편집위원 : 정수환(숭실대)

IoT 통신 환경을 위한 경량 암호 기술 동향

문시훈, 김민우, 권태경
연세대학교 정보대학원

요 약

IoT 통신 환경이 구축되면서 고사양 기기뿐만 아니라 저사양 기기를 사용하는 통신도 함께 증가하고 있다. 안전한 통신을 위해서는 메시지 암호화와 인증을 함께 제공하는 블록 암호 기술이 요구된다. 하지만, 기존 블록 암호 기술을 통신, 계산 기능이 제약된 저사양 기기에 그대로 사용하기에는 어려움이 따른다. 따라서 다양한 경량 암호 기술이 등장하게 되었다. 본 논문에서는 경량 암호 기술의 동향에 대해서 살펴보고 직접 IoT 실험 기기인 8비트 아두이노, 16비트 티모트, 32비트 라즈베리 파이2를 이용하여 구현 실험한 성능 측정 결과에 대해서 논한다.

Ⅰ. 서 론

IoT 통신 환경이 구축되면서 고사양 기기뿐만 아니라 저사양 기기를 사용하는 통신이 증가하면서 그에 따른 보안 위협도 증가하고 있다. 휴레 펫커드 사가 2015년 발표한 자료에 따르면 약 70%의 IoT 기기들이 보안 분야에서 취약점을 갖고 있다[1]. 통신 환경에서 침입자는 가장 약한 링크를 찾아내 손쉽게 침투하려 한다. 그러므로 저성능에서 고사양까지의 모든 IoT 기기들이 일정 수준의 보안성을 갖춰야 한다. 보안성을 갖추기 위해서는 메시지 암호화와 인증을 제공하는 블록 암호 알고리즘과 메시지 인증 코드가 필수적으로 사용되어야 한다. 하지만, 기존 암호 기술을 통신, 계산 기능이 제약된 저사양 기기에 그대로 사용하기에는 어려움이 따른다. 따라서, 다양한 경량 암호 기술이 등장하게 되었다.

최근 LEA, SIMOM, SIMON, SPECK 등 새로운 경량 블록 암호 알고리즘이 지속적으로 등장하고 있으며, 이를 응용하기 위한 연구가 진행되고 있다. 국내에서 Park 등은 OpenSSL 환경에서 경량 블록 암호인 LEA를 적용하는 방법을 연구하였으며[2], Ha 등은 암호 알고리즘 평가 시스템을 설계하여 암호 알고리즘의 성능 평가를 수행하였다[3]. 또한, Bae 등은 저성능 IoT 기기를 위한 저전력 통신 환경을 제안하였다[4].

본 논문에서는 최근 제안되는 경량 블록 암호 알고리즘과 대표적인 메시지 인증 코드의 특징에 대해 정리한다. 이와 더불어 각 알고리즘을 실제 IoT 실험 기기에 직접 실행을 통해 성능 측정 결과를 제시한다.

Ⅱ. 경량 블록 암호 알고리즘

IoT 환경에서 블록 암호는 메시지 암호화 및 인증을 위해 사용된다. 하지만 성능이 제약적인 기기에서 효율적인 사용을 위해서는 경량화가 요구된다. 이에 안전성을 유지한 채 암호화 및 인증이 가능하도록 개선된 경량 블록 암호들이 등장하게 되었다. 〈그림 1〉은 블록 암호의 개발 현황을 보여준다.

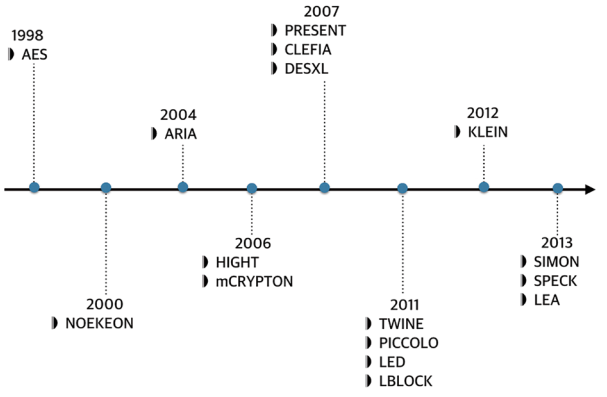

그림 1. 블록 암호 알고리즘 개발 현황

1. AES

Advanced Encryption Standard (AES)[5]는 NIST에 의해 표준화되어 가장 널리 쓰이는 블록암호이다. 128비트 블록 크기를 바탕으로 128, 192, 256비트의 키 길이의 사용이 가능하다. 또한 SPN구조로 되어있으며, 라운드 함수 내에 subbyte, shiftrows, mixcolumns, addroundkey과정을 통해 8비트 단위의 연산을 통한 암/복호화를 한다.

