ISSN 1975-8359(Print) / ISSN 2287-4364(Online)
The Transactions of the Korean Institute of Electrical Engineers Vol. 64, No. 10, pp. 1460~1468, 2015
http://dx.doi.org/10.5370/KIEE.2015.64.10.1460

# 고처리율 파이프라인 LEA 설계

## Design of the High Throughput Pipeline LEA

이 철* · 박 능 수†
(Chul Lee · Neungsoo Park)

**Abstract** - As the number of IoT service increases, the interest of lightweight block cipher algorithm, which consists of simple operations with low-power and high speed, is growing. LEA(Leightweight Encryption Algorithm) is recently adopted as one of lightweight encryption standards in Korea. In this paper a pipeline LEA architecture is proposed to process large amounts of data with high throughput. The proposed pipeline LEA can communicate with external modules in the 32-bit I/O interface. It consists of input, output and encryption pipeline stages which take 4 cycles using a muti-cycle pipeline technique. The experimental results showed that the proposed pipeline LEA achieved more than 7.5 Gbps even though the key length was varied. Compared with the previous high speed LEA in accordance with key length of 128, 192, and 256 bits, the throughput of the pipeline LEA was improved 6.45, 7.52, and 8.6 times. Also the throughput per area was improved 2, 1.82, and 2.1 times better than the previous one.

**Key Words** : Lightweight encryption algorithm(LEA), Block cipher algorithm, FPGA, Encryption hardware

## 1. 서 론

최근 모바일 산업의 급성장과 함께 사물 인터넷(IoT, Internet of Things)과 클라우드 서비스와 융합함에 따라 이러한 새로운 환경에서의 보안이 중요한 이슈가 되고 있다. 이로 인하여 기존의 환경에 적합하도록 설계되었던 정보보호 기술들이 사물인터넷 등의 새로운 환경에 적합하지 않은 경우들이 생기기 시작했다. 예를 들어 블록 암호 알고리즘 AES(Advanced Encryption Standard)[1, 2]의 경우 높은 보안 안정성 덕분에 국제표준으로 널리 사용되어왔지만 높은 연산량과 메모리 사용량으로 인해 일부 소형 디바이스에서 성능을 저하시켜 사물 인터넷 환경에서는 적합하지 않은 단점이 있다. 이러한 한계점을 극복하기 위해 보다 가볍고 빠르면서 소형으로 구현 가능한 경량 블록 암호 알고리즘들에 대한 연구들이 지속적으로 이루어지고 있다[3].

경량 블록 암호 알고리즘은 구현하는 방법에 따라 SPN (Substitution Permutation Network)기반 구조와 ARX(Addition, Rotation, Exclusive OR) 구조가 있다. SPN기반 경량 블록 암호 알고리즘은 AES와 마찬가지로 내부 연산을 구성하되 S-box와 P-box를 축소시켜 사용한 대입, 순열 연산 등을 기본으로 한다. 관련 블록 암호 알고리즘으로는 현재 ISO, IEC 표준으로 정의되어 있는 PRESENT[4], SONY사에서 제안하여 일본에서 사용되는 CLEFIA[5], CRYPTON 암호화 기법을 경량화한 mCrypton [6], AES암호 구조에서 키 스케줄(Key Schedule) 부분을 제외하여 경량화시킨 LED(Light Encryption Device)[7], 기존 DES (Data Encryption Standard) 알고리즘에서 짧은 키 길이를 개선한 DELSX[8], Revere Security 사에서 Enigma rotor machine을 기반으로 구성한 HummingBird-2[9] 등이 있다. 반면, ARX 구조는 연산량이 많은 S-box와 P-box의 사용을 배제하여 경량성을 높이면서 간단한 사칙연산을 적절하게 조합하여 라운드 수를 증가시켜 보안 안정성을 확보했다. 관련 연구로는 non-linear boolean과 shift, XOR 연산으로 구성된 KATAN[10], OpenPGP 표준에 채택된 Lai-Massey scheme구조의 IDEA(In ternational Data Encryption Algorithm)[11], NSA(National Security Agency)에 의해 제안되었으며 Feistel 구조를 기반으로 설계된 SIMON과 SPECK[12], steam cipher를 위해 개발된 Salsa[13] 등이 있다. 국내에서도 이런 추세에 따라 2013년 8월 경량 블록 알고리즘 LEA(Lightweight Encryption Algorithm)[16]을 발표하였으며 12월에는 한국정보통신협회(TTA) 표준으로 등록되어 기술 보급이 활발하게 이루어지고 있다. LEA는 128비트 블록을 입력으로 가지고 요구되는 암호화 레벨에 따라 128, 192, 256 비트의 비밀 키를 선택할 수 있다. LEA는 또한 32비트 단위 Addition과 Rotation, XOR 연산을 사용하는 ARX 구조로 구성되어 자원 사용량이 적으면서도 빠른 속도를 장점으로 한다.

사물 인터넷에 대한 관심이 증대함에 따라 최근에 암호화 표준으로 발표된 LEA 암호 알고리즘의 효율적 연산을 위한 전용 하드웨어를 구현하는 연구들도 활발히 이뤄지고 있다[14, 15]. [14]에서는 크기를 작게 구현한 공간 중점 하드웨어와 빠른 성능을 목표로 한 고속 처리 하드웨어를 각각 제시하였다. 특히 고

† Corresponding Author : Dept. of Computer Science and Engineering, Konkuk Univerity, Seoul, Korea
E-mail : neungsoo@konkuk.ac.kr
* Dept. of Smart ICT Convergence, Konkuk Univerity, Korea
Received : August 23, 2015; Accepted : September 24, 2015

속 처리 하드웨어의 경우는 블록 데이터의 효율적인 처리를 위해 128비트의 입출력 단위를 가지기 때문에 다른 하드웨어 모듈과 128 비트 단위의 인터페이스를 요구하는 문제가 있고 32 비트 인터페이스를 요구할 경우 추가적인 시간이 필요하게 된다. 또한 제시한 32비트 입출력을 하는 하드웨어의 경우 32비트 연산을 기반으로 구현하고 연산모듈을 재사용하여 크기는 작게 하였지만 성능이 떨어지는 단점이 있다. 이러한 모듈의 경우 소형 디바이스에서는 적합할 수 있으나 향후 사물인터넷 및 클라우드 환경의 서비스에서 대량으로 발생하는 데이터의 암호화 및 복호화를 위하여서는 보다 고성능의 암복호화 모듈의 필요성이 있다.

