Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.33, NO.2, Apr. 2023 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2023.33.2.165

32-bit RISC-V 프로세서 상에서의 경량 블록 암호 SIMECK, SIMON 카운터 운용 모드 최적 구현*

심 민 주,[1†] 권 혁 동,[1] 오 유 진,[1] 송 민 호,[1] 서 화 정[2‡]
[1,2]한성대학교 (대학원생, 교수)

Optimized Implementation of Lightweight Block Cipher SIMECK and SIMON Counter Operation Mode on 32-Bit RISC-V Processors*

Min-Joo Sim,[1†] Hyeok-Dong Kwon,[1] Yu-Jin Oh,[1] Min-Ho Song,[1] Hwa-Jeong Seo[2‡]
[1,2]Hansung University (Graduate student, Professor)

요 약

본 논문에서는 32-bit RISC-V 프로세서 상에서 경량 블록 암호인 SIMECK과 SIMON의 카운터 운용 모드에 대한 최적 구현을 제안한다. CTR 운용 모드의 특징을 활용하여 일부 값을 사전 연산하는 라운드 함수 최적화, 단일 평문 최적화와 2개의 평문 병렬 최적화를 제안한다. RISC-V 상에서의 SIMECK과 SIMON에 대한 선행 연구 결과가 존재하지 않기 때문에 단일 평문 최적화와 2개의 평문 병렬 최적화 구현물에 대해 사전 연산 기법이 적용된 구현물과 사전 연산이 적용되지 않은 구현물의 성능을 비교하였다. 결과적으로, 사전 연산 기법이 적용된 구현물은 사전 연산이 적용되지 않은 구현물 대비 모두 1%의 성능 향상을 확인하였다.

ABSTRACT

In this paper, we propose an optimal implementation of lightweight block ciphers, SIMECK and SIMON counter operation mode, on a 32-bit RISC-V processor. Utilizing the characteristics of the CTR operating mode, we propose round function optimization that precomputes some values, single plaintext optimization and two plaintext parallel optimization. Since there are no previous research results on SIMECK and SIMON on RISC-V, we compared the performance of implementations with and without precomputation techniques for single plaintext optimization and two plaintext parallel optimization implementations. As a result, the implementations to which the precomputation technique was applied showed a performance improvement of 1% compared to the implementations to which precomputation was not applied.

Keywords: SIMECK, SIMON, RISC-V, CTR, Optimized implementation

Received(01. 05. 2023), Accepted(02. 02. 2023)
* 본 논문은 2022년도 한국정보보호학회 호남지부 학술대회에 발표한 우수논문을 개선 및 확장한 것임.
* This work was partly supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2020R1F1A1048478, 60%) and this work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00264, Research on Blockchain Security Technology for IoT Services, 20%) and this work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00627, Development of Lightweight BIoT technology for Highly Constrained Devices, 20%).
† 주저자, minjoos9797@gmail.com
‡ 교신저자, hwajeong84@gmail.com(Corresponding author)

I. 서 론

사물인터넷이 빠르게 발전됨에 따라, 사물인터넷에 사용되는 저사양 프로세서들의 보안은 필수적으로 요구된다. 국내외에서는 공모전 등을 통해 새로운 경량 암호에 대한 개발이 진행되고 있고, 경량 암호에 대한 최적 구현 연구도 활발히 진행되고 있다. SIMECK은 2015년에 발표된 경량 블록 암호이다[1]. SIMECK에 대한 최적 구현 연구는 8-bit AVR, 16-bit MSP430, 32-bit ARM 마이크로컨트롤러 등 다양한 환경에서 활발히 진행되었다[2~7]. 하지만, 32-bit RISC-V 상에서의 SIMECK 최적 구현 연구는 진행되지 않았으며, 저사양 프로세서 상에서의 SIMECK-CTR에 대한 최적 구현 연구도 현재까지 진행되지 않았다. SIMON은 2013년에 발표된 경량 블록 암호이다[8,9]. SIMON은 다양한 저사양 프로세서 상에서의 최적화 연구가 수행되었다[10~12]. 특히, SIMON-CTR에 대한 8-bit AVR 최적 구현 연구가 제안되었다[13]. 따라서, 본 논문에서는 32-bit RISC-V 상에서의 SIMECK과 SIMON에 대한 단일 평문 최적 구현과 2개의 평문에 대한 최적 구현 기법을 제안하고, 각각에 대해 카운터 운용모드에 대한 최적화를 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 경량 블록 암호 SIMECK과 SIMON에 대해 알아보고, 32-bit RISC-V 프로세서와 카운터 운용 모드에 대해 알아본다. 3장에서는 본 논문에서 제안하는 SIMECK, SIMON에 대한 최적 구현 기법에 대해 설명한다. 4장에서는 제안 기법이 적용된 구현물에 대한 성능 평가를 진행한다. 마지막으로 5장에서는 본 논문의 결론을 내린다.

