JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 19, No. 7 : 1608~1616 Jul. 2015

IoT 보안 응용을 위한 경량 블록암호 LEA-128/192/256의 효율적인 하드웨어 구현

성미지 · 신경욱*

An Efficient Hardware Implementation of Lightweight Block Cipher LEA-128/192/256 for IoT Security Applications

Mi-Ji Sung · Kyung-Wook Shin*

School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Kyungbuk 730-701, Korea

요 약

128/192/256-비트의 마스터키 길이를 지원하는 경량 블록암호 알고리듬 LEA-128/192/256의 효율적인 하드웨어 설계를 기술한다. 저면적, 저전력 LEA 프로세서 구현을 위해 세 가지 마스터키 길이에 대한 암호/복호 키 스케줄링의 하드웨어 자원이 공유되도록 설계를 최적화하였다. 또한, 키 스케줄러의 병렬 레지스터 구조와 새로운 동작방식을 고안하여 키 스케줄링에 소요되는 클록 수를 감소시켰으며, 이를 통해 암호/복호 동작속도를 20~30% 향상시켰다. 설계된 LEA 프로세서는 FPGA 구현을 통해 하드웨어 동작을 검증하였으며, 113 MHz 클록으로 동작하여 마스터키 길이 128/192/256-비트 모드에서 각각 181/162/109 Mbps의 성능을 갖는 것으로 평가 되었다.

ABSTRACT

This paper describes an efficient hardware implementation of lightweight encryption algorithm LEA-128/192/256 which supports for three master key lengths of 128/192/256-bit. To achieve area-efficient and low-power implementation of LEA crypto- processor, the key scheduler block is optimized to share hardware resources for encryption/decryption key scheduling of three master key lengths. In addition, a parallel register structure and novel operating scheme for key scheduler is devised to reduce clock cycles required for key scheduling, which results in an increase of encryption/decryption speed by 20~30%. The designed LEA crypto-processor has been verified by FPGA implementation. The estimated performances according to master key lengths of 128/192/256-bit are 181/162/109 Mbps, respectively, at 113 MHz clock frequency.

키워드 : 경량 블록암호, LEA, IoT 보안, 정보보안, 비밀키 암호

Key word : Lightweight Encryption Algorithm, LEA, IoT Security, Information Security, Secret Key Encryption

Received 03 April 2015, **Revised** 29 April 2015, **Accepted** 12 May 2015

* **Corresponding Author** Kyung-Wook Shin(E-mail:kwshin@kumoh.ac.kr, Tel:+82-54-478-7427)
School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Kyungbuk 730-701, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2015.19.7.1608 print ISSN: 2234-4772 online ISSN: 2288-4165

ⓒThis is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/li-censes/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright Ⓒ The Korea Institute of Information and Communication Engineering.

Ⅰ. 서 론

다양한 기기들이 인터넷에 연결되어 인간의 조작이나 도움 없이 서로 정보를 교환하고 공유하는 사물인터넷(IoT: Internet of Things) 기술이 빠른 속도로 실용화되고 있으며, 스마트 홈, 스마트 보안, 지능형 교통정보 시스템 등 다양한 분야에 폭 넓게 적용되고 있다. IoT는 복잡하고 이질적인 네트워크에서 다양한 형태의 데이터를 처리하고 전송하므로, 해커나 악의적인 소프트웨어, 바이러스 등에 의해 정보가 유출, 조작, 변조되는 정보보안 위험에 노출될 수 있다. IoT 보안은 센서 네트워크, 인터넷과 유사한 보안 이슈를 가지며 프라이버시, 인증 및 접속 제어, 정보 저장 및 관리 등의 정보보안 이슈들을 포함한다[1,2].

정보보안을 위해 사용되는 대표적인 기술이 암호이다. 암호(Cryptography)는 그리스어에서 기원한 용어로서 비밀기록(secret writing)을 의미하지만, 오늘날에는 다양한 형태의 보안 공격에 안전하도록 정보를 변형하는 기술을 뜻한다. 암호화는 컴퓨터에 저장되어 있거나 네트워크를 통해 전달되는 정보를 제삼자가 가로채어 그 내용을 노출시키거나 의도적으로 조작하는 보안공격으로부터 정보를 보호하기 위한 수단으로 사용되며, IoT 시스템을 구현하는 핵심 기술 요소 중 하나로 평가되고 있다. IoT 보안을 위해서는 기존의 유무선 인터넷 보안과 동일한 대칭키 블록암호 방식과 공개키 암호 방식이 사용된다. 센서 네트워크, RFID 태그, 스마트카드 등 하드웨어/소프트웨어 자원이 제한되는 IoT 네트워크와 단말기의 보안을 위해서는 저전력, 작은 하드웨어/소프트웨어 자원을 소모하는 경량 암호(lightweight cryptography) 알고리듬의 사용이 요구된다[3]. 최근에는 IoT 보안에 적합도록 개발된 HIGHT(HIGh security and light weigHT) [4], LEA (Lightweight Encryption Algorithm) [5], CLEFIA [6], PRESENT [7], mCrypton [8], TEA (Tiny Encryption Algorithm) [9] 등 다양한 경량 블록암호 방식들이 제안되고 있다.

보안 알고리듬은 소프트웨어 또는 하드웨어로 구현되며, 물리적인 안전성과 저전력 소모가 중요한 시스템에서는 전용 하드웨어 구현 방식이 사용된다. 최근에는 IoT, 스마트카드, NFC 보안을 위한 저전력/저면적 하드웨어 구현 결과들이 발표되고 있다[10,11].