본 논문에서는 대량의 데이터를 고속으로 암호화하기 위한 방법으로 파이프라인 기법을 적용한 LEA 하드웨어를 제안하고자 한다. 제안한 파이프라인 구조는 하나의 파이프라인 스테이지 연산에 4 사이클이 필요한 대신에 매 사이클에 32 비트 입출력을 하도록 구성이 되어 있어 외부 하드웨어 모듈과 보다 효과적으로 인터페이스를 할 수 있도록 설계하였다. 또한, 6개의 암호화 스테이지 뿐만 아니라 입력과 출력을 포함한 모든 파이프라인 스테이지가 동일한 4 사이클을 사용하도록 설계하여 입출력을 암호화 연산 단계에 숨김으로써 효율을 극대화하였다. 하드웨어 기능 검증과 성능평가를 위하여 제안한 파이프라인 LEA를 FPGA를 이용하여 구현하였다. 실험을 통한 기존 고속 LEA 하드웨어와의 비교에서 시간당 처리율의 경우 128 비트 키에서 6.45배, 192비트 키에서 7.52배, 256비트 키에서 8.6배 향상하였음을 확인하였으며, 면적 대비 시간당 처리율도 128비트 키에서 2배, 192비트 키에서 1.82배, 256비트 키에서 2.1배로 향상하였음을 확인하였다.

본 논문의 구성은 2장에서는 LEA 블록 암호화 알고리즘에 대해 소개하고, 3장에서는 제안한 파이프라인 LEA에 대해 설명한다. 4장에서는 실험 결과에 따른 기존 고속 LEA하드웨어와의 성능 차이를 비교 분석하고, 마지막 5장에서 결론을 맺는다.

## 2. Lightweight Block Encyrption Algorithm(LEA)

### 2.1 LEA 알고리즘

LEA[16]은 128비트 평문(암호문)을 입력 받아 128비트 암호문(평문)을 생성하는 알고리즘이다. LEA는 S-Box와 같은 복잡한 연산을 배제하고 32 비트 워드 단위의 Addition, Rotation, Exclusive OR 세 가지 연산만으로 구성하여 이 연산들을 적절하게 배치함으로써 경량화하였다. Fig. 1은 LEA의 암·복호화 과정을 나타낸 것으로 $\text{Round}^{enc}$와 $\text{Round}^{dec}$는 각각 암호화 라운드 함수와 복호화 라운드 함수, $X_i (0 \le i \le Nr)$는 i번째 암호화(복호화) 함수에 입력되는 128비트 변수, $RK^{enc}$와 $RK^{dec}$는 각각 암호화 라운드 키와 복호화 라운드 키, $Nr$은 반복되는 라운드 수를 나타낸다. LEA는 128, 192, 256비트의 세 가지 키 길이를 지원하고 키 길이에 따라 24, 28, 32번 암호화(복호화) 라운드 함수를 반복하여 계산한다. 각 라운드 함수에는 키 스케줄을 통해 생성된 암호화(복호화) 라운드 키가 사용되며 라운드 키는 반복되는 라운드 수만큼 존재한다.