II. 관련 연구

2.1 SIMECK

SIMECK은 CHES'15에서 발표된 Feistel 구조인 경량 블록 암호 알고리즘이다. SIMECK은 NSA (National Security Agency)에서 개발한 경량 블록 암호인 SIMON의 라운드 함수를 수정하고, SPECK과 유사한 키 스케줄링을 사용한다. SIMECK의 각각의 암호화에 대한 파라미터는 Table 1. 과 같다.

Table 1. List of SIMECK ciphers and their parameters.

Cipher	n	k	r	w
SIMECK-32/64	32	64	32	16
SIMECK-48/96	48	96	36	24
SIMECK-64/128	64	128	44	32

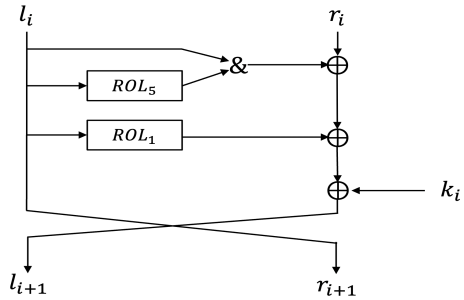

Fig. 1. Round function of SIMECK.

SIMECK의 알고리즘은 AND, Rotation, eXclusive-OR 연산으로 이뤄져 있다. SIMECK의 라운드 함수는 Fig. 1.과 같다.

2.2 SIMON

SIMON은 2013년에 NSA에서 개발한 Feistel 구조의 경량 블록 암호로, 하드웨어 구현에 최적화되어 있다. SIMON의 각각의 암호화에 대한 파라미터는 Table 2. 와 같다.

Table 2. List of SIMON ciphers and their parameters.

Block Size	Key Size	Rounds	Word Size(m)
32	64	32	16
48	72	36	24
	96		
64	96	42	32
	128	44	
96	96	52	48
	144	54	
128	128	68	64
	192	69	
	256	72	

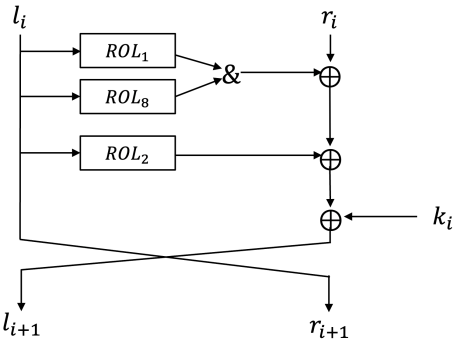

Fig. 2. Round function of SIMON.

SIMON의 라운드 함수는 2개의 블록으로 나누어서 반복적으로 라운드 함수가 진행되며, AND, Rotation, eXclusive-OR 연산으로 이뤄져 있다. SIMON의 라운드 함수는 Fig. 2. 와 같다.

2.3 RISC-V Processors

RISC-V는 32-bit 구조인 RV32I와 64-bit 구조인 RV64I가 있다[14]. RV32I는 32-bit 32개의 레지스터를 제공하며, RISC-V 프로세서 중 32-bit 구조인 RV32I는 32개의 32-bit 레지스터를 제공한다. x0 레지스터는 항상 0의 값을 갖는 레지스터이고, 이외 x1~x4의 레지스터는 특수 용도에 사용되는 레지스터로 지정되어 있다. RISC-V 프로세서의 레지스터 용도는 Table 3.과 같다. 본 논문에서 사용하는 RISC-V 명령어는 Table 4.와 같다.

Table 3. Purpose of registers in RISC-V processors.