본 논문에서는 국가보안기술연구소(NSRI)에서 개발한 128비트 블록암호 알고리듬 LEA를 IoT 환경에 적합하도록 최적화한 LEA 프로세서를 설계하고, FPGA 구현을 통해 하드웨어 동작을 검증하였다. Ⅱ장에서는 블록암호 알고리듬 LEA에 대해 간략히 설명하고, Ⅲ장에서는 LEA-128/192/256 암호/복호 프로세서 설계에 대해 설명한다. 설계된 회로의 기능 검증 및 FPGA 구현에 대해 Ⅳ장에서 기술하며, Ⅴ장에서 결론을 맺는다.

Ⅱ. 경량 블록암호 알고리듬 LEA [5]

블록암호 LEA는 128비트의 평문/암호문 블록을 128/192/256비트의 마스터키로 암호화/복호화하여 128비트의 암호문/평문을 생성하는 대칭키 방식의 암호 알고리듬이다. LEA는 ARX(Addition, Rotation, XOR) 연산을 기반으로 한 Feistel 유사 구조이며, 마스터키의 길이에 따라 24/28/32 라운드의 연산을 통해 암호화/복호화가 이루어진다. 128/192/256비트의 마스터키로부터 생성되는 192비트의 라운드 키가 라운드 변환에 사용된다. 라운드 함수는 32비트 단위의 ARX 연산으로 구성되어 이들 연산을 지원하는 범용 32비트 소프트웨어 플랫폼에서 고속으로 동작한다. 또한 라운드 함수 내부의 ARX 연산 배치는 충분한 안전성을 보장함과 동시에 S-box의 사용을 배제하여 경량 구현이 가능하도록 한다. LEA의 암호화 및 복호화 과정은 그림 1과 같으며, 라운드 키를 생성하는 키 스케줄러와 암호화/복호화를 수행하는 라운드 함수로 구성된다.

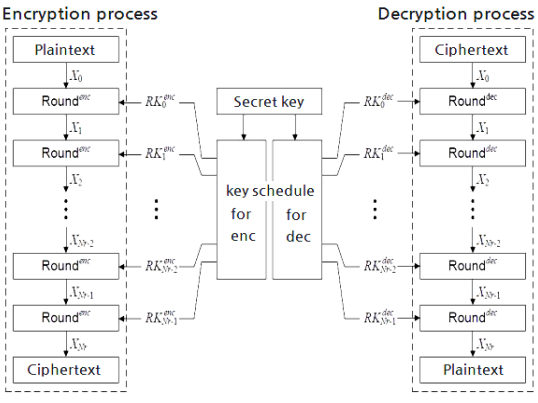


그림 1. LEA 블록암호 알고리듬
Fig. 1 LEA block cipher algorithm

암호화 과정과 복호화 과정은 서로 역순으로 이루어지며, 암호화 과정에서의 모듈로 가산은 복호화 과정에서 모듈로 감산으로 구현되고, 순환이동도 반대방향으로 이루어진다. 암호화 과정은 비밀키 K로부터 Nr개의 192비트 암호화 라운드 키 RK_i^{enc} (단, $0 \le i \le Nr-1$)를 생성하는 암호화 키 스케줄 함수 $KeySchedule_i^{enc}$와 라운드 키 RK_i^{enc} 와 라운드 함수 $Round^{enc}$를 이용하여 128비트 평문 P를 128비트 암호문 C로 변환하는 암호화 함수 $Encrypt$로 구성된다. 암호화 함수의 슈도 코드는 그림 2(a)와 같다. 복호화 과정은 비밀키 K로부터 Nr개의 192비트 복호화 라운드 키 RK_i^{dec} (단, $0 \le i \le Nr-1$)를 생성하는 키 스케줄 함수 $keySchedule_k^{dec}$와 라운드 키 RK_i^{dec}와 라운드 함수 $Round^{dec}$를 이용하여 128비트 암호문 C를 128비트 평문 P로 변환하는 복호화 함수 $Decrypt$로 구성된다. 복호화 함수의 슈도 코드는 그림 2(b)와 같다.

Encryption function : $C \leftarrow \text{Encrypt}(P, RK_0^{enc}, RK_1^{enc}, \cdots, RK_{Nr-1}^{enc})$

Input : 128-bit plaintext P, 192-bit round key $RK_0^{enc}, RK_1^{enc}, \cdots, RK_{Nr-1}^{enc}$

Output : 128-bit ciphertext C

1: $X_0 \leftarrow P$
2: for i = 0 to Nr-1 do
3: $X_{i+1} \leftarrow \text{Round}^{enc}(X_i, RK_i^{enc})$
4: end for
5: $C \leftarrow X_{Nr}$

(a)

Decryption function : $P \leftarrow \text{Decrypt}(C, RK_0^{dec}, RK_1^{dec}, \cdots, RK_{Nr-1}^{dec})$

Input : 128-bit ciphertext C, 192-bit round key $RK_0^{dec}, RK_1^{dec}, \cdots, RK_{Nr-1}^{dec}$

Output : 128-bit plaintext P

1: $X_0 \leftarrow C$
2: for i = 0 to Nr-1 do
3: $X_{i+1} \leftarrow \text{Round}^{dec}(X_i, RK_i^{dec})$
4: end for
5: $P \leftarrow X_{Nr}$

(b)

