Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.3, Jun. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.3.545

저사양 8-bit AVR 프로세서 상에서의 초경량 블록 암호 알고리즘 CHAM 메모리 최적화 구현*

서 화 정†‡
한성대학교

Memory-Efficient Implementation of Ultra-Lightweight Block Cipher Algorithm CHAM on Low-End 8-Bit AVR Processors*

Hwajeong Seo†‡
Hansung University

요 약

간단한 덧셈, 회전연산, 그리고 XOR 연산자로 구성된 초경량 블록 암호 알고리즘 CHAM은 저사양 사물인터넷과 고사양의 플랫폼 상에서 모두 효율적인 구현이 가능하다. 특히 CHAM 블록 암호 알고리즘은 저사양 사물인터넷 플랫폼 상에서 연산속도를 향상시키는 방안에 대해 심도있게 고민된 알고리즘이다. 본 논문에서는 저사양 사물인터넷 플랫폼 8-비트 AVR 상에서 매우 제한적인 프로그램 메모리 공간을 최소로 하면서 연산속도는 극대화하는 방안에 대해 확인해 보도록 한다. 이를 위해 프로그램 코드는 1 라운드 혹은 2 라운드 기반의 부분 반복문을 활용하였으며 라운드 키 접근을 효율화하기 위해 메모리 공간을 정렬하였다. 최소한의 레지스터 활용 및 데이터 업데이트를 통해 성능 향상 및 코드 크기를 최적화하였다. 그 결과 CHAM 64/128, 128/128, 그리고 128/256의 경우 RANK 파라미터 상에서 29.9, 18.0, 그리고 13.4를 달성하였다. 이는 현존하는 블록암호 알고리즘 구현 중 최상의 결과이다.

ABSTRACT

Ultra-lightweight block cipher CHAM, consisting of simple addition, rotation, and eXclusive-or operations, enables the efficient implementations over both low-end and high-end Internet of Things (IoT) platforms. In particular, the CHAM block cipher targets the enhanced computational performance for the low-end IoT platforms. In this paper, we introduce the efficient implementation techniques to minimize the memory consumption and optimize the execution timing over 8-bit AVR IoT platforms. To achieve the higher performance, we exploit the partly iterated expression and arrange the memory alignment. Furthermore, we exploit the optimal number of register and data update. Finally, we achieve the high RANK parameters including 29.9, 18.0, and 13.4 for CHAM 64/128, 128/128, and 128/256, respectively. These are the best implementation results in existing block ciphers.

Keywords: Software Implementation, Internet of Things, Block Cipher, 8-bit AVR Processors

Received(02. 13. 2018), Modified(05. 08. 2018), Accepted(05. 08. 2018)
* 본 연구는 한성대학교 교내학술연구비 지원과제 임.

† 주저자, hwajeong84@gmail.com
‡ 교신저자, hwajeong84@gmail.com(Corresponding author)

I. 서 론

사물인터넷 플랫폼 상에서 유통되는 대량의 정보는 사용자의 개인정보를 포함할 수 있기 때문에 안전한 보안 통신을 기반으로 교환 되어야 한다. 이를 효과적으로 수행하기 위해 활용할 수 있는 기법이 대칭키 암호 알고리즘이며 대표적인 대칭키 암호 알고리즘으로는 AES 블록암호 알고리즘이 있다 [1]. 최근에는 사물인터넷 플랫폼 상에서 효과적으로 구현 가능한 ARX 기반 암호 알고리즘들이 제안되고 있으며 대표적인 알고리즘으로는 LEA, HIGHT, SIMON, SPECK 그리고 CHAM이 있다 [2, 3, 4, 5]. 특히 CHAM 암호 알고리즘은 stateless기반 round key를 사용하는 초경량 암호 알고리즘으로서 저사양 사물인터넷 플랫폼에 매우 효과적인 특징을 가지고 있다. 사물인터넷 플랫폼 상에서의 구현의 경우 활용 가능한 자원이 제한적이기 때문에 연산속도와 더불어 프로그램 코드의 크기 그리고 RAM 사용량을 종합적으로 최적화하는 방안에 대한 연구가 활발히 진행되고 있다.