Register	Purpose	Saver
x0	zero register	
x1(ra)	return address	
x2(sp)	stack pointer	callee
x3(gp)	global pointer	
x4(tp)	thread pointer	
a0~a7	function arguments and return value	
s0~s11	saved registers	callee
t0~t6	temporal registers	

2.4 카운터(CTR) 운용 모드

블록 암호는 ECB(Electronic Code Block), CBC(Cipher Block Chaining), CFB(Cipher FeedBack), OFB(Output FeedBack), CTR(Counter) 운용 모드가 존재한다.

이 중 CTR 운용 모드는 임의의 고정된 상수 nonce 값과 블록을 암호화할 때마다 1씩 증가하는 카운터를 결합한 값을 입력 값으로 사용해 암호화하여 키 스트림을 만든다. 그리고 생성된 키 스트림과 평문을 XOR 연산을 한 결과가 암호문이 되는 특징을 갖고 있다.

Table 4. Instruction of RISC-V processors.

Instruction	Description
MV	Move
ADD	Add
ADDI	Add Immediate
XOR	Exclusive or
OR	Inclusive or
SRLI	Shift right logical immediate
SLLI	Shift left logical immediate
LW	Load word
SW	Store word
LH	Load half-word
SH	Store half-word

III. 제안 기법

본 장에서는 제안하는 기법인 단일 평문과 2개의 평문 병렬에 대해 최적화된 SIMECK-32/64, SIMECK-64/128, SIMON-64/128에 대한 카운터 운용 모드 최적화에 대해 설명한다. SIMECK과 SIMON의 단일 평문 최적 구현과 2개의 평문 병렬 최적 구현을 설명하고, 사전 연산을 활용한 최적화에 대해 설명한다.

SIMECK과 SIMON은 두 개의 블록에 대해 연산이 진행된다. SIMECK과 SIMON의 라운드 수는 모두 짝수이기 때문에 모든 연산이 끝난 블록의 위치는 처음과 동일하다. 이러한 라운드 함수의 특징을 활용해 라운드 함수 마지막에 진행되는 블록의 이

동을 생략하였다. 결과적으로, 한 라운드에서 두 개의 블록 이동을 위한 2번의 연산을 생략하였다.

3.1 SIMECK과 SIMON 단일 평문 최적화 구현

SIMECK-32/64는 워드 사이즈가 16-bit이다. 따라서, 32-bit 레지스터의 절반인 16-bit만 사용하여 구현하였다. SIMECK-64/128과 SIMON-64/128은 워드 사이즈가 32-bit이기 때문에 32-bit 레지스터를 모두 활용하여 사용하여 구현하였다. RISC-V에서는 로테이션 명령어를 제공하지 않아 쉬프트 연산과 XOR 연산을 이용하여 구현하였다.

3.2 SIMECK과 SIMON 2개의 평문 병렬 최적화 구현

3.2.1 SIMECK-32/64

16-bit 블록 단위로 연산하는 SIMECK-32/64를 구현하기 위해 32-bit 레지스터의 절반만 사용하여 하나의 평문만 암호화하는 것은 효율적이지 않다. 따라서, 32-bit 레지스터를 모두 활용하기 위해 2개의 평문에 대한 암호화를 진행하는 병렬 구현을 수행하였다. Fig. 3. 과 같이 레지스터 내부 정렬을 수행하였다.

단일 평문 구현과 달리 병렬 구현에서는 섞이지 않아야 하는 서로 다른 두 개의 16-bit 값이 연속으로 정렬되어 하나의 레지스터에 위치되어 있다. 따라서, 로테이션 연산을 하기 위해서는 서로 다른 두 개의 값이 섞이지 않아야 한다. 하지만, RISC-V에서는 로테이션 명령어를 제공하지 않아 쉬프트 연산과 XOR 연산을 이용하여 구현해야 한다. 쉬프트 연산을 할 때, 서로 다른 값들이 섞이게 되는데 이를 막기 위해 AND 연산을 사용하여 섞이는 위치의 값을 0으로 바꿔주고, 섞이는 위치에 대한 값들을 temp로 사용하는 레지스터에 따로 저장해 준다. 쉬프트 연산을 완료한 후, temp에 저장된 값을 XOR 연산하여 로테이션을 구현하는 기법을 Fig. 4. 와 같이 구현하였다.

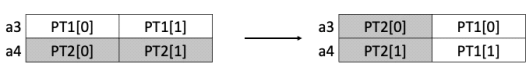

Fig. 3. Register internal alignment in SIMECK-32/64.