2. NOEKEON

NOEKEON[6]은 Daemen 등이 제안하였으며, NESSIE 프로젝트에 2000년에 제출되었다. 128비트의 블록 크기와 키 길이, 16번의 라운드를 통해 암/복호화를 한다. 각 라운드에서는 하드웨어, 소프트웨어로 용이하게 구현 가능한 self-inverse변환을 이용한다.

3. ARIA

ARIA[7]는 국내에서 개발한 블록 암호 체계로 학계, 연구소, 정부기관이 공동으로 개발을 하였다. SPN구조이며, 128비트 블록 크기를 가진다. 128, 192, 256비트의 키 길이가 사용 가능하며, 대부분의 연산은 XOR과 같은 단순한 바이트 연산을 통해 암/복호화를 한다.

4. HIGHT

HIGHT[8]는 2005년 국내에서 공동 개발한 블록 암호다. RFID, USN 등과 같이 저전력, 경량화를 요구하는 컴퓨팅 환경에서 기밀성을 제공한다. 8비트 단위 산술연산만으로 구성되며, AES보다 간단한 알고리즘 구조로 설계되었다. 64비트 블록 크기를 바탕으로 128비트 키 길이를 통해 암/복호화를 한다.

5. mCRYPTON

mCRYPTON[9]은 Lim 등이 제안한 Crypton 암호화 방식의 경량화 버전이다. 64비트 블록 크기를 바탕으로 64, 96, 128비트 키 길이를 사용한다. SPN 구조로 총 12라운드로 이루어지며, 라운드 함수 내에 sbox, bitpermutation, column-to-row transposition, subkey addtion을 통해 암/복호화를 한다.

6. PRESENT

PRESENT[10]는 Bodganov 등이 제안한 AES를 기반으로 하는 블록 암호다. 64비트 블록 크기를 바탕으로 80비트 키 길이를 사용하며, AES에 비해 암호화 강도는 떨어지지만 면적과 소비전력을 개선시켜 RFID, smart card, USN과 같은 시스템에 최적화시켰다. 32라운드로 구성되며 sbox, XOR, bitshift 연산을 통해 암/복호화를 한다.

7. CLEFIA

CLEFIA[11]는 SONY에서 발표한 AES와 유사한 블록 암호다. 128비트 블록 크기를 바탕으로 128, 192, 256 키 길이를 사용한다. Feistel 구조로 설계되었고, 키 길이에 따라 라운드 수가 달라진다. 기본적으로 평문을 4개의 32비트로 나누어 연산하며, 내부 함수에는 2종류의 sbox와 데이터 처리부와 키 스케줄링을 통해 암/복호화를 한다.

8. DESXL

DESXL[12]은 Leander 등이 제안하였으며, 기존 DES의 sbox를 새롭게 설계하여 경량화한 DESL에 선형분석과 차분석 공격 등에 대항하기 위한 keywhitening 기법을 추가하였다. 64비트 블록 크기를 바탕으로 196비트 키 길이를 사용하여 암/복호화를 한다.

9. TWINE

TWINE[13]은 Suzaki 등이 제안하였으며, Feistel 구조를 적용하였다. 64비트 블록 크기를 바탕으로 80, 128비트 키 길이를 사용한다. 36라운드로 구성되었고, XOR 연산을 이용하여 암/복호화를 한다.

10. PICCOLO

PICCOLO[14]는 Shibutani 등이 제안하였으며, Feistel 구조로 무선 센서 네트워크 환경에 적합하도록 설계되었다. 64비트 블록 크기를 바탕으로 80, 128비트 키 길이를 가지며, 키 길이에 따라 25, 31라운드를 수행하며 암/복호화를 한다.

11. LBLOCK

LBLOCK[15]은 Wu 등이 제안하였으며, Feistel 구조를 적용하였다. 64비트 블록 크기를 바탕으로 80비트 키 길이를 사용한다. 32라운드를 수행하며 라운드 함수 내에서 4비트 크기의 워드로 나누어 permutation을 사용하여 암/복호화를 한다.