본 논문에서는 저사양 8-비트 AVR 프로세서 상에서 기존에 제안된 구현 결과물보다 현격히 적은 메모리 공간만을 활용하여 CHAM 암호 알고리즘을 구현하는 방안을 제시한다. 구현된 결과물은 연산속도, 코드 크기, 그리고 RAM 사용량을 종합적으로 평가하는 좌표인 RANK 상에서 기존의 구현 결과보다 높은 성능을 나타낸다.

본 논문의 구성은 다음과 같다. 2장에서는 CHAM 블록 암호 알고리즘과 기존 8-비트 AVR 프로세서 상에서 메모리 최적화 블록암호 알고리즘 구현 결과에 대해 확인해 본다. 3장에서는 제안하는 메모리 최소구현 기법에 대해 확인해 보도록 한다. 4장에서는 제안하는 구현결과의 성능을 평가한다. 5장에서는 본 논문에 대한 결론을 내린다.

II. 관련 연구

2.1 초경량 블록 암호 알고리즘 CHAM

ICISC'17에서 발표된 초경량 블록 암호 알고리즘 CHAM은 저성능 사물인터넷 플랫폼 (8-비트 AVR, 16-비트 MSP430, 32-비트 ARM 프로세서) 상에서 최적화 구현이 가능하다. 뿐만 아니라 하드웨어 크기도 기존의 블록 암호 알고리즘 구현과 비트 시리얼 구현 비교 시 최소 크기로 구현이 가능한 특징을 가진다. 이러한 초경량 특징은 경량 Addition, Rotation, 그리고 eXclusive-or (ARX) 연산에서 비롯된다. 특히 CHAM에서는 키의 상태를 저장하지 않는 stateless 라운드 키 기법을 사용하여 키 저장 공간을 획기적으로 줄이는 데 성공하였다. CHAM은 총 3가지 암호화 모드를 제공하며 각 모드별 파라미터 값은 [표 1]과 같다. CHAM의 암호 알고리즘은 크게 홀수 번째와 짝수 번째 라운드에서 사용되는 오프셋이 상이하며 자세한 암호화 과정은 [그림 1]과 같다. 평문이 4개로 나뉘어서 X[0]부터 X[3]까지 들어가게 된다. X[0]는 카운터와 XOR 연산이 되고 이후에 X[1]은 왼쪽 회전이 한번 취해진 형태로 들어오게 된다. 이는 라운드 키와 XOR된 이후에 X[0]와 덧셈이 취해지고 왼쪽 회전 연산이 8번 취해지게 된다. 이후의 값은 한 워드씩 이동되어 저장된다.

Table 1. List of CHAM ciphers and their parameters, where n, k, r, and w are bit-length of plaintext, bit-length of master key, number of round, and bit-length of word.

cipher	n	k	r	w	k/w
CHAM-64/128	64	128	80	16	8
CHAM-128/128	128	128	80	32	4
CHAM-128/256	128	256	96	32	8

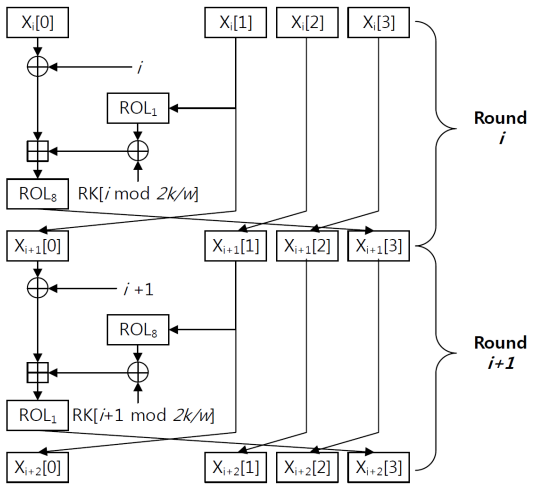

Fig. 1. CHAM block cipher in block diagram, where i, X, RK, and ROL are index, plaintext, roundkey, and rotation, respectively.

2.2 저사양 8-비트 AVR 프로세서

경량 사물인터넷 프로세서 중에서 기본 워드의 크기가 가장 작은 Atmel AVR ATmega128은 8-비트 단위로 연산을 수행한다. 해당 프로세서는 아두이노 우노에 적용되어 활발히 활용되고 있다. 하드웨어 아키텍처는 Harvard 구조를 따르며 동작 주파수는 16MHz이다. 또한 32개의 8-비트 일반 목적 레지스터를 탑재하고 있으며 총 81개의 명령어 셋을 가지고 있다. 플래시 메모리는 128KB이며, 4KB의 EEPROM과 4KB의 SRAM을 탑재하고 있다.