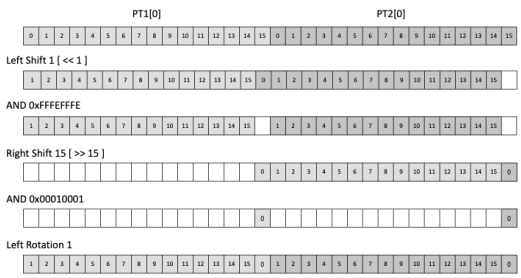

Fig. 4. Process of Left Rotation 1 in SIMECK-32/64.

3.2.2 SIMECK-64/128

32-bit 블록 단위로 연산하는 SIMECK-64/128을 구현하기 위해서 하나의 레지스터에 서로 다른 평문의 상위 16-bit과 하위 16-bit를 분리하여 위치하게 하기 위해 Fig. 5. 와 같이 레지스터 내부 정렬을 수행하였다.

단일 평문 구현과 SIMECK-32/64 병렬 구현은 왼쪽 쉬프트 연산과 오른쪽 쉬프트 연산한다.

하지만 SIMECK-64/128의 경우, 왼쪽 쉬프트 연산만 수행한다. 하위 비트가 위치한 레지스터를 왼쪽 쉬프트 연산하여 없어지는 값은 상위 비트가 위치한 레지스터에 위치해야 한다. 그리고 상위 비트에 위치한 값도 동일하게 하위 비트에 위치하여야 하기 때문에 쉬프트 연산을 하기 전에 해당 값들을 temp에 저장한다. temp에 저장된 값들은 로테이션 연산 이후에 로테이션 연산으로 값이 섞이는 위치에 해당되는 값이기 때문에 오른쪽 쉬프트 연산을 해준다.

SIMECK-32/64 병렬 구현과 동일하게 서로 다른 두 개의 16-bit 값이 연속으로 정렬되어 있기 때문에 이 값들이 서로 섞이지 않게 구현하였다. temp에 위치한 값은 하위 비트에 대한 값은 상위 비트에 상위 비트에 대한 값은 하위 비트에 값을 XOR 연산하여 Fig. 6. 과 같이 구현하였다. 이 기

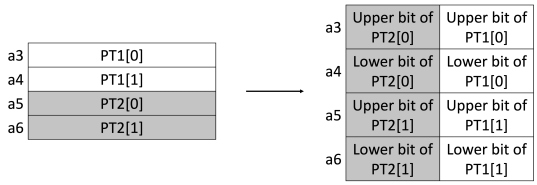

Fig. 5. Register internal alignment in SIMECK-64/128.

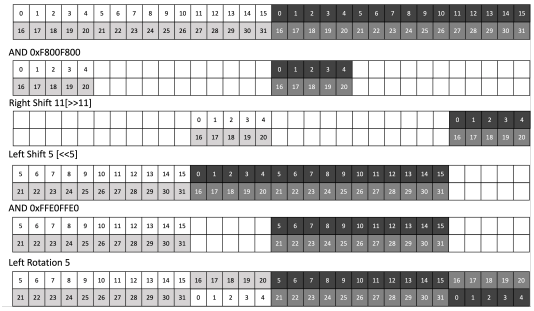

Fig. 6. Process of Left Rotation 5 in SIMECK-64/128.

법을 사용하면, 워드 사이즈가 32-bit인 암호의 병렬 구현 시 효율적인 로테이션 연산이 가능하다.

3.2.3 SIMON-64/128

32-bit 블록 단위로 연산을 수행하는 SIMON-64/ 128은 SIMECK-64/128과 구조가 유사하다. 따라서, SIMECK-64/128과 동일하게 레지스터 내부 정렬을 수행 후, 구현 기법은 SIMECK-64/128과 동일하게 구현하였다.

3.3 SIMECK과 SIMON 카운터 운용 모드 최적화 구현

카운터 운용 모드는 고정된 nonce 블록이 카운터 블록에 영향을 받기 전까지에 해당되는 연산에 대해 사전 연산이 가능하다. 2개의 블록으로 이뤄진 SIMECK에 대해 사전 연산은 카운터 블록에 영향 받기 전까지인 1라운드의 일부만 가능하다.