12. KLEIN

KLEIN[16]은 Gong 등이 제안하였으며, SPN 구조를 적용하였다. 성능이 제약적인 무선 센서와 RFID에 적합하도록 설계되어 소프트웨어, 하드웨어에서 컴팩트한 구현이 가능하도록 하였다. 64비트 블록 크기를 바탕으로 64, 80, 96비트 키 길이를 가지며, 키 길이에 따라 12, 16, 20라운드르 수행하며, sbox, rotation, mixcolumns를 통해 암/복호화를 한다.

13. SIMON, SPECK

SIMON[17], SPECK[17]은 Beaulieu 등이 제안하였으며 NSA를 통해 발표되었다. Feistel 구조를 적용하였으며, SIMON은 하드웨어, SPECK은 소프트웨어 측면에 초점을 맞춰 설계되었다. 다양한 블록 크기, 키 길이, 이에 따른 라운드 연산이 가능하다. SIMON은 XOR, AND, rotation 연산을 사용하고, SPECK은 XOR, addition, rotation 연산을 통해 암/복호화를 한다.

14. LEA

LEA[18]는 Hong 등이 제안하였으며, Feistel 구조를 적용하였다. 현재까지 알려진 모든 블록 암호에 대한 공격에 대해 안전하며, 키 스케줄 특성에 기인한 이론적 취약성이 존재하지 않는다. 128비트 블록 크기를 바탕으로 128, 192, 256비트 키 길이를 사용할 수 있다. 라운드 함수 내에서 4개의 32비트 나누어 addition, rotation, XOR 연산을 사용하여 암/복호화를 한다.

Ⅲ. 메시지 인증 코드

메시지 인증 코드(Message Authentication Code, MAC)는 메시지의 무결성과 함께 이에 대한 인증을 보장하는 보안 기법이다. 메시지와 인증 태그를 함께 전송하여 전송 중 메시지 내용이 부당하게 변경되지 않고 안전하게 수신되었음을 검증한다. 해시 함수와 유사하게 임의의 길이의 입력 값을 고정된 길이의 출력 값으로 압축한다. 하지만, 메시지 인증 코드에서는 연산 과정 중 사전 공유된 비밀키를 사용한다는 차이점이 있다. 이러한 특징으로 메시지 인증 코드는 Keyed Hash Function이라고도 불린다[19]. 인증 태그 값을 비교하는 단순한 연산이지만 인증 태그의 안전성과 효율성을 높이기 위한 다양한 변종 메시지 인증 코드가 꾸준히 제안되고 있다.

메시지 인증 코드는 기반 기술을 기준으로 3가지로 분류된다. 블록 암호, 해시 함수, 유니버셜 해싱으로 각 분류마다 명확한 특징을 가진다. 블록 암호 기반은 짧은 키를 이용하여 오버헤드가 작다. 해시 함수 기반은 큰 블록 크기로 낮은 충돌 확률을 갖지만, 오버헤드 발생 위험이 있다. 유니버셜 해싱은 타 기술에 비해 충돌 확률이 낮다.

기반 기술별로 다양한 알고리즘이 존재하지만, 통신 프로토콜에서는 표준으로 채택된 기반 기술별 대표 알고리즘이 사용된다. 현재 통신 프로토콜에서 사용 중인 알고리즘은 블록 암호 기반은 CMAC[20], 해시 함수 기반은 HMAC[21], 유니버셜 해싱 기반은 Poly1305[22]가 있다. 모두 RFC로 채택되었으며, CMAC은 NIST[23], HMAC은 FIPS[24] 표준이다. 따라서, 본 논문에서는 통신 프로토콜에서 사용되는 대표 알고리즘의 특징에 대해 정리한다.

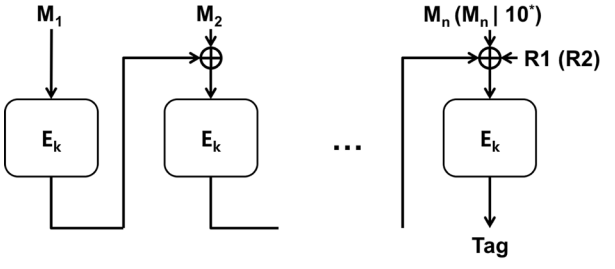

그림 2. CMAC 생성 과정

1. CMAC

CMAC은 Iwata와 Kurosawa가 제안한 블록 암호 기반 메시지 인증 코드다[26]. 잘 알려진 메시지 인증 코드인 CBC-MAC의 단점을 개선한 알고리즘이다. 고정된 길이에서만 안전성이 보장되던 문제점을 키 연산 과정을 추가하여 해결하였다. 인증 태그 생성 과정은 CBC-MAC 연산 과정과 동일하지만, 마지막 블록 연산 시 사용되는 키가 메시지 패딩 여부에 따라 달라진다. 자세한 연산 과정은 〈그림 2〉와 같다. R1은 패딩을 수행하지 않는 경우이며, R2는 패딩 수행 시 사용되는 키이다.