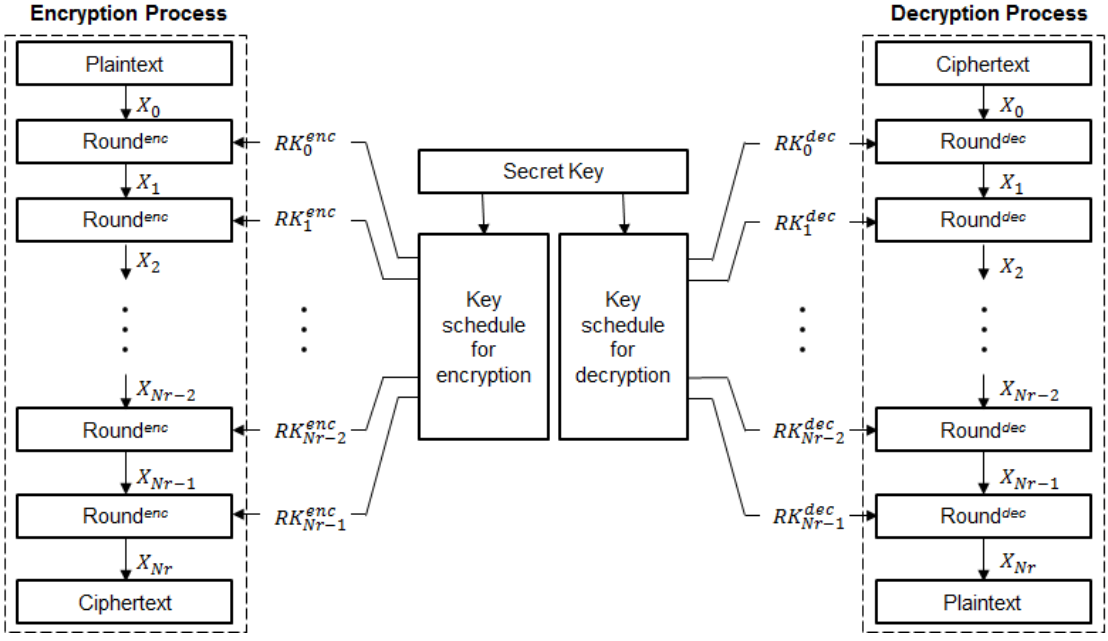



**그림 1** LEA 암·복호화 과정

**Fig. 1** Encryption and Description Process of LEA

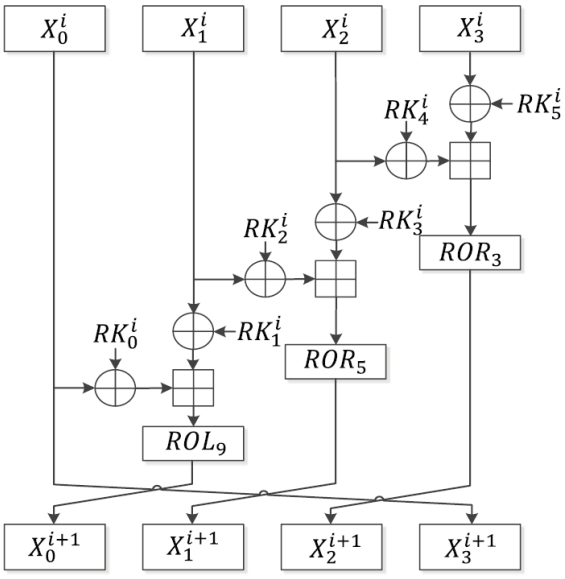



**그림 2** LEA 라운드 함수
**Fig. 2** Round Function of LEA

Fig. 2는 LEA의 라운드 함수를 도식화한 것으로 사용된 연산자는 32 비트 연사자로 ⊞은 Moduler Addtion, ⊕은 XOR, ROR과 ROL은 각각 Rotation Right와 Rotation Left 연산을 나타낸다. $X_i$는 $i$번째 라운드에 입력되는 변수로 $X_i=(X_i[0], X_i[1], X_i[2], X_i[3])$, $RK_i^{enc}$는 $i$번째 라운드에 사용되는 192비트 라운드 키로 $RK_i^{enc}=(RK_i^{enc}[0], RK_i^{enc}[1], RK_i^{enc}[2], RK_i^{enc}[3], RK_i^{enc}[4], RK_i^{enc}[5])$를 나타낸다. 키 스케줄에서 생성되는 라운드 키 $RK_i^{enc}$는 비밀키의 길이에 관계없이 192비트로 일정하다. 128비트 라운드 키의 경우 $RK_i^{enc}=(RK_i^{enc}[0], RK_i^{enc}[1], RK_i^{enc}[2], RK_i^{enc}[1], RK_i^{enc}[3], RK_i^{enc}[1])$와 같이 일부 키를 재사용하고, 256비트 키의 경우 $RK_i^{enc}=(RK_i^{enc}[(6i)\bmod 8], RK_i^{enc}[(6i+1)\bmod 8], RK_i^{enc}[(6i+2)\bmod 8], RK_i^{enc}[(6i+3)\bmod 8], RK_i^{enc}[(6i+4)\bmod 8], RK_i^{enc}[(6i+5)\bmod 8])$와 같이 라운드에 따라 일부 키를 선별하여 사용한다. LEA에서 키 스케줄은 입력된 비밀키와 상수를 32비트 단위로 더하여 일정횟수 Rotation하여 비밀키를 생성한다. 상수는 키 스케줄 내부에 저장되어 있는 상수를 일정횟수 Rotation 시킴으로써 생성해 주며 키 길이에 따라 32비트 4, 6, 8개가 사용한다.

### 2.2 LEA 하드웨어 관련 연구

LEA는 최근에 전용 하드웨어 구현에 대한 연구가 활발히 진행이 되고 있다[14, 15]. [14]에서는 성능을 최대화하기 위한 고속 LEA 하드웨어 구조와 크기를 소형화하는 공간 중심 LEA 하드웨어 구조를 제안하였다. 이 중 공간 중심 LEA 하드웨어는 라운드 연산에 사용되는 32비트 덧셈 모듈을 하나로 제한함으로써 32비트 단위의 작은 입출력으로 모듈 구현이 가능하면서 전체 면적을 축소하였다. 그러나 공간 중심 LEA 하드웨어는 사이클당 32비트 연산만을 수행하여 하나의 라운드를 여러 사이클 동안 처리해야하기 때문에 속도가 저하되는 단점을 가진다. 반면, 고속 LEA 모듈은 128 비트 데이터를 한 사이클에 처리하기 위하여 한 라운드에 필요한 모든 32비트 연산들을 동시에 수행함으로써 암호화 속도를 향상시켰다. 고속 LEA 하드웨어는 1 라운드를 1 사이클에 처리 가능하게 되었지만 한 번에 128비트 단위의 입출력을 필요로 한다. 이러한 구조는 외부 하드웨어 모듈과의 연동에 어려운 단점이 있으며 이를 위하여 인터페이스를 32비트 단위로 변경하면 추가 지연 시간이 발생되게 된다. 따라서 기존의 LEA 하드웨어는 32비트 입출력 단위를 가지는 대신 속도가 느리거나 빠른 대신 128비트 입출력 단위를 가지는 단점이 있다. [15]는 공간 중심의 LEA 하드웨어를 기초로 하여 암·복호화 연산이 하드웨어 자원을 공유되도록 설계하였으나 속도가 느린 단점이 있다.

## 3. 파이프라인 LEA

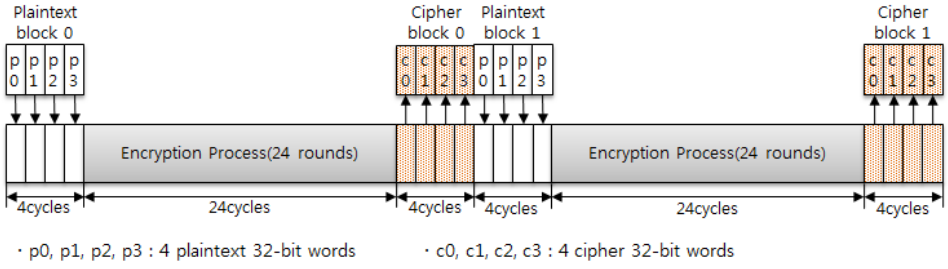
128 비트 평문 블록을 암호화하여 128 비트의 암호문을 빠르게 생성하는 고속 LEA 하드웨어 구현하기 위해서는 모든 입출력이 128 비트 단위로 하는 것이 효율적일 것이다. 그러나 이러한 구조의 경우 기존의 CPU 등 외부 하드웨어 모듈과 연동하기 위하여 128 비트의 입출력 버스가 필요하게 된다. 따라서 다른 모듈과의 효율적인 인터페이스를 위하여 입출력 단위를 32비트 단위로 변경할 경우 추가 시간이 요구되는 단점이 생기게 된다. 본 연구에서는 이러한 문제들을 극복하기 위하여 입출력 폭을 32비트 단위로 하면서 데이터를 고속으로 암호화(복호화)하기 위하여 파이프라인 기법의 LEA 하드웨어를 제안하고자 한다. 제안된 파이프라인 LEA 하드웨어는 기존의 하드웨어와 같이 라운드 블록, 키 스케줄 블록을 포함하고 있으나 이를 파이프라인 기법으로 재구성하고자 한다. 특히 파이프라인의 효율적인 연산을 위하여 키 스케줄 블록의 경우 각 연산에 사용되는 라운드 키를 on-the-fly 방식으로 생성하여 적용한다. 먼저 128 비트 키를 가지는 파이프라인 LEA 하드웨어를 설명하고 이를 기반으로 192비트와 256 비트 키를 가지는 파이프라인 LEA로 확장하고자 한다.