2.3 고정키 암호화

대칭키 암호 구현 결과물을 평가하기 위해서는 해당 암호 구현 결과물이 목표로 하는 운영방식을 선정하는 것이 중요하다 [6]. 운영방식은 크게 고정키 암호화와 가변키 암호화로 생각해 볼 수 있다. 고정키 암호화의 경우 암호화에 사용되는 비밀키가 고정되어 사용되는 경우로써 키스케줄링을 구현물에 포함시키지 않고 모든 라운드 키를 사전에 계산하여 메모리 공간에 저장한 이후에 사용하도록 한다. 암호화가 필요한 데이터가 작은 경우에는 간단한 인증 응용 프로그램에 활용되며 데이터가 큰 경우에는 센서 응용 프로그램과 같이 스트림 데이터 처리에 보다 적합하다. 가변 키 암호화의 경우 비밀키가 주기적으로 변경되는 경우를 의미한다. 따라서 비밀키에 대한 라운드 키가 매번 계산되기 위하여 키 스케줄링을 포함하게 된다. 마찬가지로 데이터가 작은 경우에는 하나의 암호 모듈 형식이 되며 데이터가 큰 경우에는 스트림 형식의 암호화로 생각해 볼 수 있다.

2.4 성능 평가 지표

저성능 사물인터넷 플랫폼 상에서의 블록 암호 알고리즘 구현은 암호화 연산 속도와 더불어 프로그램 코드 크기와 저장 공간의 크기도 같이 고려되어야 한다 [6]. 이를 효과적으로 평가하기 위해 NSA에서 도입한 지표는 RANK로서 (10^6 / CPB) / (FLASH + 2 × RAM)으로 구성된다. 여기서 CPB는 cycle per byte를 의미하며 FLASH는 프로그램 코드 저장 공간 그리고 RAM은 임시 저장 공간을 의미한다. RANK 값은 크면 클수록 효율적인 구현임을 나타낸다.

2.5 이전 CHAM 구현 결과

초경량 블록 암호 알고리즘 CHAM을 저사양 8-비트 프로세서 AVR 상에서 고정키로 최적화 구현한 결과물은 ICISC'17에서 발표되었다 [4]. 기본적인 CHAM 암호 알고리즘의 16 혹은 32-비트 단위의 ARX 연산은 AVR 프로세서의 8-비트 연산자인 'add', 'xor', 'lsl', 그리고 'rol'로 구성하였다. CHAM의 구조는 [그림 1]과 같이 홀수 번째와 짝수 번째 라운드에서 상이한 연산 구조를 가지며 이는 지속적으로 반복되게 된다. 해당 논문에서는 4 라운드를 어셈블리 언어로 구현하였으며 이를 반복적으로 수행하는 구조로 구현되었다. 4 라운드만을 구현할 경우 전체 코드 크기는 CHAM 64/128과 CHAM 128/128의 경우 약 1/20 그리고 CHAM 128/256의 경우 약 1/24의 크기로 줄여서 나타낼 수 있는 특징을 가진다.

III. 제안하는 메모리 최적화 구현

메모리 사용량을 줄이기 위해서는 ICISC'17 상에서의 구현과 같이 프로그램 코드의 양을 줄이고 반복문을 이용하여 전체 연산을 재연하면 된다. ICISC'17에서의 구현 결과가 4번의 라운드를 반복하는 결과였다면 본 논문에서 제안하고자 하는 구현은 1번 그리고 2번의 라운드를 어셈블리로 구현하고 이를 반복 수행하는 것이다. 먼저 두 구현 모두에 적용된 기법을 살펴보면 다음과 같다.