2. HMAC

HMAC은 Bellare 등이 제안한 해시 함수 기반 메시지 인증 코드다[27]. 암호학적 해시 함수를 이용하여 메시지 위조, 키 복구 공격에 저항성을 가진다. 하나의 키를 패딩 문자 ipad와 opad를 이용하여 두 개의 키로 확장하여 사용한다. 인증 태그 생성 과정은 다음 수식과 같다.

$$\mathbf{HMAC(x) = H(k \oplus opad, H(k \oplus ipad, x))} \quad (1)$$

다양한 암호학적 해시 함수를 변경없이 사용할 수 있어 확장성이 좋다.

3. Poly1305

Bernstein이 제안한 Poly1305는 블록 암호를 사용하는 유니버셜 해싱 기반 메시지 인증 코드다[28]. 블록 암호의 키 스케줄링을 통해 비밀키를 확장하여 키의 안전성을 향상시킨다. 또한, 각 메시지마다 Nonce 포함하여 메시지 위조를 더욱 어렵게 만든다. 이로인해 위조 공격 시 공격자는 2^{64}개의 메시지를 도청한 후 2^{75}번의 위조 시도가 필요하다. 블록 암호에 기반한 안전성을 가져 사용하는 블록 암호가 깨지기 전까지 안전성을 보장받는다.

Ⅳ. 성능 평가

본 절에서는 블록 암호의 성능을 IoT 환경에서 널리 사용되는 실험용 기기인 8비트 아두이노 우노[26], 16비트 티모트[27], 32비트 라즈베리 파이2[28]를 통해 실험한 결과를 제시한다. 〈표 1〉은 실험에 이용되는 기기의 명세를 나타낸다.

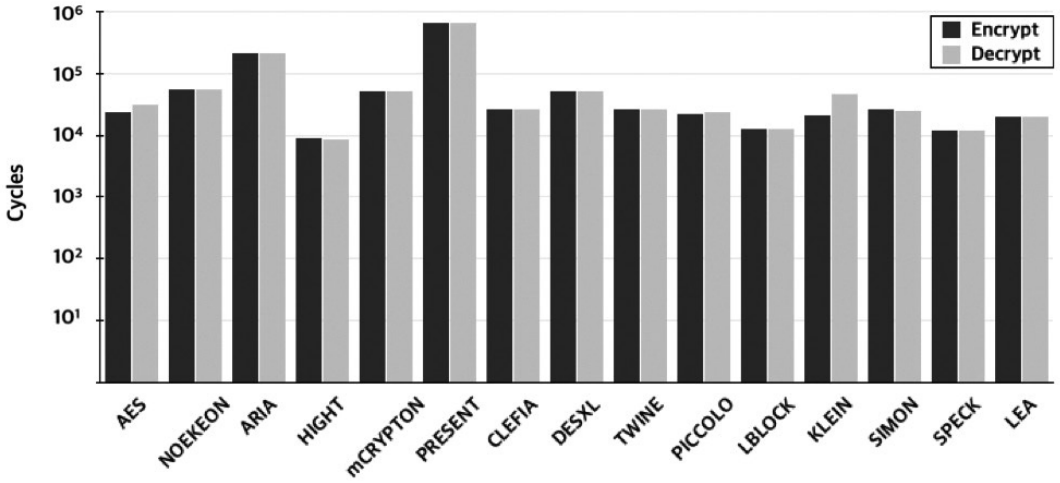

그림 3. 아두이노 환경 성능 측정 결과 (로그 스케일)