### 3.1 128 비트 파이프라인 LEA 하드웨어

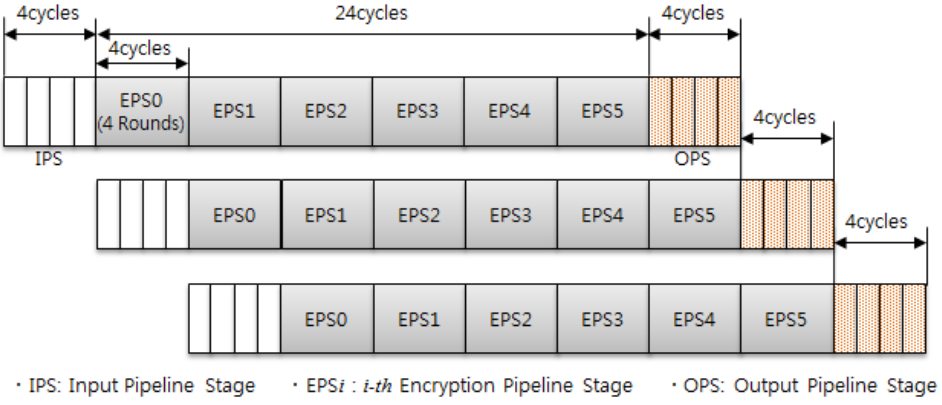
파이프라인 LEA를 효율적으로 설계하기 위하여 각 파이프라인 스테이지가 균등하도록 설계하는 것은 성능 향상을 높이기 위하여 매우 중요하다. 먼저 파이프라인을 고려하지 않은 처리 방식을 분석하여 이를 바탕으로 효율적인 파이프라인 구조를 제안하고자 한다. 파이프라인을 고려하지 않은 고속 128 비트 LEA는 암호화를 위하여 128 비트 블록 단위의 24 라운드 연산을 필요로 하여 최소 24 사이클이 요구된다. 만일 입출력을 32 비트 단위로 구성을 할 경우 Fig. 3(a)와 같이 입력을 위하여 4 사이클 그리고 출력을 위하여 4 사이클이 추가로 필요하게 된다. 따라서 하나의 128 비트 블록을 암

호화하기 위하여 매번 32 사이클이 요구된다. 이를 파이프라인으로 재구성하기 위하여 본 논문에서는 4개의 사이클을 하나의 파이프라인 스테이지에서 처리하도록 정의하는 멀티 사이클 파이프라인 기법을 제안한다. 입력과 출력은 하나의 파이프라인 스테이지로 정의하고 암호화 과정을 6개의 암호화 파이프라인 스테이지(EPS: encryption pipeline stage)로 나누어 Fig. 3(b)와 같이 총 8개의 파이프라인 스테이지로 구성한다. 따라서 각 파이프라인 스테이지의 소요시간이 균등하게 4 사이클로 구성되어 있어 불필요한 지연시간을 최소화하고 입출력에 처리되는 시간과 암호화 연산에 처리되는 시간을 모두 감출 수 있어 효율을 극대화하는 것이 가능하다.

Fig. 4는 제안된 파이프라인 LEA 구조를 묘사하고 있다. 제안된 파이프라인 LEA는 각 암호화 파이프라인 스테이지를 처리하기 위한 EPS Block(Encryption Pipeline Stage Block), IS(Input




(a) non-pipeline process




(b) pipeline process

**그림 3** 32비트 입출력을 가진 128비트 LEA 프로세스
**Fig. 3** 128-bit LEA process with 32-bit I/O

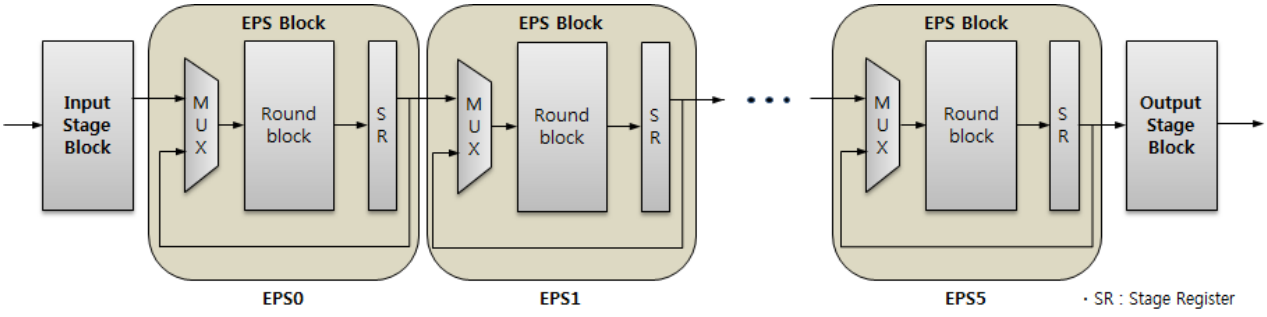



**그림 4** 파이프라인 LEA 구조
**Fig. 4** Architecture of pipeline LEA

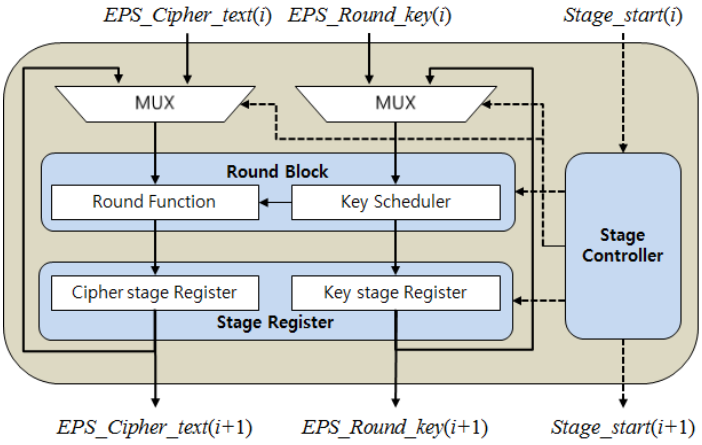



**그림 5** 암호화 파이프라인 단계(EPS) 블럭
**Fig. 5** Encryption pipeline stage(EPS) block

Stage) Block, 그리고 OS(Output Stage) Block으로 구성되어 있다. 파이프라인 LEA의 IS Block은 매 사이클 32 비트의 입력을 받아 저장을 하여 128 비트의 블록을 형성하며, OS Block은 암호화된 결과를 매 사이클마다 32 비트씩 출력을 하도록 구성되어 있다.