그림 2. LEA 암호화/복호화의 슈도 코드 (a) 암호화 (b) 복호화
Fig. 2 Pseudo code for encryption/decryption of LEA (a) Encryption (b) Decryption

암호화 과정의 i (단, $0 \le i \le Nr-1$) 번째 라운드연산은 그림 3(a)와 같이 구성된다. 128비트 상태변수 $X_i = (X_i[0], X_i[1], X_i[2], X_i[3])$와 192비트 암호화 라운드 키 $RK_i^{enc} = (RK_i^{enc}[0], RK_i^{enc}[1], \cdots, RK_i^{enc}[5])$ 가산, 32비트 modulo 가산, 비트 순환이동(ROL_9, ROR_5, ROR_3), 32비트 단위의 순환이동을 통해 상태변수 $X_{i+1} = (X_{i+1}[0], X_{i+1}[1], X_{i+1}[2], X_{i+1}[3])$가 생성된다. 복호화 과정의 i (단, $0 \le i \le Nr-1$) 번째 라운드연산은 그림 3(b)와 같이 구성된다. 128비트 상태변수 $X_i = (X_i[0], X_i[1], X_i[2], X_i[3])$와 192비트 라운드 키 $RK_i^{dec} = (RK_i^{dec}[0], RK_i^{dec}[1], \cdots, RK_i^{dec}[5])$의 가산, 32비트 modulo 가산, 비트 순환이동(ROR_9, ROL_5, ROL_3), 32비트 단위의 순환이동을 통해 상태변수 $X_{i+1} = (X_{i+1}[0], X_{i+1}[1], X_{i+1}[2], X_{i+1}[3])$가 생성된다. LEA의 키 스케줄링은 AR(Addition-Rotation) 연산과 32비트 modulo 연산에 사용되는 상수를 생성하는 상수생성부로 구성된다.

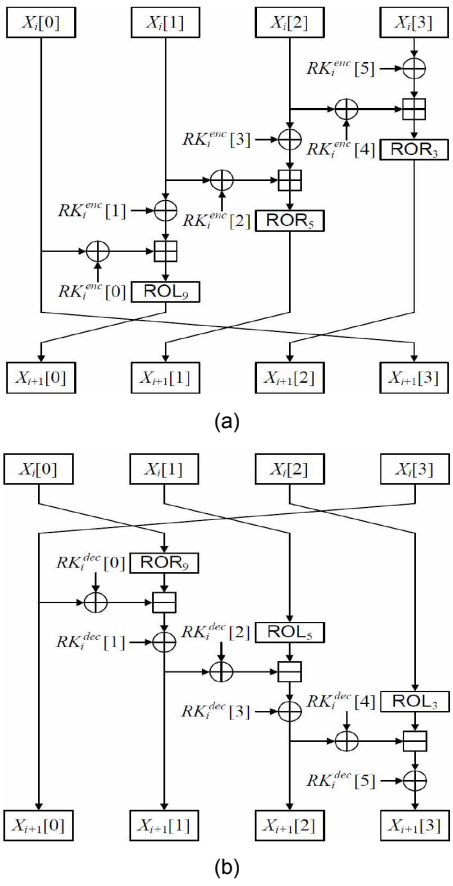


그림 3. LEA 암호화/복호화 라운드 함수 (a) 암호화 (b) 복호화
Fig. 3 Round functions for LEA encryption/decryption (a) Encryption (b) Decryption

마스터키 길이에 따라 생성되는 상수의 개수가 달라지며, 암호화/복호화에 따라 AR 연산의 순서가 반대로 적용된다. 암호화 키 스케줄링은 마스터키 K로부터 암호화 연산에 필요한 Nr개의 192비트 라운드 키 RK_i^{enc} (단, $0 \le i \le Nr-1$)를 생성하는 과정이며, 마스터키와 상수를 32비트 modulo 가산한 후, 비트 순환이동을 통해 이루어진다.

복호화 라운드 키는 $RK_i^{dec} = RK_{Nr-i-1}^{enc}$ 의 관계를 제외하면 암호화 키 생성과 동일한 방법으로 생성된다. LEA- 128/192/256의 암호화 키 스케줄 함수의 슈도 코드는 그림 4와 같다.

Key schedule function for encryption of LEA-128 : $(RK_0^{enc}, \cdots, RK_{23}^{enc}) \leftarrow \text{KeySchedule}_{128}^{enc}(K)$
Input : 128-bit secret key K
Output : 192-bit encryption round key $RK_i^{enc}(0 \le i \le 23)$
$T \leftarrow K$
for i = 0 to 23 do
  $T[0] \leftarrow ROL_1(T[0] \boxplus ROL_i(\delta[i \bmod 4]))$
  $T[1] \leftarrow ROL_3(T[1] \boxplus ROL_{i+1}(\delta[i \bmod 4]))$
  $T[2] \leftarrow ROL_6(T[2] \boxplus ROL_{i+2}(\delta[i \bmod 4]))$
  $T[3] \leftarrow ROL_{11}(T[3] \boxplus ROL_{i+3}(\delta[i \bmod 4]))$
  $RK_i^{enc} \leftarrow (T[0], T[1], T[2], T[1], T[3], T[1])$
end for

(a)