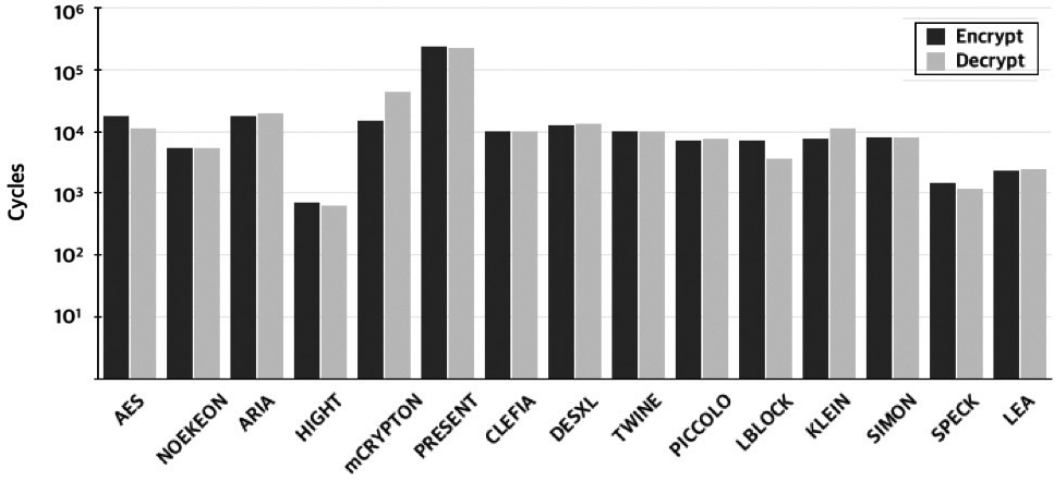

그림 4. 티모트 환경 성능 측정 결과 (로그 스케일)

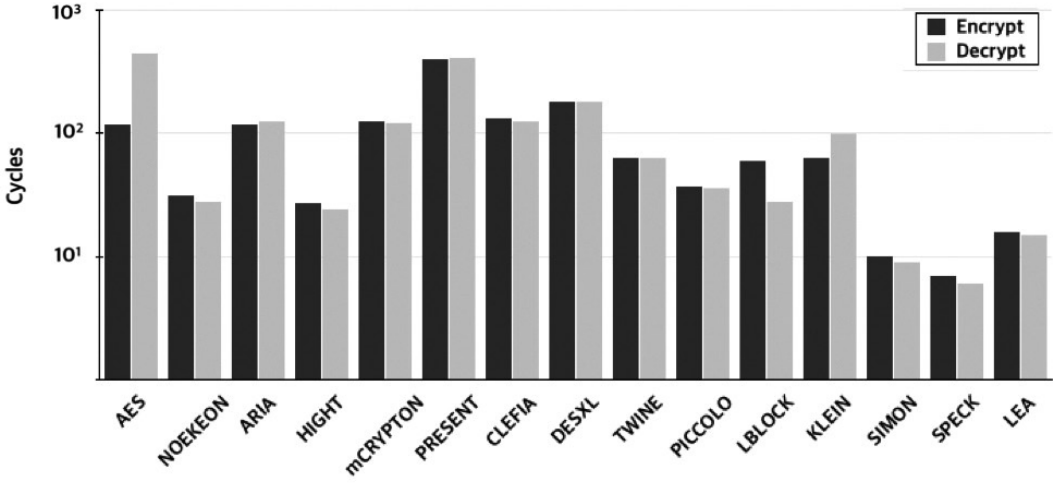

그림 5. 라즈베리 파이2 환경 성능 측정 결과 (로그 스케일)

표 1. 기기 명세서

	Arduino	Tmote	RP2
MCU	AVR ATmega328	MSP MSP430f1611	ARM Cortex-A7
CPU	8-bit RISC	16-bit RICS	32-bit RISC
Frequncy	16MHz	8Mhz	900Mhz
I/O	14 pins	16 pins	HDMI, USB
GPU	–	–	Videocore IV
MEMORY	32KB Flash 2KB SRAM	48KB Flash 10KB SRAM	1GB DRAM (LPDDR2)
EEPROM	4	–	–
Storage	External SD	1024KB	Micro SD

1. 실험 환경

우리는 앞서 언급한 실험 기기에 C로 구현한 경량 블록 암호를 실행하여 Cycles 성능을 측정하였다. 아두이노 환경에서는 기본으로 제공해주는 스케치 IDE를 사용했고, 티모트 환경에서는 RIOT 운영체제를 올린 후 실행을 통해 측정하였다. 마지막으로 라즈베리 파이2는 라즈비안 운영체제를 올린 후 리눅스 환경에서 실행을 통해 측정을 하였다.