따라서, Fig. 1. 과 같은 구조를 가진 SIMECK

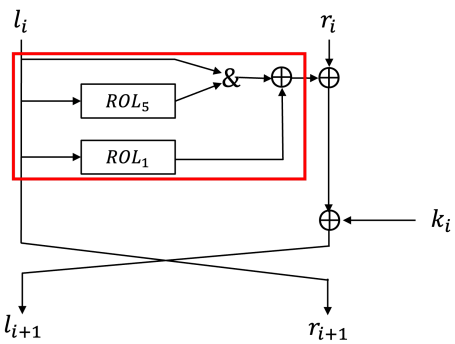

Fig. 7. Omitable part due to Nonce value of SIMECK.

에 대해 Fig. 7. 과 같이 XOR 연산에 대해 순서가 바뀌어도 동일한 연산 결과를 갖는 가능한 특징을 활용한다.

3.3.1 SIMECK-32/64 카운터 운용 모드

일반적으로 카운터 값은 32-bit를 사용하지만, SIMECK-32/64는 평문 길이가 32-bit이기 때문에 블록 하나의 크기인 16-bit를 카운터 값으로 사용하였다. 왼쪽 블록에는 고정 값인 nonce 값이 위치해 있어 Fig. 7. 의 빨간 박스에 해당되는 연산은 카운터 값이 아닌 고정된 nonce 값에 대한 연산이기 때문에 사전 연산을 진행하였다.

단일 평문 구현과 2개의 평문을 병렬 구현은 사전 연산을 통해 동일하게 3번의 XOR 연산, 2번의 로테이션 연산을 구현하기 위해 사용된 4번의 쉬프트 연산, 5번의 AND 연산을 생략하였다. 암호화가 시작되기 전, 서로 다른 2개의 평문이 연속 정렬 되게 사전에 레지스터 내부 정렬을 진행해 주기 때문에 단일 평문과 2개의 평문을 병렬 구현은 사전 연산을 통한 생략된 연산이 동일하다.

3.3.2 SIMECK-64/128 카운터 운용 모드

SIMECK-64/128은 평문 길이가 64-bit로, 두 개의 블록은 각각 32-bit로 나뉘기 때문에 하나의 블록은 카운터 값으로 사용하였다. SIMECK-64/ 128도 SIMECK-32/64와 동일한 라운드 구조를 갖고 있기 때문에 동일한 부분의 연산에 대한 생략이 가능하다.

2개의 평문에 대해 병렬 구현하기 위해 암호화 시작 전 하나의 레지스터에 서로 다른 평문의 상위 16-bit 혹은 하위 16-bit를 묶어주기 위해 〔Fig 3〕과 같이 레지스터 내부 정렬을 하였다. 병렬 구현은 〔Fig 3〕과 같이 상위 비트와 하위 비트가 나누어져 있기 때문에 로테이션 연산을 구현을 하기 위해서 더 많은 연산이 필요하다.

결과적으로, 단일 평문에 대한 구현은 사전 연산을 통해 3번의 XOR 연산, 4번의 쉬프트 연산, 1번의 AND 연산을 생략하였다. 2개의 평문에 대한 병렬 구현은 사전 연산을 통해 6번의 XOR 연산, 8번의 쉬프트 연산, 10번의 AND 연산을 생략하였다.

3.3.3 SIMON-64/128 카운터 운용 모드

SIMON-64/128은 SIMECK-64/128과 동일하게 평문 길이가 64-bit로, 두 개의 블록은 각각 32-bit로 나뉘기 때문에 하나의 블록은 카운터 값으로 사용하였다. SIMON-64/128도 SIMECK-64/128과 유사한 라운드 구조를 갖고 있기 때문에 Fig.8.의 빨간 박스에 해당되는 연산은 카운터 값이 아닌 고정된 nonce 값에 대한 연산이기 때문에 사전 연산을 진행하였다.

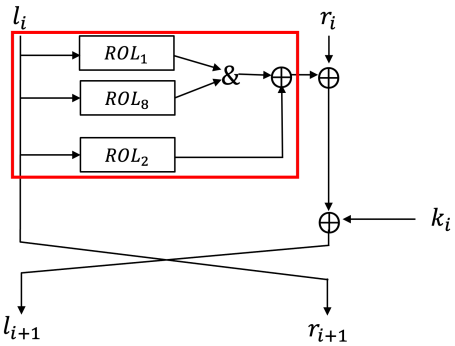

Fig. 8. Omitable part due to Nonce value of SIMON.