Key schedule function for encryption of LEA-192 : $(RK_0^{enc}, \cdots, RK_{27}^{enc}) \leftarrow \text{KeySchedule}_{192}^{enc}(K)$
Input : 192-bit secret key K
Output : 192-bit encryption round key $RK_i^{enc}(0 \le i \le 27)$
$T \leftarrow K$
for i = 0 to 27 do
  $T[0] \leftarrow ROL_1(T[0] \boxplus ROL_i(\delta[i \bmod 6]))$
  $T[1] \leftarrow ROL_3(T[1] \boxplus ROL_{i+1}(\delta[i \bmod 6]))$
  $T[2] \leftarrow ROL_6(T[2] \boxplus ROL_{i+2}(\delta[i \bmod 6]))$
  $T[3] \leftarrow ROL_{11}(T[3] \boxplus ROL_{i+3}(\delta[i \bmod 6]))$
  $T[4] \leftarrow ROL_{13}(T[4] \boxplus ROL_{i+4}(\delta[i \bmod 6]))$
  $T[5] \leftarrow ROL_{17}(T[5] \boxplus ROL_{i+5}(\delta[i \bmod 6]))$
  $RK_i^{enc} \leftarrow (T[0], T[1], T[2], T[3], T[4], T[5])$
end for

(b)

Key schedule function for encryption of LEA-256 : $(RK_0^{enc}, \cdots, RK_{31}^{enc}) \leftarrow \text{KeySchedule}_{256}^{enc}(K)$
Input : 256-bit secret key K
Output : 192-bit encryption round key $RK_i^{enc}(0 \le i \le 31)$
$T \leftarrow K$
for i = 0 to 31 do
  $T[6i \bmod 8] \leftarrow ROL_1(T[6i \bmod 8] \boxplus ROL_i(\delta[i \bmod 8]))$
  $T[6i+1 \bmod 8] \leftarrow ROL_3(T[6i+1 \bmod 8] \boxplus ROL_{i+1}(\delta[i \bmod 8]))$
  $T[6i+2 \bmod 8] \leftarrow ROL_6(T[6i+2 \bmod 8] \boxplus ROL_{i+2}(\delta[i \bmod 8]))$
  $T[6i+3 \bmod 8] \leftarrow ROL_{11}(T[6i+3 \bmod 8] \boxplus ROL_{i+3}(\delta[i \bmod 8]))$
  $T[6i+4 \bmod 8] \leftarrow ROL_{13}(T[6i+4 \bmod 8] \boxplus ROL_{i+4}(\delta[i \bmod 8]))$
  $T[6i+5 \bmod 8] \leftarrow ROL_{17}(T[6i+5 \bmod 8] \boxplus ROL_{i+5}(\delta[i \bmod 8]))$
  $RK_i^{enc} \leftarrow (T[6i \bmod 8], T[6i+1 \bmod 8], T[6i+2 \bmod 8], T[6i+3 \bmod 8], T[6i+4 \bmod 8], T[6i+5 \bmod 8])$
end for

(c)

그림 4. LEA 암호화 키 스케줄 함수의 슈도 코드 (a) LEA-128 (b) LEA-192 (c) LEA-256
Fig. 4 Pseudo code for LEA encryption key schedule (a) LEA-128 (b) LEA-192 (c) LEA-256

Ⅲ. LEA 암호/복호 회로 설계

본 논문에서는 128비트의 평문/암호문 블록을 128/192/256비트의 마스터키로 암호화/복호화하여 128비트의 암호문/평문을 생성하는 LEA 암호/복호 프로세서를 설계하였다. 설계된 LEA 코어의 전체 구조는 그림 5와 같으며, 라운드 블록, 키 스케줄링 블록, 제어 블록으로 구성된다. 라운드 블록은 마스터키 길이 128/192/256비트에 따라 24/28/32번의 라운드 변환을 통해 암호/복호 연산을 수행하며, 키 스케줄링 블록은 각 라운드 연산에 사용되는 192비트의 라운드 키를 on-the-fly 방식으로 생성한다. 저면적 구현을 위해 라운드 블록을 32비트 회로로 설계하였으며, 입력 핀(data_in)을 공유하여 마스터키와 평문/암호문이 시분할 방식으로 입력되도록 하였다. 또한, 마스터키의 길이에 따른 3가지 모드와 암호화/복호화 연산의 하드웨어 자원이 공유되도록 설계하였다.

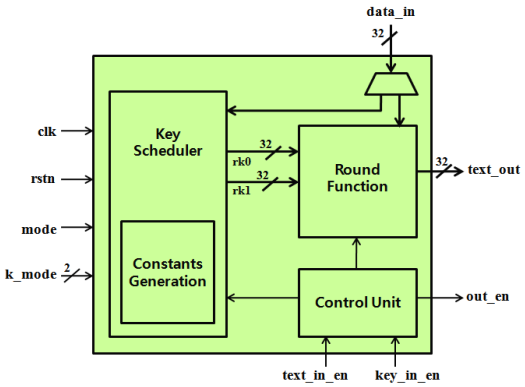


그림 5. LEA 암호/복호 프로세서
Fig. 5 LEA encryption/decryption processor

3.1. 라운드 블록

라운드 블록은 128비트의 평문/암호문과 키 스케줄러에 의해 생성되는 192비트의 라운드 키를 입력받아 라운드 연산을 반복적으로 처리하여 암호/복호 연산을 수행한다. 라운드 블록은 그림 6과 같이 4개의 32비트 레지스터($X0 \sim X3$), XOR 연산기, 32비트 modulo 가산기, 비트 순환이동(ROL, ROR) 등으로 구성된다.