2. 성능 측정

가. 경량 블록 암호 알고리즘 성능 측정

〈그림 3〉,〈그림 4〉,〈그림 5〉는 블록 암호의 성능을 아두이노, 티모트, 라즈베리 파이2에서 암/복호화 성능을 측정한 결과를 보여준다. 측정 결과를 보면 경량 암호의 성능은 기기의 성능이 높아질수록 향상되는 것을 볼 수 있다. 대표적 블록 암호인 AES는 8비트 단위의 연산 처리를 하기 때문에 저성능의 아두이노에서는 비교적 좋은 성능을 보였지만, 기기 성능이 향상됨에 따라 성능이 저하되는 모습을 보였다.

반면에 SIMON, SPECK, LEA는 전체적으로 가장 우수한 성능을 보여주었다. 그 이유를 살펴보면, SIMON은 하드웨어 성능에 초점을 맞춰 설계된 만큼 고성능의 기기로 갈수록 성능이 우수해졌다고 볼 수 있다. SPECK은 소프트웨어에 초점을 맞춰 설계되어 계산 연산량의 성능이 우수해지는 고성능 기기에서 저성능에 비해 우수해졌다고 볼 수 있다. LEA는 32비트 환경에 최적화하여 설계된 블록 암호이기 때문에, 고성능의 기기에서 타 환경에 비해 월등한 성능을 보여주었다고 볼 수 있다. 또한 이 세가지 경량 암호의 공통점은 ARX의 간단한 연산만을 사용하기 때문에 라운드 함수 내에서 보다 빠른 연산 속도를 보여주었다고 할 수 있다.

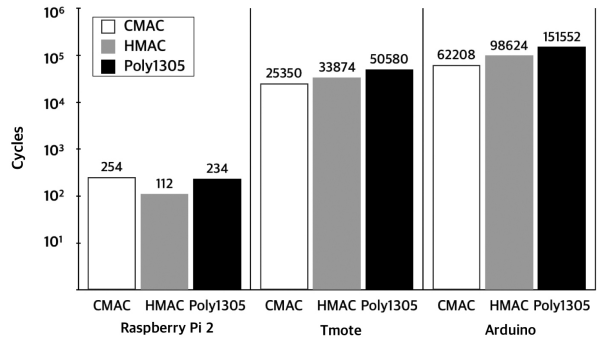

그림 6. MAC 성능 측정 결과 (로그 스케일)

나. MAC 성능 측정

〈그림 6〉은 기기별 MAC 성능을 나타낸 그래프다. 성능 측정 결과 기기의 성능에 따라 큰 성능 차이를 보인다. 고성능인 라즈베리 파이2에서 가장 뛰어난 성능을 보이며 티모트와 아두이노는 라즈베리 파이2의 약 10배, 30배 이상의 성능 저하를 보인다.

HMAC과 Poly1305는 고사양 기기에서는 좋은 성능이 보이나 저사양으로 갈수록 성능이 급격하게 떨어진다. 그러나, CMAC은 성능 저하 폭이 작아 모든 기기에서 일정 수준의 성능을 보여준다. 특히 저성능인 아두이노에서 타 알고리즘에 비해 월등히 높은 성능을 보인다.

블록 암호 기반 MAC인 CMAC은 타 알고리즘에 비해 작은 크기의 블록을 사용하여 오버헤드가 작다. 즉, 요구 연산량이 적기 때문에 평균적으로 가장 좋은 성능을 보여준다.

가장 좋은 성능은 보인 CMAC은 사용하는 블록 암호를 자유롭게 변경 가능하다. 이 특징을 고려하여 경량 암호 알고리즘 적용 시의 CMAC 성능을 측정하였다. 〈그림 7〉은 경량 블록 암호 적용 시 CMAC의 성능을 보여준다.

측정 결과, 32비트 기기인 라즈베리 파이2에서는 32비트 경량 블록 암호인 LEA가 가장 좋은 성능을 보인다. 그 외에 16비트 티모트, 8비트 아두이노에서는 8비트 경량 블록 암호인 HIGHT가 가장 좋은 성능을 보인다. 경량 암호 별로 평균 성능을 계산하였을 때 HIGHT가 가장 좋은 성능을 보인다. 위 결과를 다음과 같은 결과를 도출할 수 있다. 기기별로 최적화된 경량 암호 알고리즘이 가장 좋은 성능을 보이지만, MAC에 적용한 경우, 8비트 환경에 최적화된 알고리즘이 평균적으로 가장 좋은 성능을 보인다. 따라서, 모든 기기에 적용 가능한 보안 기술은 8비트 기기에 최적화되어야 한다.