결과적으로, 단일 평문에 대한 구현은 사전 연산을 통해 2번의 XOR 연산, 6번의 쉬프트 연산, OR 연산 3번, 1번의 AND 연산을 생략하였다. 2개의 평문에 대한 병렬 구현은 사전 연산을 통해 10번의 XOR 연산, 10번의 쉬프트 연산, 12번의 AND 연산, 2번의 MV 연산을 생략하였다.

IV. 성능 평가

본 장에서는 SiFive 사의 HiFive1 Rev B 플랫폼인 RISC-V를 활용하여 제안 기법에 대한 성능을 비교한다. RISC-V는 4MB의 Quad SPI 플래시 메모리, E31 core가 탑재되어 있기 때문에 320MHz로 연산이 수행된다. 구현과 성능 측정은 SiFive 사에서 제공하는 Freedom Studio 프레임워크를 사용하였다.

32-bit RISC-V 프로세서 상에서의 SIMECK과 SIMON 최적 구현에 관한 기존 연구가 없기 때문에, SIMECK과 SIMON의 Reference-c과 제안하는 기법이 적용된 구현에 대해 Clock Cycles을 측정하여 성능을 비교하였다.

Table 5. Evaluation result on RISC-V.

SIMECK-32/64		
Ref-c	3,298	
1-pt	our work	655
	our work*	646
2-pt	our work	635
	our work*	625
SIMECK-64/128		
Ref-c	2,734	
1-pt	our work	612
	our work*	594
2-pt	our work	1,560
	our work*	1,540
SIMON-64/128		
Ref-c	2,354	
1-pt	our work	728
	our work*	691
2-pt	our work	2,055
	our work*	2,022

성능 비교 결과는 Table 5.와 같다. 1-pt는 단일 평문 최적 구현을 의미하고, 2-pt는 2개의 평문에 대한 최적 구현을 의미한다. our work는 사전 연산이 적용되지 않은 최적 구현이고 our work*는 사전 연산을 통한 CTR 최적 구현을 의미한다.

SIMECK-32/64, SIMECK-64/128, SIMON-64/ 128은 사전 연산을 통한 단일 평문 CTR 최적 구현은 레퍼런스 대비 각각 5.1배, 4.6배, 3.5배 성능 향상을 확인하였다. 그리고 사전 연산을 통한 2개의 평문 병렬 CTR 최적 구현은 레퍼런스 대비 각각 10.6배, 3.6배, 2.3배 성능 향상을 확인하였다. 결과적으로, 단일 평문 최적화와 2개의 평문 병렬 최적화된 구현물에 대해 사전 연산 기법이 적용된 SIMECK-CTR과 SIMON-CTR 최적 구현은 사전 연산이 적용되지 않은 최적 구현 대비 1% 성능 향상을 확인하였다.

V. 결 론

본 논문에서는 32-bit RISC-V 프로세서 상에서 SIMECK과 SIMON 카운터 운용 모드 최적 구현

을 최초로 제안한다. 제안 기법은 SIMECK과 SIMON에 대해 단일 평문 최적 구현과 2개의 평문 병렬 최적 구현 기법과 고정된 nonce 값을 활용하는 CTR 운용모드의 특징을 활용한 사전 연산을 통한 연산 생략 구현을 제안한다. SIMECK-32/64, SIMECK-64/128, SIMON-64/128에 대해 단일 평문과 2개의 평문이 병렬된 구현에 적합한 사전 연산을 하여 각각 적용하여 연산을 생략하였다. 그 결과, 본 논문에서 제안하는 6개의 구현물에 대해 사전 연산이 적용된 최적 구현은 적용되지 않은 최적 구현 대비 모두 1%의 성능 향상을 확인하였다. 향후 과제로 다른 경량 블록 암호의 CTR 운용 모드에 대한 최적 구현을 제안한다.

References

[1] G. Yang, B. Zhu, V. Suder, M.D. Aagaard, G. Gong, "The SIMECK family of lightweight block ciphers," In Cryptographic Hardware and Embedded Systems CHES 2015: 17th International Workship, pp. 307-329, Sep. 2015.

[2] T. Park, H. Seo, B. Bae, and H. Kim, "Efficient implementation of Simeck family block cipher on 8-bit processor," Journal of information and communication convergence engineering, vol. 14, no. 3, pp. 177-183, Aug. 2016.

[3] T. Park, H. Seo, G. Lee, and H. Kim, "Efficient implementation of simeck family block cipher on 16-bit MSP430," 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 983-988, July. 2017.