EPS Block은 실질적인 암호화 연산이 이루어지는 모듈이다. 기본적으로 하나의 EPS Block은 Multiplexer와 Round Block 그리고 Stage Register, 그리고 Stage Controller로 구성되어 있다. Fig. 5와 같이 $i$-번째 EPS Block을 가정하여 보자. 이 경우 이전 단계로 부터 $EPS_Cipher_text(i)$과 $EPS_Round_key(i)$를 데이터 입력으로 받고 $Stage_start(i)$을 제어신호로 받게 된다. 하나의 파이프라인 단계에서는 4개의 라운드 연산을 반복하여 처리하여야 한다. 따라서 Multiplexer를 이용하여 내부의 Stage Register에 저장된 값이나 이전 파이프라인 스테이지에서 전달되는 결과값을 선택하여 Round Block의 입력으로 전달하게 된다. Round Block은 라운드 연산을 하는 라운드 함수 모듈과 이때 필요로 하는 라운드 키 값을 on-the-fly 방식으로 계산하는 Key Scheduler로 구성하였다. Round Block에서 계산된 결과 값은 Stage Register에 저장하게 된다. Stage Register에는 암호화 결과를 저장하는 Cipher Stage Register와 On-the-fly 키 스케줄을 지원하기 위한 Key Stage Register로 구성되어 있다. Stage Register에 저장된 값은 Multiplexer를 통하여 Round Block의 입력으로 ($i$+1)번째 EPS Block의 입력으로 전달된다. Stage Controller는 이전 ($i$-1)번째 EPS 블록으로부터 $Start\_start(i)$ 신호를 받아서 ($i$)번째 EPS 블록이 동작하도록 한다. Stage Controller는 암호화 라운드 연산이 4번 실행되는지를 판단하며 이때 필요한 제어신호를 생성하여 Multiplexer, Round Block, 그리고 Stage Register를 제어한다. 라운드 연산이 4회 완료되면 $Start\_start(i+1)$ 신호를 생성하여 ($i$+1)번째 EPS 블록이 동작하도록 전달한다.

### 3.2 파이프라인 LEA의 키 확장과 성능 분석

LEA의 평문 블록의 크기는 128 비트로 동일하지만 키 길이는 128, 192, 256 비트로 확장할 수 있으며 이에 따라 라운드 수는 24, 28, 그리고 32 라운드로 증가된다. 따라서 192 비트와 256 비트 키를 사용하는 LEA의 경우에도 라운드 연산 수가 모두 4의 배수이므로 이들을 위한 EPS 수는 7 스테이지와 8 스테이지로 확장할 수가 있다. 이때 각 라운드 함수에 사용되는 라운드 키의 경우 모두 192 비트로 동일하므로 LEA의 라운드 연산은 키 길이에 상관없이 동일하다. 따라서 키가 192, 256으로 확장되어도 EPS Block의 Round Function과 Stage Register는 128비트 LEA와 동일하게 구현할 수 있다. 키 스케줄은 키 길이에 관계없이 192비트의 일정한 라운드 키를 생성하기 위해 128 비트 키의 경우 2.1절에서와 같이 일부를 중복 사용하여 라운드 키로 사용하지만, 192 비트 키의 경우에는 키 모두를 라운드 키로 사용하며 256 비트 키의 경우 키의 일부를 선택하여 라운드 키로 사용하도록 되어 있다. 따라서 192 비트와 256 비트 키의 LEA 경우 EPS Block의 Key Scheduler에 키 연산 로직을 추가하고 Key stage Register는 매 라운드의 라운드 키 연산 결과를 저장하기 위해 각각 192비트와 256비트로 확장한다.

LEA 구현의 경우 구현 방법에 따라 다양하게 구현을 할 수가 있다[14]. Table 1은 키 길이와 입출력 버스 구조 변화에 따른 LEA 이론적 성능을 정리한 것이다. 여기서 $N$은 128 비트 평문 블록의 수를 나타낸다. [14]에서 제안된 고속 LEA는 128 비트 입출력 버스를 이용하여 구현하도록 제안이 되었으며 이때 $N$ 개의 평문 블록을 암호화하기 위하여 필요한 시간을 $T_{speed128}$로 나타낸다. 키가 128 비트인 경우 한 라운드 연산 당 하나의 사이클이 필요하여 총 24 사이클이 소요된다. 따라서 키가 192, 256인 경우에 그 시간도 28과 32이 사이클이 소요된다. $T_{speed32}$는 이러한 구조를 32 비트 입출력으로 전환할 경우 암호화하기 위하여 필요한 시간을 나타낸 것으로 앞에 분석에서와 같이 128 비트의 평문입력과 128 비트 암호문

**표 1** 다양한 LEA 구현 결과 성능 분석

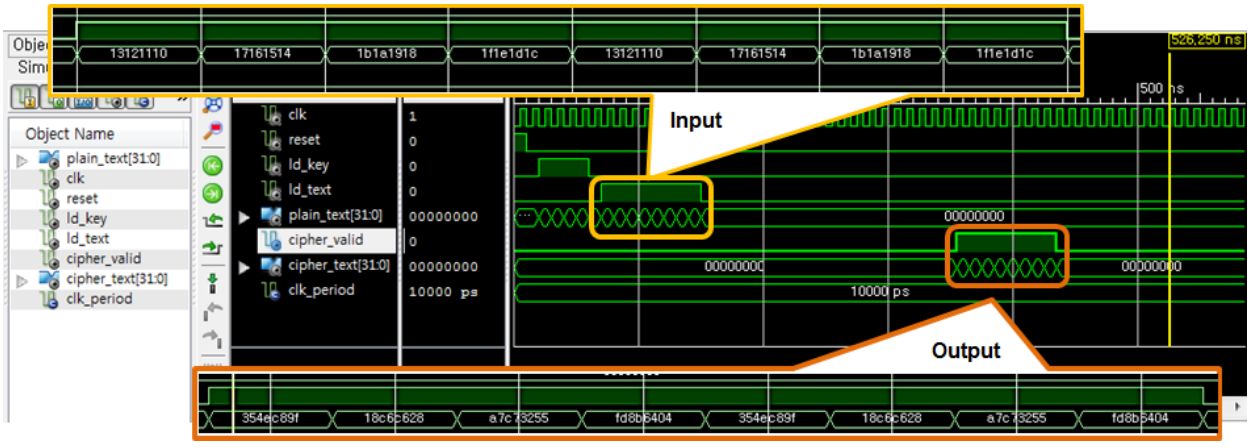
**Table 1** The performance analysis of various LEA implementations

| Key | $T_{speed128}$ | $T_{speed32}$ | $T_{area}$ | $T_{pipeline}$ |
|---|---|---|---|---|
| 128 | $24\times N$ | $(4+24+4)\times N$ | $(4\times 24)\times N$ | $28+4\times N$ |
| 192 | $28\times N$ | $(4+28+4)\times N$ | $(3\times 28)\times N$ | $32+4\times N$ |
| 256 | $32\times N$ | $(4+32+4)\times N$ | $(6\times 32)\times N$ | $36+4\times N$ |