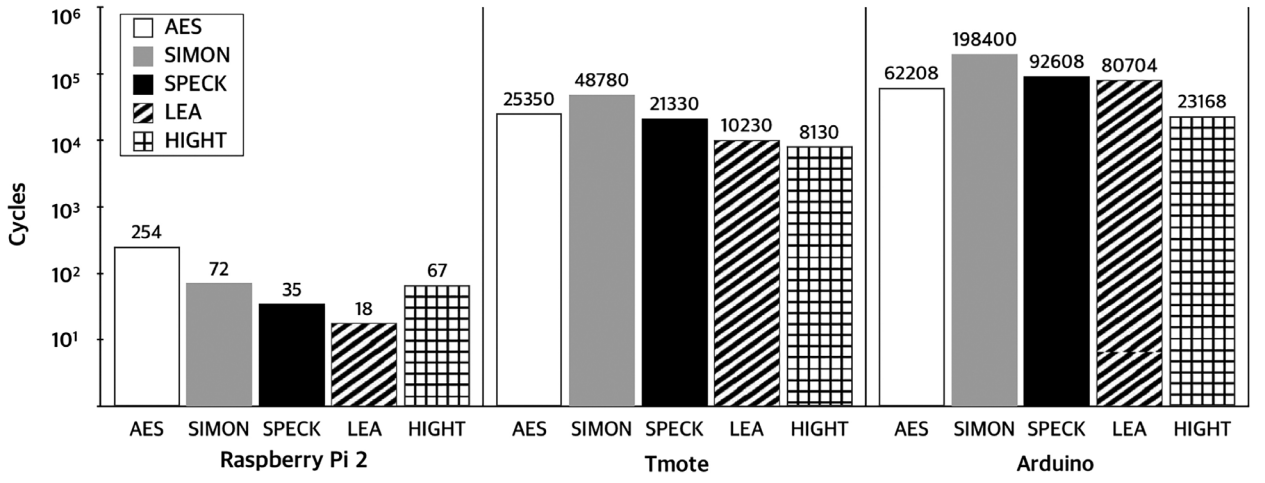

그림 7. 경량 블록 암호 기반 CMAC 성능 측정 결과 (로그 스케일)

V. 결 론

본 논문에서는 IoT환경에서 기기 간 안전한 통신을 위해 사용되는 대표적인 블록 암호 알고리즘과 메시지 인증 코드의 특징을 정리하였다. 또한, 실제 IoT 실험 기기에 각 알고리즘을 직접 실행하여 성능 측정 결과를 제시하였다.

실험을 통해 저성능의 기기일수록 알고리즘의 성능이 낮다는 점을 볼 수 있었다. 특히 8비트 기기에서는 타 기기에 비해 많은 성능 저하를 보였고, 이를 개선하기 위한 연구가 필요함을 인지하였다. 향후에는 저성능 기기에서 경량화 연구를 통해 보다 향상된 성능을 보일 수 있는 연구를 진행할 예정이다.

참 고 문 헌

[1] HP, "Internet of things research study," 2015.

[2] GT. Park, HJ. Ham, and JH. Lee, "Design and Implementation of Lightweight Encryption Algorithm on OpenSSL," 12th Ed., KISC Press, 2014.

[3] KJ. Ha, CH. See, and DY. Kim, "Design of Validation for a Crypto-Algorithm Implementation," 4th Ed., KISC Press, 2014.

[4] P. Bae, YM. Jo, EK. Moon, and YB. Ko, "Heterogeneous Interface Decision Engine and Architecture for Constructing Low Power Home Networks," 2nd Ed., KISC Press, 2015.

[5] J. Daemen, and V. Rijmen, "The design of Rijndael: AES-the advanced encryption standard," Springer, 2013.

[6] J. Daemen, M. Peeters, G. V. Assche, and V. Rijmen, "Nessie Proposal: Noekeon," NESSIE Workshop, 2000.

[7] D. Kwon, J. Kim, S. Park, S. Sung, Y. Sohn, J. Song, Y. Yeom, E. Yoon, S. Lee, J. lee, S. Chee, D. Han, and J. Hong, "New Block Cipher: ARIA," LNCS 2971, pp. 432-445, Springer, 2004.