Text_in 포트를 통해 입력되는 128비트의 평문/암호문은 32비트 단위로 4클록 주기에 걸쳐 상태변수 레지스터 $X0$, $X1$, $X2$, $X3$에 저장된다.

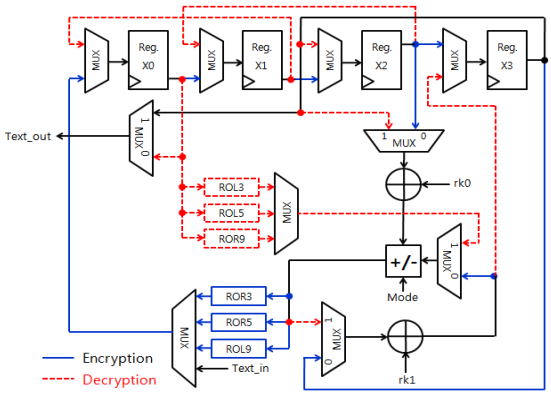


그림 6. 라운드 블록 다이어그램
Fig. 6 Round block diagram

128비트 평문/암호문의 최상위 32비트가 레지스터 $X0$에 입력되고, 매 클록마다 32비트씩 순차적으로 레지스터 $X0$에 입력되어 $X0 \rightarrow X1 \rightarrow X2 \rightarrow X3$으로 이동한다. 따라서 4클록이 지난 후에는 128비트의 평문/암호문이 32비트 단위로 나누어져 4개의 레지스터 $X0$, $X1$, $X2$, $X3$에 저장된다. 128비트의 평문/암호문의 입력이 완료된 후, mode 신호에 따라 mode=0이면 암호화 라운드 연산이 수행되고, mode=1이면 복호화 라운드 연산이 수행된다. 라운드 블록의 내부 레지스터 $X0$, $X1$, $X2$, $X3$, XOR, 32비트 modulo 가산/감산 등이 암호화 동작과 복호화 동작에 공유되어 하드웨어 자원이 최소화되도록 하였다. 라운드 연산은 마스터키의 길이 128/192/256비트에 따라 각각 3/3/4 클록 주기가 소요되며, 라운드 연산이 24/28/32회 반복된 후, 마지막 라운드 연산이 끝나면 암호문/복호문이 출력된다.

가) 암호화 모드 동작

키 스케줄링 블록에서 생성된 라운드 키 rk0와 레지스터 $X2$ 값이 XOR 연산되고, 라운드 키 rk1과 레지스터 $X3$ 값이 XOR 연산된다. XOR 연산의 두 결과 값이 32비트 modulo 가산되고, modulo 가산 결과는 라운드 연산의 클록 순서에 따라 다음과 같이 순환 이동된다. 첫 번째 클록에서는 오른쪽으로 3비트 순환이동(ROR_3) 되고, 두 번째 클록에서는 오른쪽으로 5비트 순환이동(ROR_5) 되며, 세 번째 클록에서는 왼쪽으로 9비트 순환이동(ROL_9) 된다. 순환 이동된 결과 값은 다시 레지스터 $X0$에 저장되며, 동시에 각 레지스터에 저장되어 있던 값은 $X0 \rightarrow X1 \rightarrow X2 \rightarrow X3$으로 이동된다. 이와 같은 연산이 3 클록 주기 동안 반복되어 한 라운드 연산이 완료된다. 결과적으로 레지스터 $X0$, $X1$, $X2$, $X3$에 저장된 값은 ($X2$,$X3$), ($X1$,$X2$), ($X0$,$X1$)이 짝을 이루어 순차적으로 연산이 적용된다.

나) 복호화 모드 동작

복호화 모드에서는 암호화 모드와 반대의 순서로 라운드 키가 사용된다. 암호화 과정의 XOR→Addition→Rotation의 연산 순서는 복호화 과정에서는 Rotation→Subtraction→XOR의 순서로 적용된다. 비트 순환이동도 반대 방향으로 이루어지며, 32비트 modulo 가산은 감산으로 대체된다. 또한, 내부 레지스터의 데이터 이동방향도 $X3 \rightarrow X2 \rightarrow X1 \rightarrow X0$로 암호화 모드와 반대 방향이 된다. 암호화 모드와 동일하게 한 라운드 연산은 3 클록 주기 동안 이루어지며, 레지스터 $X0$의 데이터는 라운드 연산의 클록 순서에 따라 다음과 같이 순환 이동된다. 첫 번째 클록에서는 오른쪽으로 9비트 순환이동(ROR_9) 되고, 두 번째 클록에서는 왼쪽으로 5비트 순환이동(ROL_5) 되며, 세 번째 클록에서는 왼쪽으로 3비트 순환이동(ROL_3) 된다.

3.2. 키 스케줄링 블록

암호/복호 라운드 연산에 사용되는 192비트의 라운드 키는 키 스케줄링 알고리듬에 의해 생성된다.