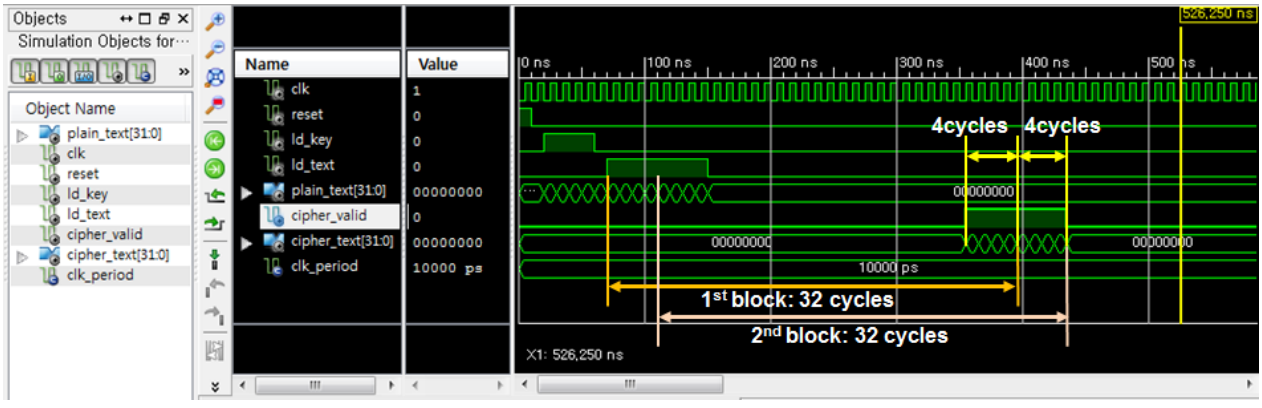
출력을 위하여 각각 4 사이클씩 추가로 요구된다. 또한 [14]는 작은 크기로 효율적으로 구현하기 위하여 사이클 당 32비트 연산을 하도록 구현한 공간 중심의 LEA도 제안하였다. 이 구현은 한 라운드에 3-6 사이클을 필요로 하며 전체적인 암호화 시간($T_{area}$)도 이에 비례하게 증가한다. 본 연구에서 제안한 파이프라인 LEA의 암호화 시간($T_{pipeline}$)는 입출력 버스 크기가 32비트이며 첫 평문이 암호화 된 후 매 4 사이클마다 암호문을 출력하게 되어 있다. 따라서 제안한 파이프라인 LEA는 키 길이에 관계없이 일정한 성능을 보장할 수 있는 장점이 있다. 이는 파이프라인을 적용하지 않은 고속 LEA와 비교할 때 이론적으로 128비트 키의 경우 약 6배, 192비트 키의 경우 약 7배, 256비트 키의 경우 약 8배의 속도의 향상을 이룰 수 있을 것이다.

## 4. 실 험

모든 구현은 Xilinx의 ISE Design Suit Ver. 14.7을 사용하여 Register Transfer Level(RTL)의 VHDL로 개발되었다. 개발 칩은 Xilinx Vertex 5 Series XC5vfx70T을 사용하였다. 구현한 128비트 파이프라인 LEA 모듈의 기능을 검증하기 위하여 한국정보통신기술협회(TTA) 표준문서[16]에 표시된 참조구현 값인 16진수 128비트 평문 "10111213 14151617 18191a1b 1c1d1e1f"와 16진수 키 "3c2d1e0f 78695a4b b4a59687 f0e1d2c3"을 사용하였으며 암호화 결과로 문서와 같이 16진수 128비트 암호문 "354ec89f 18c6c628 a7c73255 fd8b6404"가 출력되는 것을 확인하였다.




(a) Results verification




(b) Functional verification

**그림 6** 128비트 키 파이프라인 LEA 시뮬레이션 결과

**Fig. 6** Simulation results of an implemented pipeline LEA using 128bits key

**표 2** 단위 시간당 처리량에 따른 LEA 구현 결과
**Table 2** Throughput of LEA implementations

| | Key(bits) | Latency (cycles) | Max. Frequency (MHz) | Max. Throughput (Mbps) | Throughput@200MHz (Mbps) |
|---|---|---|---|---|---|
| $LEA_{speed128}$ | 128 | 24 | 218.25 | 1164.00 | 1066.66 |
| | 192 | 28 | 218.70 | 999.75 | 914.29 |
| | 256 | 32 | 218.70 | 874.78 | 800.00 |
| $LEA_{area}$ | 128 | 96 | 255.98 | 341.31 | 266.67 |
| | 192 | 84 | 199.27 | 303.65 | 304.76 |
| | 256 | 192 | 269.80 | 179.86 | 133.33 |
| $LEA_{pipeline}$ | 128 | 24 | 234.55 | 7505.47 | 6400.00 |
| | 192 | 28 | 235.06 | 7521.98 | 6400.00 |
| | 256 | 32 | 235.06 | 7521.98 | 6400.00 |

**표 3** 총면적에 따른 LEA 구현 결과
**Table 3** Area of LEA implementations

| | Key(bits) | Area | | |
|---|---|---|---|---|
| | | FFs | LUTs | Slices |
| $LEA_{speed128}$ | 128 | 389 | 614 | 719 |
| | 192 | 517 | 936 | 913 |
| | 256 | 646 | 1066 | 1250 |
| $LEA_{area}$ | 128 | 400 | 366 | 506 |
| | 192 | 526 | 440 | 670 |
| | 256 | 649 | 549 | 816 |
| $LEA_{pipeline}$ | 128 | 1950 | 2513 | 2764 |
| | 192 | 2734 | 3378 | 3792 |
| | 256 | 3634 | 4365 | 5121 |

Fig. 6(a)는 시뮬레이션을 통하여 파이프라인 LEA 암호 결과를 보여주고 있다. Fig. 6(b)를 통하여 하나의 블록을 암호화하는데 필요한 시간(latency)으로 32 사이클이 소요되지만 그 후 매 4 사이클마다 다음 암호문이 출력되는 것을 확인하여 설계된 파이 프라인 LEA 하드웨어 모듈이 제안한 방식으로 정상적으로 동작을 하고 또한 이론적 성능을 보여주고 있음을 확인할 수 있었다.