[8] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee, "HIGHT: A New Block Cipher Suitable for Low-Resource Device," CHES 2006, LNCS vol. 4249, pp. 46-59, Springer, 2006.

[9] C. Lim, and T. Korkishko, "mCrypton-A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors," WISA 2005, LNCS vol. 3786, pp. 243-258, Springer, 2005.

[10] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," CHES 2007, LNCS vol. 4727, pp. 450-466, Springer, 2007.

[11] Sony Corporation, "The 128-bit Blockcipher CLEFIA Algorithm specification,", 2007.

[12] Eisenbarth, T. Paar, C. Poschmann, A. Kumar, S. Uhsadel, "A Surveyof Lightweight Cryptograp

y Implementations," Design&Test of Computers, vol. 24(6), pp. 522-533, IEEE, 2007.
[13] T. Suzake, K. Minematsu, S. Morioka, and E. Kobayashi, "TWINE : A Lightweight Block Cipher for Multiple Platforms," LNCS vol. 7707, pp. 339-354, Springer, 2013.
[14] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: An Ultra-Lightweight Blockcipher," CHES 2011, LNCS vol. 6917, pp. 342-357, Springer, 2011.
[15] W. Wu and L. Zhang, "LBlock: A Lightweight Block Cipher," ACNS, LNCS vol. 6715, pp. 327-344, Springer, 2011.
[16] Z. Gong, S. Nikova, and Y. Law, "KLEIN: A New Family of Lightweight Block Ciphers," RFID SP, LNCS vol. 7055, pp. 1-18, Springer, 2012.
[17] R. Beaulieu, D. Shors, J. Smith, S. T. Lark, B. Weeks, and L. Wingers, "The Simon and Speck Families Of Lightweight Block Ciphers," NSA, 2013.
[18] D. Hong, J. Lee, D. Kim, D. Kwon, K. Ryu, and D. Lee, "LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors," ISA, LNCS vol. 8267, pp. 3-27, Springer, 2013.
[19] A.J. Menezed, P.C. van Oorschot, and S.A. Vanstone, "Handbook of Applied Cryptography," CRC Press, Boca Raton, USA, 1999.
[20] T. Iwata and K. Kurosawa, "OMAC: One-Key CBC MAC," Fast Software Encryption, vol. 2887, pp. 129-153, 2003.
[21] M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication," Proc. 16th Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '96), vol. 1109, pp. 1-15, 1996.
[22] D. Bernstein, "The Poly1305-AES Message-Authentication Code," Proc. 12th Int'l Conf. Fast Software Encryption (FSE '05), vol. 3557, pp. 32-49, 2005.
[23] M. Dworkin, NIST special publication 800-38B. NIST special publication, 800(38B), 38B, 2005.
[24] F. Pub, "198, the keyed-hash message authentication code (hmac)." Federal Information Processing Standards Publication 198 (2002).
[25] A. Langley, "ChaCha20 and Poly1305 for TLS." Adam Langley's Weblog (October 2013), from https://www.imperialviolet. org/2013/10/07/chacha20. html.
[26] Arduino UNO, https://www.arduino.cc/en/Main/ArduinoBoardUno
[27] Tmote SKY TelosB, https://capsil.org/capsilwiki/index.pp/TELOSB/Tmote_Sky
[28] Raspberry pi2, https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

약 력

문 시 훈

2014년 한국산업기술대학교 게임공학과 학사
2015년~현재 연세대학교 정보대학원 석사과정
관심분야: 사물인터넷, 네트워크 보안, 모바일 보안 등

김 민 우

2015년 세종대학교 컴퓨터공학과 학사
2015년~현재: 연세대학교 정보대학원 석사과정
관심분야: 사물인터넷, 네트워크 보안, 모바일 보안 등

권 태 경

1992년 연세대학교 컴퓨터과학과 학사
1995년 연세대학교 컴퓨터과학과 석사
1999년 연세대학교 컴퓨터과학과 박사
1999년~2000년 U.C. Berkely Post-Doc.
2001년~2013년 세종대학교 컴퓨터공학과 교수
2007년~2008년 Univ. Maryland at College Park 교환 교수
2013년~현재 연세대학교 정보대학원 교수
관심분야: 암호 프로토콜, 네트워크 프로토콜, 센서 네트워크 보안, HCI 보안 등