3.1 라운드 키 접근 최적화

현재 8-비트 AVR 프로세서 상에서 메모리에 대한 접근은 2 개의 8-비트 레지스터를 합쳐서 주소에 접근하게 된다. 따라서 메모리에 대한 접근을 위해서는 2 개의 8-비트 레지스터를 동시에 업데이트를 해주어야 하는 특징을 가진다. 하지만 라운드 키는 CHAM 64/128와 CHAM 128/128의 경우에는 32 바이트이며 CHAM 128/256의 경우에는 64 바이트가 사용된다. 해당 메모리에 대한 접근 범위는 각각 32 혹은 64를 최대치로 가지므로 한 바이트의 범위 안(256)에서 모든 메모리 접근 범위 설정이 가능하다. 이는 라운드 키를 메모리 주소에서 하위 주소가 0x00번지가 오도록 정렬하면 라운드 키에 대한 접근을 1 바이트 오프셋 연산으로 해결가능함을 의미한다. 라

운드 키 접근을 위해 필요했던 2 바이트 단위의 오프셋 정렬이 아닌 1 바이트 단위의 오프셋 정렬로 최적화함으로써 라운드 키 접근 시마다 1 클록 사이클이 감소하는 효과를 얻을 수 있다. 이는 〔그림 1〕에서 *RK[i mod 2k/w]*로 표기된 라운드 키를 불러오는 부분을 의미한다. 만약 Z 포인터 (R31, R30)를 사용할 경우 R31은 항상 고정으로 사용하고 R30만을 업데이트하도록 한다.

3.2 카운터 최적화

〔그림 1〕에서와 같이 각 라운드에서는 평문 X〔0〕 값이 현재 카운터와 XOR 연산을 수행하게 된다. CHAM 64/128 그리고 CHAM 128/128의 카운터 최대값은 80이며 CHAM 128/256의 카운터 최대값은 96이 된다. 따라서 1 바이트 카운터만을 활용하여 효과적으로 카운터를 관리하였다.

3.3 메모리 접근 최적화

메모리에 대한 접근 이전 그리고 이후에 그 다음에 연산할 메모리 주소로 자동으로 이동해주는 연산자를 활용할 경우 추가적인 연산자 없이 메모리 주소 계산이 가능하다. 따라서 이를 이용하면 〔그림 1〕에서 평문 (*X*)과 라운드 키 (*RK*) 접근 시 모든 메모리 접근은 메모리 주소 자동 계산 명령어 (post incremental)를 활용하여 수행하였다. 3.4장과 3.5장에서는 1라운드와 2라운드에 각각 적용된 기법에 대해 확인해 보도록 한다.

3.4 CHAM ver. 1

첫 번째 구현 결과물은 전체 라운드를 하나의 라운드만 구현 후 이를 반복적으로 수행하는 방법이다. CHAM은 〔그림 1〕에서와 같이 2 라운드가 반복적으로 수행되는 구조를 가지고 있기 때문에 1 라운드로 2 라운드를 구현하기 위해서는 2라운드를 포함하는 코드를 작성해 주어야 한다. 특히 회전 연산의 오프셋이 홀수 번째와 짝수 번째가 서로 상이한 특징을 가진다. 이를 효과적으로 수행하기 위해 〔표 2〕 그리고 〔표 3〕과 같이 하나의 명령어 셋을 통해서 홀수와 짝수 오프셋을 한 번에 결정하였다. 오프셋이 결정되고 나면 해당 오프셋에 따라 회전 연산이 수행된다. 〔표 4〕에는 오프셋에 따른 회전연산을 16-비트 단위로 취

Table 2. program code for offset #1, where odd and even rounds are set to 7 and 0, respectively.

Program	Odd round (R0 is odd)	Even round (R0 is even)
ANDI R0, 1	0x01	0x00
DEC R0	0x00	0xFF
COM R0	0xFF	0x00
ANDI R0, 7	0x07	0x00

Table 3. program code for offset #2, where odd and even rounds are set to 0 and 7, respectively.

Program	Odd round (R0 is odd)	Even round (R0 is even)
ANDI R0, 1	0x01	0x00
DEC R0	0x00	0xFF
ANDI R0, 7	0x00	0x07

Table 4. rotation by offset

```
STEP:
        LSL X0
        ROL X1
        ADC X0, ZERO
        DEC COUNT
        CPI COUNT, 0
        BRGE STEP
```

하는 코드가 기술되어 있다. 연산은 COUNT 수에 따라 반복되는 구조로 설계되었다.

해당 기법의 특징은 CHAM 암호 알고리즘 기법의 장점이었던 8-비트 기반 회전 연산 생략의 장점을 취할 수 없다는 점이다. 이는 라운드의 수를 반으로 줄임으로써 오프셋 1과 8을 한 번에 나타내어야 하는 점 때문에 발생한 성능 저하라고 할 수 있다.