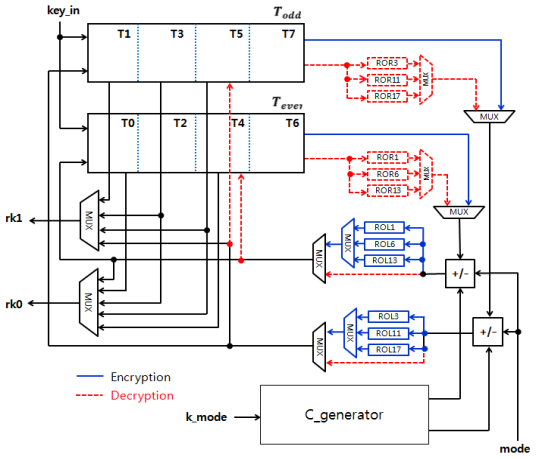


그림 7. 키 스케줄링 블록 다이어그램
Fig. 7 Key scheduling block diagram

본 논문에서 설계된 키 스케줄링 블록은 그림 7과 같으며, 8개의 32비트 내부 레지스터($T0 \sim T7$), XOR 연산기, 32비트 modulo 가산기, 비트 순환이동(ROL, ROR) 등으로 구성되며, 키 생성에 사용되는 상수값 δ를 만드는 상수생성기를 포함한다. 상수값 생성기는 그림 8과 같이 구현하였다.

키 스케줄링 블록은 마스터키의 길이(128/192/256비트)와 암호화/복호화 동작모드에 따라 회로의 동작이 달라진다. 신호 mode=0일 때 암호화 키 스케줄링이 이루어지고, mode=1일 때 복호화 키 스케줄링이 이루어진다. 마스터키의 길이를 지정하는 2비트의 신호 k_mode에 따라 멀티플렉서(MUX)의 동작이 제어되어 LEA-128, LEA-192, LEA-256 모드의 키 스케줄링이 선택적으로 수행된다. 레지스터($T0 \sim T7$), XOR 연산기, 32비트 modulo 가산기/감산기 등이 암호화/복호화와 마스터키 길이에 따른 모드에서 공유되도록 하여 하드웨어 자원이 최소화되도록 하였다.

본 논문에서는 그림 7과 같이 32비트 레지스터 8개($T0 \sim T7$)를 짝수 인덱스 레지스터 $T0, T2, T4, T6$와 홀수 인덱스 레지스터 $T1, T3, T5, T7$로 분할하여 구성하고 이들을 병렬로 동작시키는 방법을 적용하였다. 마스터키는 32비트 단위로 레지스터 $T0$와 $T1$에 교대로 입력되며, 32비트 단위로 $T0 \rightarrow T2 \rightarrow T4 \rightarrow T6$, $T1 \rightarrow T3 \rightarrow T5 \rightarrow T7$으로 시프트 되어 T 레지스터에 저장된다.

LEA-128 모드는 32비트 레지스터 4개를 사용하여 4클록 주기로 구현이 가능하지만, 32비트 레지스터 6개를 사용하여 생성된 키를 저장하고 멀티플렉서를 통해 출력하는 방법을 적용하였다. 본 논문의 방법은 라운드 키 생성에 3클록 주기가 소요되며, 키 스케줄링에 소요되는 클록 주기를 감소시킬 수 있다. 한편, 문헌 [12]의 방법에서는 라운드 키 생성에 4클록 주기가 소요된다. LEA-256 모드의 동작에서는 해당 라운드의 키 스케줄링에 사용되지 않는 중간 변환 값이 레지스터 $T6, T7$에 저장되도록 함으로써 라운드 키 생성에 총 4클록 주기(키 스케줄링을 위한 3클록 주기와 키 시프트/저장을 위한 1클록 주기)가 소요되도록 하였다.

그림 9는 본 논문에서 제안한 방법(그림 7)과 문헌 [12]의 방법에 의한 키 스케줄링의 라운드 연산에 소요되는 클록 수를 비교한 것이다. 그림 9-(a),(b)는 마스터

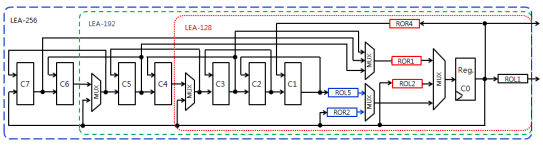


그림 8. 상수값 δ 생성 블록 다이어그램
Fig. 8 Constant δ generation block diagram

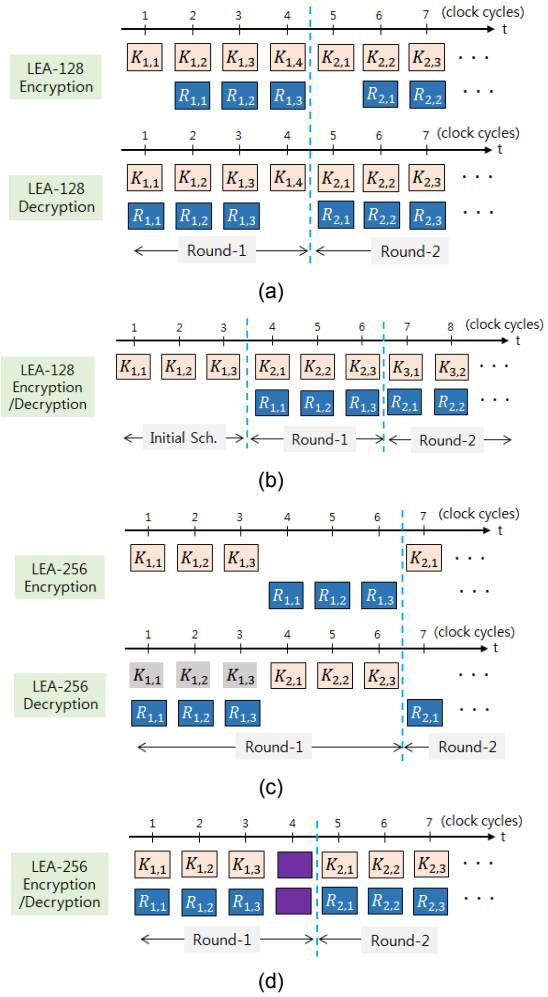


그림 9. 키 스케줄링 및 라운드 변환의 동작 타이밍 (a) 문헌 [12]의 LEA-128 모드의 키 스케줄링 (b) LEA-128 모드의 제안된 키 스케줄링 (c) 문헌 [12]의 LEA-256 모드의 키 스케줄링 (d) LEA-256 모드의 제안된 키 스케줄링
Fig. 9 Operation timing of key scheduling and round transformation (a) key scheduling of LEA-128 mode in [12] (b) proposed key scheduling of LEA-128 mode (c) key scheduling of LEA-256 mode in [12] (d) proposed key scheduling of LEA-256 mode