Table 2는 키 길이 변화에 따른 LEA 하드웨어의 단위 시간당 처리량(Throughput)을 나타낸 것이다. $LEA_{pipeline}$은 제안한 파이프라인 LEA를 나타낸 것이고 $LEA_{speed128}$와 $LEA_{area}$는 각각 [14]에서 제안된 128 비트 입출력 버스를 사용하여 구현한 고속 LEA 하드웨어와 32비트 단순 연산 모듈만을 사용하여 구현한 공간 중심 LEA 하드웨어를 나타낸다. Latency는 한 블록을 암호화 하는데 소요되는 필요한 사이클 수를, Max. Frequency는 구현된 모듈이 FPGA에서 가질 수 있는 최대 주파수를, Max. Throughput은 최대 주파수를 이용할 경우 단위시간 당 처리량을 나타내고 있다. $LEA_{pipeline}$의 최대 주파수는 모두 약 235 MHz정도로 $LEA_{area}$와 비교했을 때 192비트 키에서만 35MHz 가량 증가되고 나머지 키에서는 각각 21MHz, 35MHz 가량 감소된 것을 확인 가능했으며 $LEA_{speed128}$와 비교했을 때는 최소 모든 키에서 15MHz 이상 향상된 것을 확인할 수 있었다. 최대 단위 시간 당 처리량은 $LEA_{area}$에 비해 128비트 키에서 22배, 192비트 키에서 24.77배, 256비트 키에서 41.82배 증가되었으며 $LEA_{speed128}$에 비해서도 128비트 키에서 6.45배, 192비트 키에서 7.52배, 256비트 키에서 8.6배로 보다 향상된 결과를 보여주고 있다. 또한 $LEA_{area}$와 $LEA_{speed128}$은 키 길이의 증가에 따라 그 처리량이 감소되는 반면 제안한 $LEA_{pipeline}$은 키 길이의 증가에 따라 latency는 $LEA_{speed128}$와 같이 증가하나 Max. Throughput은 키 변화와 관계없이 모두 7.5Gbps 이상의 일정한 성능을 유지하는 것을 확인하였다. 주파수를 200MHz로 일정하게 고정할 경우, Table 1의 성능 분석 결과와 같이 파이프라인 LEA 하드웨어가 기존 고속 $LEA_{speed128}$ 하드웨어보다 128비트 키의 경우 6배, 192비트 키의 경우 7배, 256비트 키의 경우 8배만큼 단위 시간당 처리량이 증가하는 것을 확인하였다.Table 3은 키

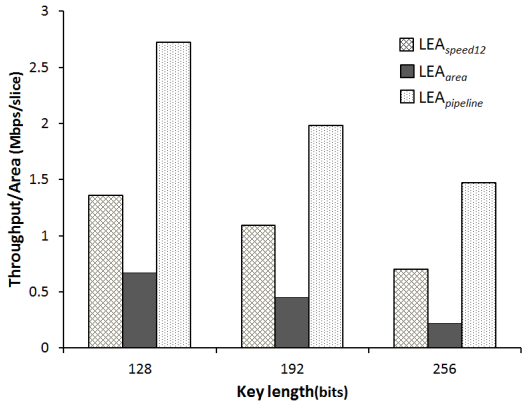



**그림 7** LEA의 면적 대비 처리량
**Fig. 7** Throughput per area of LEA implementations

길이에 변화에 따른 LEA 구현결과의 총면적(Area)을 비교한 것이다. FPGA 상에서 구현된 크기는 사용된 Flip Flop(FF)의 수와 Look Up Table(LUT)의 수, 그리고 FF와 LUT를 포함하고 있는 슬라이스(slice)의 총수를 이용하여 표현하였다. FPGA 상에서 하드웨어 모듈이 차지하는 총면적은 슬라이스의 수를 의미한다. 각 모듈의 면적은 $LEA_{area}$가 가장 작고 $LEA_{pipeline}$이 가장 큰 면적 차지하고 있다. 이는 $LEA_{area}$와 비교하였을 때 128비트 키에서 5.46배, 192비트 키에서 5.66배, 256비트 키에서 6.28배 크며, $LEA_{speed128}$과 비교하면 128비트 키의 경우 3.84배, 192비트 키의 경우 4.15배, 256비트 키의 경우 4.1 배 증가된 수치이다.

Fig. 7은 키 길이에 변화에 따른 LEA 구현결과의 면적 대비 처리량(Throughput/Area: TpA)을 그래프로 비교한 것이다. 면적 대비 처리량은 단위 면적이 단위 시간 당 처리 가능한 데이터양을 나타내고 있다. 다양한 구현의 방식에 따라 크기가 변화하므로, 이 지표는 크기 비용에 따른 성능의 향상도를 평가하기 위하여 사용하고자 한다. 면적 대비 처리량에서는 $LEA_{pipeline}$이 가장 효율이 뛰어났다. $LEA_{area}$와 비교하였을 때 128비트 키에서 4.06배, 192비트 키에서 4.4배, 256비트 키에서 6.68배로 4배 이상의 높은 수치를 보였으며 $LEA_{speed128}$과 비교하였을 때는 128비트 키에서 2배, 192비트 키에서 1.82배, 256비트 키에서 2.1배로 $LEA_{pipeline}$이 평균적으로 약 2배가량의 효율이 증가하였다. 면적대비 처리량을 통해 비교한 $LEA_{pipeline}$의 키에 따른 효율은 128비트 키가 2.72로 다른 키들에 비해 가장 우수한 것으로 나타났으며 키의 길이가 증가됨에 따라 192비트 키에서 1.98, 256비트 키에서 1.47로 약 1.35배씩 감소되었다. 이것은 단위 시간당 처리량이 약 7500MHz로 키와 관계없이 일정한 성능을 나타내는 것에 비해 각 모듈의 총면적은 라운드 수 증가에 따른 파이프라인 단계가 늘어남에 따라 1.35배씩 증가되기 때문이다. 반면에 $LEA_{speed128}$과 $LEA_{area}$는 키의 길이가 증가함에 따른 면적의 증가가 $LEA_{pipeline}$과 비교하여 상대적으로 작았지만, Throughput이 더 많이 감소하여 전체적인 면적 대비 처리량은 더 급격히 낮아졌다. 결과적으로 $LEA_{pipeline}$은 사용하는 라운드 연산 모듈과 키 스케줄 모듈의 증가로 총면적이 $LEA_{area}$ 보다 약 6배, $LEA_{speed128}$ 보다 약 4배 증가되었으나, 성능은 $LEA_{area}$ 보다 최소 22배에서 최대 40배까지, $LEA_{speed128}$ 보다 6~8배 정도 증가되어 전체적인 효율은 $LEA_{area}$와 비교할 경우 3~7배가량, $LEA_{speed128}$보다 약 2배 가량 향상되었음을 확인할 수 있었다.