[4] T. Park, H. Seo, and H. Kim, "Fast implementation of simeck family block ciphers using avx2," 2018 International Conference on Platform Technology and Service (PlatCon), pp. 1-6, Jan. 2018.

[5] T. Park, H. Seo, C. Park, and H. Kim, "Parallel Implementation of Simeck Family Block Cipher by Using ARM NEON." 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), July. 2018.

[6] Kyung-bae Jang, Hyun-jun Kim, Se-jin Lim, and Hwa-jeong Seo, "Parallel Implementation of SPECK, SIMON and SIMECK by Using NVIDIA CUDA PTX," Journal of the Korea Institute of Information Security & Cryptology, 31(3), pp. 423-431, Jun. 2021.

[7] T. Park, H. Seo, M.A.A. Khandaker, Y. Nozami, and H. Kim, "Efficient Parallel Simeck Encryption with GPGPU and OpenCL," 2018 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW), pp. 1-2, May. 2018.

[8] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith and L. Wingers, "The SIMON and SPECK lightweight block ciphers," Proceedings of the 2015 IEEE Design Automation Conference, pp. 1-6, Jun. 2015.

[9] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The Simon and Speck Families of Lightweight Block Ciphers," Cryptology ePrint Archive, Jun. 2013.

[10] T. Park, H. Seo, G. Lee, M.A.A. Khandaker, Y. Nogami, and H. Kim, "Parallel implementations of Simon and speck, revisited," Information Security Appication: 18th International Conference, WISA 2017, pp. 283-294, Aug. 2018.

[11] T. Park, H. Seo, and H. Kim, "Parallel implementations of SIMON and SPECK", 2016 International

Conference on Platform Technology and Service (PlatCon), pp. 1-6, Feb. 2016.

[12] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The Simon and Speck Block Ciphers on AVR 8-Bit Microcontrollers," in International Workshop on Lightweight Cryptography for Security and Privacy, Istanbul, pp. 3-20, Sep. 2014.

[13] Hyeok-dong Kwon, Kyung-bae Jang, Hyun-ji Kim, and Hwa-jeong Seo, "The fast implementation of block cipher SIMON using pre-computation with counter mode of operation," Journal of the Korea Institute of Information and Communication Engineering , 25(4), pp 588-594, Apr. 2021.

[14] SiFive, Inc. "The RISC-V Instruction Set Manual Volume I: User-Level ISA Document Version 2.2," May. 2017. https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf

〈저자소개〉

심 민 주 (Min-Joo Sim) 학생회원
2021년 2월: 한성대학교 IT융합공학부 학사 졸업
2023년 2월: 한성대학교 IT융합공학부 석사 졸업
2023년 3월~현재: 한성대학교 정보컴퓨터공학과 박사과정
〈관심분야〉 암호구현, 정보보안

권 혁 동 (Hyeok-Dong Kwon) 학생회원
2018년 2월: 한성대학교 정보시스템공학과 학사 졸업
2020년 2월: 한성대학교 IT융합공학부 석사 졸업
2020년 3월~현재: 한성대학교 정보컴퓨터공학과 박사과정
〈관심분야〉 정보보안, 암호구현

오 유 진 (Yu-Jin Oh) 학생회원
2023년 2월: 한성대학교 IT융합공학부 학사 졸업
2023년 3월~현재: 한성대학교 융합보안학과 석사과정
〈관심분야〉 인공지능보안, 정보보호

송 민 호 (Min-Ho Song) 학생회원
2023년 2월: 한성대학교 IT융합공학부 학사 졸업
2023년 3월~현재: 한성대학교 융합보안학과 석사과정
〈관심분야〉 정보보안, 정보보호

서 화 정 (Hwa-Jeong Seo) 종신회원
2010년 2월: 부산대학교 컴퓨터공학과 학사 졸업
2012년 2월: 부산대학교 컴퓨터공학과 석사 졸업
2015년 4월~5월 : 싱가포르 난양공대 인턴쉽
2016년 2월: 부산대학교 컴퓨터공학과 박사 졸업
2017년 3월: 싱가포르 과학기술청 연구원
2017년 4월~2023년 2월: 한성대학교 IT융합공학부 조교수
2023년 3월~현재: 한성대학교 융합보안학과 부교수
〈관심분야〉 정보보호, 암호화 구현, IoT