3.5 CHAM ver. 2

두 번째 구현 결과물은 전체 라운드를 두 개의 라운드만 구현 후 이를 반복적으로 수행하여 나타내는 방법이다. CHAM ver. 1과 비교하여 가장 크게 다른 점은 CHAM의 강점인 8-비트 회전 연산 생략이 가능하다는 것이다. 회전 연산을 생략하는 대신 레지스터 주소 중에 8-비트 회전이 된 부분에 대해서는 연산 시 회전된 형태로 연산을 지속적으로 수행하는 방법을 취했다. 또한 〔그림 1〕에서 1 라운드 마다 워

드 단위로 회전을 취해야 하지만 이를 효율적으로 수행하기 위해 2 라운드가 끝나고 난 이후에 2 워드 단위로 회전을 한 번에 수행하였다. 이때 회전을 수행하면서 8-비트 회전 연산이 수행된 워드를 원래 비트 배열로 변경해 주는 작업도 동시에 수행하였다. [표 5]를 보면 X0과 X1는 8-비트 회전을 취해주며 워드를 회전하고 나머지는 16-비트 워드 단위로만 회전을 수행한다. 특히 MOVW 연산의 경우 1 클록에 워드 전송이 가능한 장점을 가진다.

각 라운드마다 라운드 키의 값이 특정한 범위를 넘어서지 않는지 확인하기 위해 라운드 키의 하위 주소와 마스킹 연산을 하여 CHAM 64/128 그리고 CHAM 128/128인 경우에는 오프셋이 32를 넘어서지 않도록 하고 CHAM 128/256인 경우에는 오프셋이 64를 넘어서지 않도록 하였다. 카운터는 0부터 시작하기 때문에 짝수 번째 라운드에서 32 혹은 64를 넘어서는 오프셋이 나타나게 된다. 따라서 짝수 번째 라운드에서만 오프셋과 마스킹을 하는 연산자 (ANDI OFFSET, 31 혹은 ANDI OFFSET, 63)를 넣어 수행하고 홀수 번째에서는 마스킹을 수행하지 않아 코드 크기와 연산을 효율적으로 관리하는 것이 가능하도록 하였다.

Table 5. register alignment

```
MOV X4, X1
MOV X5, X0
MOVW X0, X2
MOVW X2, X6
MOVW X6, X8
```

IV. 성능 평가

[표 6]에는 고정키 기반 암호화를 저사양 8-비트 AVR 프로세서 상에서 구현했을 때 얻을 수 있는 RANK 파라미터를 비교하여 나타내고 있다. 본 논문에서 구현한 1 라운드 기반 그리고 2 라운드 기반 구현 결과는 각각 CHAM ver. 1 그리고 CHAM ver. 2로 나타나 있다. 64-비트 평문과 128-비트 비밀키를 사용하는 경우 이전 CHAM의 구현 결과는 27.9 RANK를 달성하였다. 하지만 본 논문에서는 이전 구현 결과에서 프로그램 코드 크기를 줄이고 연산속도를 최적화 함으로써 보다 나은 29.9 RANK를 2 라운드 기반 구현에서 달성하였다. 이는 해당 블록 암호 알고리즘 상에서 가장 높은 성능을 나타냈던 SPECK의 RANK 값 29.8을 넘어서는 결과임을 나타낸다. 따라서 CHAM 암호 알고리즘은 SPECK 보다 나은 결과 도출이 가능함을 확인할 수 있다. 해당 구현 결과는 128-비트 평문과 128-비트 비밀키를 사용하는 경우와 128-비트 평문과 256-비트 비밀키를 사용하는 경우에서 모두 동일하게 나타나며 모두 최고로 높은 RANK 값을 기록하고 있음을 확인할 수 있다.

1 라운드 기반 구현의 경우 1라운드만을 구현에 프로그램으로 작성함으로써 코드 크기를 획기적으로 줄일 수 있을 것으로 기대되었다. 하지만 CHAM의 경우 2 라운드마다 연산이 반복되는 매우 간단한 구조로 되어 있어서 1 라운드 프로그램 코드로 2 라운드를 범용적으로 동작시키는 코드는 연산 속도와 코드 크기 모두에서 좋지 않은 성능을 도출함을 확인할 수 있었다. 유일하게 128-비트 평문과 128-비트 비밀키를 사용하는 경우와 128-비트 평문과 256-비트 비밀키를 사용하는 경우에서 프로그램 코드가 2 바이트 그리고 RAM이 1 바이트 씩 최적화가 가능하였지만

Table 6. Performance comparison using the rank metric on 8-bit AVR processors, under the fixed-key scenario.