## 5. 결 론

본 연구에서는 대량의 데이터를 고속으로 암호화하기 위한 방법으로 파이프라인 기법의 LEA 하드웨어를 제안하였다. 제안된 파이프라인 LEA는 입력과 출력 그리고 각 암호화 파이프라인 스테이지 시간을 균등하게 하여 파이프라인 효과를 극대화하도록 하였으며 이를 위하여 파이프라인 스테이지를 4사이클로 구성한 멀티 사이클 파이프라인 구조로 설계하였다. 실험을 통하여 처리 지연 시간이 지난 후 매 4 사이클마다 128 비트의 암호문을 출력하는 것을 확인하였다. 특히 제안된 파이프라인 LEA가 키 길이와 관계없이 모두 7.5 Gbps의 처리율을 달성하였으며 기존 고속 LEA하드웨어와 비교하였을 때 128비트 키에서 6.45배, 192비트 키에서 7.52배, 256비트 키에서 8.6배 향상하였다. 비록 키의 길이가 길어짐에 따라 파이프라인 스테이지 수가 커져서 파이프라인 LEA의 크기가 함께 증가하였지만, 면적 대비 처리율은 파이프라인 LEA가 기존 고속 LEA 구현 보다 128비트 키에서 2배, 192비트 키에서 1.82배, 256비트 키에서 2.1배 증가되어 파이프라인 LEA가 기존 고속 LEA보다 약 2배가량 효율이 향상되었음을 알 수 있었다. 본 연구는 IoT와 클라우드 서비스 환경에서 발생하는 대량의 데이터를 효과적으로 암복호화하기 위한 기술로 사용될 수 있으나 향후 IoT와 웨어러블 환경의 소형 디바이스에서도 효율적으로 암복호화 할 수 있는 저전력 LEA 하드웨어에 대한 연구가 필요하여 이를 진행하고자 한다.


### 감사의 글

이 논문은 2014년도 건국대학교 KU학술연구비 지원에 의한 논문임.


### References


[1] NIST, “FIPS 197: Advanced encryption standard (AES)." Techincal Report, National Institute of Standards and Technology (NIST), November 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[2] Ohyoung Song, and Jiho Kim, “Compact Design of the Advanced Encryption Standard Algorithm for IEEE 802.15.4 Devices,” Journal of Electrical Engineering &

Technology, Vol. 6, No. 3, pp. 418-422, March, 2011.
[3] WU Wenling, Lei ZHANG, "LBlock: a lightweight block cipher," In proceedings of 9th International Conference on Applied Cryptography and Network Security(ACNS 2011), LNCS, Vol. 6715, pp. 327-344, 2011.
[4] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, C. Vikkelsoe, "PRESENT: An ultra-lightweight block cipher," In proceedings of 9th International Workshop on Cryptographic Hardware and Embedded Systems(CHES 2007), LNCS, Vol. 4727, pp. 450-466, 2007.
[5] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata, "The 128-bit blockcipher CLEFIA," In proceedings of 14th International Workshop on Fast software encryption(FSE 2007), LNCS, Vol 4593, pp. 181-195, 2007.
[6] Chae Hoon Lim, Tymur Korkishko, "mCrypton-A lightweight block cipher for security of low-cost RFID tags and sensors," In proceedings of 6th International Workshop on Information Security Applications(WISA 2006), LNCS, Vol. 3786, pp. 243-258, 2006.
[7] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matt Robshaw, "The LED block cipher," In proceedings of 13th International Workshop on Cryptographic Hardware and Embedded Systems(CHES 2011), LNCS, Vol 6917, pp. 326-341, 2011.
[8] Gregor Leander, Christof Paar, Axel Poschmann, Kai Schramm, "New lightweight DES variants," In proceedings of 14th International Workshop on Fast Software Encryption(FSE 2007), LNCS, Vol. 4593, pp. 196 - 210, 2007.
[9] Daniel Engels, Markku-Juhani O. Saarinen, Peter Schweitzer, Eric M. Smith, "The Hummingbird-2 lightweight authenticated encryption algorithm," In proceedings of 7th International Workshop on RFID. Security and Privacy(RFIDSec 2011), LNCS, Vol. 7055, pp. 19-31, 2012.
[10] Christophe De Cannière, Orr Dunkelman, Miroslav Knežević, "KATAN and KTANTAN-A family of small and efficient hardware-oriented block ciphers," In proceedings of 11th International Workshop on Cryptographic Hardware and Embedded Systems(CHES 2009), LNCS, Vol. 5747, pp. 272-288, 2009.
[11] Xuejia Lai, James L. Massey, "A proposal for a new block encryption standard," In proceedings of Workshop on the Theory and Application of Cryptographic Techniques Aarhus(EUROCRYPT '90), LNCS, Vol. 473, pp. 389-404, 1991.
[12] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, Louis Wingers, "The SIMON and SPECK Block Ciphers on AVR 8-bit Microcontrollers," In proceedings of 3rd International Workshop on Lightweight Cryptography for Security and Privacy(LightSec), LNCS, Vol. 8898, pp. 3-20, 2015.
[13] Daniel J. Bernstein, "The Salsa20 family of stream ciphers," New Stream Cipher Designs, LNCS, Vol, 4986, pp. 84-97, 2008.
[14] Donggeon Lee, Dong-Chan Kim, Daesung Kwon, Howon Kim, "Efficient Hardware Implementation of the Lightweight Block Encryption Algorithm LEA," Sensors, Vol. 14, no. 1, pp. 975-994, 2014.
[15] Mi-Ji Sung, and Kyung-Wook Shin, "A Small-area Hardware Design of 128-bit Lightweight Encryption Alogrithm LEA," The Journal of Korean Institute of Communications and Information Sciences, Vol. 19, no. 4, pp. 888-894, 2015.
[16] TTAK.KO-12.0223, "128-Bit Lightweight Block Cipher LEA," December, 2013.

## 저 자 소 개

**이 철 (Chul Lee)**

2014년 건국대학교 컴퓨터공학과(학사).
2014년~현재 건국대학교 스마트 ICT 융합학과(석사).
관심분야 : FPGA, 병렬프로그래밍 등.

**박 능 수 (Neungsoo Park)**

1991년 연세대학교 전기공학과(학사).
1993년 연세대학교 전기공학과 (석사).
2002년 미국 University of Southern California 전기공학과(공학박사).
2002년~2003년 삼성전자 책임연구원.
2003년~현재 건국대학교 정보통신대학 컴퓨터공학부 교수
관심분야 : 컴퓨터구조, 임베디드 시스템, 병렬시스템, 멀티미디어 컴퓨팅 등