키가 128-비트인 LEA-128 모드의 동작을 보이고 있다. 문헌 [12]의 방식은 라운드 당 4클록 주기가 필요한 반면에, 본 논문의 방식은 라운드 당 3클록 주기가 소요되어 암호 연산에 20%, 복호 연산에 23%의 속도 개선이 얻어진다. 그림 9-(c),(d)는 마스터키가 256-비트인 LEA-256 모드의 동작을 보이고 있다. 문헌 [12]의 방식은 라운드 당 6클록 주기가 필요한 반면에, 본 논문의 방식은 라운드 당 4클록 주기가 소요되어 암호 연산에 약 33%, 복호 연산에 32%의 속도 개선이 얻어진다.

Ⅳ. 기능검증 및 FPGA 구현

Verilog HDL로 설계된 LEA 코어의 기능검증 결과는 그림 10과 같으며, 128비트의 평문 “30313233 34353637 38393a3b 3c3d3e3f”와 256비트의 마스터키 “0f1e2d3c 4b5a6978 8796a5b4 c3d2e1f0 f0e1d2c3 b4a59687 78695a4b 3c2d1e0f”를 사용한 시뮬레이션 결과를 보이고 있다. 암호화 결과로 암호문 “f6af51d6 c189b147 ca00893a 97e1f927”이 출력되었고, 이를 다시 복호화하여 평문 “30313233 34353637 38393a3b 3c3d3e3f”이 출력되었으며, 설계된 LEA 프로세서의 논리기능이 정상 동작함을 확인하였다.

기능검증이 완료된 LEA 코어는 FPGA 구현을 통해 하드웨어 동작을 검증하였다. 그림 11(a)는 FPGA 보드, UART 인터페이스, 구동 소프트웨어로 구성된 검증시스템 구성도이며, Xilinx Virtex5 XC5VSX-50T 디바이스가 사용되었다. PC에서 입력된 비밀키와 평문/암호문 데이터가 RS232C 통신을 통해 FPGA로 입력되고, FPGA에서 출력되는 암호문/평문 데이터가 표시된다.

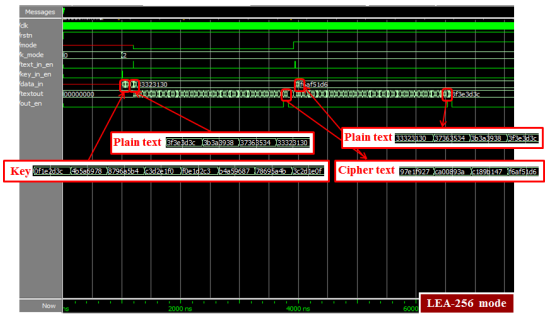
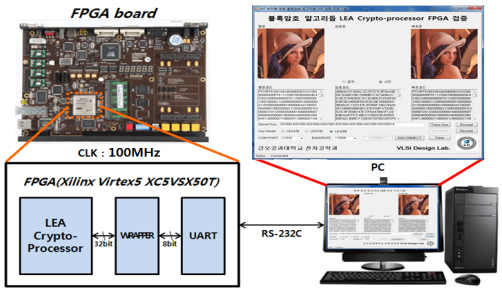


그림 10. LEA 코어의 기능검증 결과
Fig. 10 Simulation result of LEA crypto-core



(a)

(b)

그림 11. LEA 코어의 FPGA 검증 결과 (a) FPGA 검증 시스템 (b) FPGA 검증 결과
Fig. 11 FPGA verification result of LEA core (a) FPGA verification system (b) FPGA verification result

그림 11(b)는 FPGA 검증 결과를 보이고 있다. 평문을 암호화하고, 암호문을 복호화하여 원래의 평문과 일치하는 복호결과가 출력되어 FPGA에 구현된 LEA 프로세서가 올바르게 동작함을 확인하였다. 설계된 LEA 프로세서는 FPGA 합성 결과 2,364 슬라이스로 구현되었다. FPGA에서 최대 113 MHz 클록 주파수로 동작할 때, 128/192/256비트의 마스터키 길이에 따른 암호화

표 1. LEA 코어의 비교
Table. 1 Comparison of LEA cores

		[12]		This work		
		cycles/round	Throughput @ 100 MHz (Mbps) LEA core	cycles/round	Throughput @ 100 MHz (Mbps) LEA core	Including I/O
LEA-128	Enc	4	133	3	170	160
	Dec	4	133	3	177	166
LEA-192	Enc	3	152	3	152	143
	Dec	3	152	3	152	143
LEA-256	Enc	6	66	4	100	97
	Dec	6	66	4	99	96

동작 시 각각 181/162/109 Mbps, 복호화 동작 시 각각 188/162/109 Mbps의 성능을 갖는 것으로 평가되었다.