Alg.	ROM (byte)	RAM (byte)	Time (cpb)	RANK
64-bit plaintext / 128-bit secret key				
Proposed CHAM ver. 2	**152**	**3**	**211**	**29.9**
SPECK [3]	218	0	154	29.8
CHAM [4]	202	3	172	27.9
SIMON [3]	290	0	253	13.6
HIGHT [3]	336	0	311	9.6
Proposed CHAM ver. 1	**166**	**2**	**966**	**6.1**
128-bit plaintext / 128-bit secret key				
Proposed CHAM ver. 2	**270**	**13**	**187**	**18.0**
CHAM [4]	362	16	148	17.1
SPECK [3]	460	0	171	12.7
LEA [4]	754	17	203	6.3
Proposed CHAM ver. 1	**270**	**12**	**657**	**5.2**
128-bit plaintext / 256-bit secret key				
Proposed CHAM ver. 2	**302**	**13**	**223**	**13.6**
CHAM [4]	396	16	177	13.2
SPECK [6]	476	0	181	11.6
Proposed CHAM ver. 1	**302**	**12**	**786**	**3.9**

연산 속도가 느려지는 단점으로 인해 높은 RANK 값은 달성할 수 없다. 하지만 3.4 장에서 제시한 오프셋 설정 프로그램 루틴은 다른 시큐어 프로그램 코딩에 활용이 가능하다.

V. 결 론

본 논문에서는 초경량 블록 암호 알고리즘 CHAM을 저사양 8-비트 AVR 프로세서 상에서 메모리 효율 기반 구현을 함으로써 고정키 암호화 상에서 가장 높은 RANK 값을 달성하였다. 해당 결과는 초경량 구현이 중요한 사물인터넷 환경 상에서 주어진 자원(연산 속도, 프로그램 코드, 램)을 가장 효율적으로 활용할 수 있는 기법으로써 LEA, HIGHT, SPECK, 그리고 SIMON와 같은 ARX 기반 암호화 구현에 바로 적용이 가능하다. 추후 연구로는 연산 속도를 최우선으로 하는 기법을 적용하여 고속 암호화 연산이 가능한 방법에 대해 확인해 볼 계획이다.

References

[1] Standard, N. F, "Announcing the advanced encryption standard (AES)," *Federal Information Processing Standards Publication*, 197, pp. 1-51, 2001.

[2] Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu, and Dong-Geon Lee, "LEA: A 128-bit block cipher for fast encryption on common processors," *In International Workshop on Information Security Applications (WISA'13)*, pp. 3-27, 2013.

[3] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, Louis Wingers, "The SIMON and SPECK lightweight block ciphers," *In Design Automation Conference (DAC'15)*, pp. 1-6, 2015.

[4] Bonwook Koo, Dongyoung Roh, Hyeonjin Kim, Younghoon Jung, Dong-Geon Lee, and Daesung Kwon, "CHAM: A Family of Lightweight Block Ciphers for Resource-Constrained Devices," *International Conference on Information Security and Cryptology (ICISC'17)*, 2017.

[5] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bon-Seok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee, "HIGHT: A new block cipher suitable for low-resource device," *In International Workshop on Cryptographic Hardware and Embedded Systems (CHES'06)*, pp. 46-59, 2006.

[6] 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 International Workshop on Lightweight Cryptography for Security and Privacy*, pp. 3-20, 2014.

〈저자소개〉

서 화 정 (Hwa-jeong Seo) 종신회원
2010년 2월: 부산대학교 컴퓨터공학과 학사 졸업
2012년 2월: 부산대학교 컴퓨터공학과 석사 졸업
2016년 2월: 부산대학교 컴퓨터공학과 박사 졸업
2015년 4월~5월: 난양공대 인턴쉽
2016년 1월~2017년 3월: 싱가포르 과학기술청 연구원
2017년 4월~현재: 한성대학교 조교수
〈관심분야〉 정보보호, 암호화 구현, IoT