이의 메시지가 마이크로프로세서로부터 FPGA로 전달되어야 하는데, 이로 인해 오히려 성능 저하가 있을 수 있으며, 일반적으로 해쉬 알고리즘의 경우 소프트웨어로 구현시 비교적 쉽게 구현이 가능하기 때문이다.

연산 모듈 역시 Dual Port RAM을 통해 구현하였다. 한쪽 포트는 내부 데이터 버스를 통해 연산의 입력이나 결과 값을 전달하고, 다른쪽 포트는 연산에 필요한 조합 회로로 연결되어 연산이 수행되는데 사용되도록 하였다. 연산의 입출력 및 임시 값 저장에 필요한 저장 공간을 Block RAM을 통해 구현하였으며, 8비트 단위의 연산을 수행함으로써 연산 수행 시간은 오래 걸리지만, 조합 회로 구성에 최소의 Slice를 이용해 구현 가능하도록 하였다. 상기 구현시 10MHz로 칩이 동작할 경우 서명은 20ms, 서명 검증은 41ms의 시간이 소요된다. 합성시에는 2,780개의 Slice Register와 7,068개의 Slice LUT, 그리고 5개의 RAMB16BWER와 32개의 RAM8BWER을 이용하여 구현 가능하다.

## 4. 결 론

본 논문에서는 기존의 ZigBee 기술이나 수동형 RFID 태그 기술의 단점을 보완하여 물류 응용 분야에서 많은 장점을가지는 DASH 7의 보안 표준 및 구현 사례를 살펴봤다. DASH 7 기술은 AES나 HummingBird와 같은 경량 대칭키 암호 알고리즘 뿐만 아니라 타원곡선 암호 알고리즘과 같은 공개키 암호 알고리즘 기반으로 구체적인 보안 프로토콜을 제시하고 있다. 이로서, 상호 인증, 서명/검증, 메시지 암호화 통신이 모두 가능하다. 또한, 향후 표준에서는 메모리 보호 기능, 키 보안 관련 내용이 추가될 것으로 보인다. 이러한 표준으로 인해 DASH 7은 더욱 폭넓은 응용이 가능할 것으로 보인다.

## 참 고 문 헌

[1] DASH7 Security WG, “ISO/IEC 18000-7 Security Services Extension,“ 2011.

[2] Texas Instrument, “CC430 Family User’s Guide,” http://www.ti.com/lit/ug/slau259e/slau259e.pdf, 2009

[3] Hwajeong Seo, Howon Kim, “Network and Data Link Layer Security for DASH7,“ Journal of information and communication convergence engineering, vol. 10, no. 3, pp. 248-252 , 2012

[4] An Liu, Peng Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in

Wireless Sensor Networks," in Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), SPOTS Track, pp. 245-256, April 2008.

[5] SECG, "SEC 2: Recommended Elliptic Curve Domain Parameters version 2.0," 2000.

[6] NIST, "FIPS-186-3, Digital Signature Standard(DSS)," 2009.

이 동 건

• 2009년 부산대학교, 정보컴퓨터공학과, 공학학사
• 2011년 부산대학교, 컴퓨터공학과, 공학석사
• 현 재 부산대학교, 컴퓨터공학과 박사과정 재학중
• 관심분야: 정보보호 기술, IoT기술, 암호 하드웨어 설계/구현

김 호 원

• 1993년 2월 경북대학교 전자공학과 공학학사
• 1995년 2월 포항공과대학교 전자전기공학과 공학석사
• 1999년 2월 포항공과대학교 전자전기공학과 공학박사
• 2003년 7월~2004년 6월 독일 Ruhr University Bochum Post Doc
• 1998년 12월~2008년 2월 한국전자통신연구원(ETRI) 정보보호연구단 선임연구원/팀장
• 현 재 부산대학교 정보컴퓨터공학부 교수
• 관심분야: IoT 기술, 데이터마이닝, 지능형 시스템 RFID/USN 정보보호 기술