표 1은 본 논문에서 설계된 LEA 코어와 문헌 [12]에 발표된 사례의 성능 비교를 보이고 있다. 문헌 [12]의 LEA 코어는 단일 마스터키 길이만 지원하는 반면에 본 논문의 설계는 세 가지 키 길이를 동시에 지원하므로, 하드웨어 복잡도에 대한 직접적인 비교는 어렵다. 100 MHz 클록으로 동작하는 경우에, 본 논문의 LEA 코어는 99~177 Mbps의 출력율을 가져 문헌 [12]의 LEA 코어에 비해 30% 정도 높은 성능을 갖는 것으로 평가되었다.

Ⅴ. 결 론

본 논문에서는 한국정보통신기술협회(TTA) 표준으로 등록된 128비트 블록암호 알고리듬 LEA-128/192/256을 하드웨어로 구현하여 동작을 확인하였다. 마스터키 길이에 따른 3가지 동작모드와 암호/복호 연산의 하드웨어 자원이 공유되도록 최적화하여 설계하였다. 키 스케줄러의 구조와 동작방식을 개선하여 암호/복호 동작속도를 20~30% 향상시켰다. 설계된 LEA 프로세서는 Xilinx Vertex5 FPGA에서 최대 113 MHz 클록으로 동작 가능하며, 마스터키 길이 128/192/256-비트 모드에서 각각 181/162/109 Mbps의 성능을 갖는 것으로 평가 되었다. 설계된 LEA-128/192/256 암호/복호 프로세서는 IoT 및 모바일 기기 보안 등과 같이 저전력과 경량화가 요구되는 응용분야의 보안 IP로 활용이 가능하다.

ACKNOWLEDGMENTS

- This work was supported by the Industrial Core Technology Development Program (10049009, Development of Main IPs for IoT and Image-Based Security Low-Power SoC) funded by the Ministry of Trade, Industry & Energy.
- The authors are thankful to IDEC for EDA software support.

REFERENCES

[1] Dong-hui Kim et al, "Security for IoT Service", *Journal of Korea Institute of Communication and Information Services,* vol. 30, no. 8, pp.53, July 2013.

[2] C. Lu, “Overview of Security and Privacy Issues in the Internet of Things”, http://www.cse.wustl.edu/~jain/cse574-14/ftp/security/

[3] T. Eisenbarth, C. Paar, A. Poschmann, S. Kumar and L. Uhsadel, “A Survey of Lightweight Cryptography Implementations,” *IEEE Design & Test of Computers*, vol. 24, no. 6, pp. 522-533, 2007.

[4] Korea Internet & Security Agency, “HIGHT Algorithm Specification”, 2009.

[5] Telecommunications Technology Association, “128-Bit Block Cipher LEA”, TTA Standard, TTAK.KO-12.0223, 2013.

[6] T. Akishita and H. Hiwatari, “Very Compact Hardware Implementations of the Blockcipher CLEFIA”, in Selected Areas in Cryptography－*SAC 2011, ser. LNCS*, vol. 7118, pp. 278-292, Springer-Verlag, 2012.

[7] A. Bogdanov et al., ‘‘PRESENT: An Ultra-Lightweight Block Cipher,’’ *Proc. Workshop Cryptographic Hardware and Embedded Systems (CHES 07), LNCS 4727,* Springer, pp. 450-466, 2007.

[8] C.H. Lim and T. Korkishko, “mCrypton - A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors,” *Proc. of Information Security Applications, LNCS*, vol. 3786, Aug. 2005,pp. 243-258.

[9] D. Wheeler and R. Needham, “TEA, a Tiny Encryption Algorithm,” *Proc. of the Second International Workshop on Fast Software Encryption,* pp. 97-110, 1995.

[10] N. Hanley and M. O’Neill, “Hardware Comparison of the ISO/IEC 29192-2 Block Ciphers”, *2012 IEEE Computer Society Annual Symposium on VLSI,* pp. 57-66, 2012.

[11] S.S.M. AlDabbagh and I.A. Shaikhli, “Lightweight Block Cipher: a Comparative Study,” *Journal of Advanced Computer Science and Technology Research* Vol.2 No.4, pp. 159-165, Nov., 2012.

[12] Donggeon Lee et al, “Efficient Hardware Implementation of the Lightweight Block Encryption Algorithm LEA”, *Sensors*, pp. 982-983, 2014.

성미지(Mi-Ji Sung)

2015년 2월 금오공과대학교 전자공학부(공학사)
※관심분야 : 통신 및 신호처리용 반도체 IP 설계, 정보보호용 반도체 IP 설계

신경욱(Kyung-Wook Shin)

1984년 2월 한국항공대학교 전자공학과(공학사)
1986년 2월 연세대학교대학원 전자공학과(공학석사)
1990년 8월 연세대학교대학원(공학박사)
1990년 9월~1991년 6월 한국전자통신연구소 반도체연구단(선임연구원)
1991년 7월~현재 금오공과대학교 전자공학부(교수)
1995년 8월~1996년 7월 University of Illinois at Urbana-Champaign(방문교수)
2003년 1월~2004년 1월 University of California at San Diego(방문교수)
2013년 2월~2014년 2월 Georgia Institute of Technology(방문교수)
※관심분야 : 통신 및 신호처리용 SoC 설계, 정보보호 SoC 설계, 반도체 IP 설계