32-bit RISC-V 프로세서 상에서의 경량 블록 암호 SIMECK-CTR 최적 구현

심민주*, 권혁동*, 오유진**, 송민호**,서화정*†

*한성대학교 IT융합공학부(대학원생)

**한성대학교 IT융합공학부(학부생)

*† 한성대학교 IT융합공학부 (교수)

Optimized Implementation of Lightweight Block Cipher SIMECK-CTR on 32-bit RISC-V Processor

Min-Joo Sim*, Hyeok-Dong Kwon*, Yu-Jin Oh**, Min-Ho Song**, Hwa-Jeong Seo*†

*Dept. of IT Convergence Engineering, Hansung University (Graduate student)

**Dept. of IT Convergence Engineering, Hansung University (Undergraduate student)

*† Dept. of IT Convergence Engineering, Hansung University (Professor)

요 약

사물인터넷의 성능이 빠른 속도로 향상됨에 따라 저사양 프로세서에 안전하고 효율적인 경량 암호에 대한 개발은 활발히 진행되고 있다. 본 논문에서는 32-bit RISC-V 프로세서 상에서 경량 블록 암호인 SIMECK의 CTR 운용 모드에 대한 최적 구현을 제안한다. 고정된 nonce 값을 사용하는 CTR 운용 모드의 특징을 활용하여 일부 값을 사전 연산하는 라운드 함수 최적화와 SIMECK의 라운드 함수 특징을 활용한 블록 이동 연산을 생략하는 최적화를 제안한다. 제안 기법을 통해 SIMECK-32/64의 단일 평문과 병렬된 2개의 평문에 대한 구현은 12번의 연산 생략이 가능하였고, SIMECK-64/128의 단일 평문과 병렬된 2개의 평문에 대한 구현은 각각 8번과 24개의 연산 생략이 가능하였다. 결과적으로, 기존 32-bit RISC-V 상에서의 최적 구현 연구 대비 제안 기법을 적용하였을 때 1%의 성능 향상을 확인하였다.

I. 서론

사물인터넷이 빠르게 발전됨에 따라, 사물인터넷에 사용되는 저사양 프로세서들의 보안은 필수적으로 요구된다. 국내외에서 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 마이크로컨트롤러, 16-bit MSP430 마이크로 컨트롤러, 32-bit ARM 마이크로 컨트롤러 등 다양한 환경에서 활발히 진행되었다[2~8]. SIMECK-CTR에 대한 최적 구현 연구는 현재까지 진행되지 않았다. 하지만, SIMECK과 유사한 구조를 지닌 SIMON-CTR에 대한 8-bit AVR 상에서의 최적 구현 연구가 제안되었다[9]. 따라서, 본 논문에서는 기존 연구의 최적 구현 기법을 참조하여 32-bit RISC-V 프로세서 상에서의 SIMECK-CTR 최적 구현을 진행한다.

본 논문의 구성은 다음과 같다. 2장에서 SIMECK, CTR 운용 모드, RISC-V에 관해 알아본다. 3장에서 제안 기법을 설명하고, 4장에서는

제안 기법에 대한 성능 평가를 진행한다. 마지막으로 5장에서 본 논문의 결론을 내린다.

II. 관련 연구

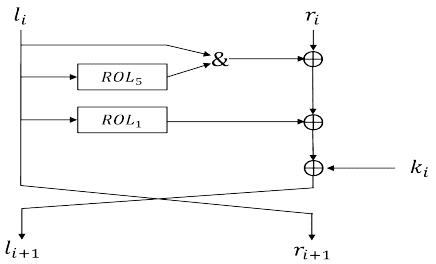
2.1 SIMECK 경량 블록 암호

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.

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

[Table 1] List of SIMECK ciphers and their parameters.

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



[Fig 1] Round function of SIMECK.

2.2 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 사용하여 암호화하여 키 스트림을 만든다. 그리고 생성된 키 스트림과 평문을 XOR 연산을 한 결과가 암호문의 이 되는 특징을 갖고 있다.

2.3 RISC-V

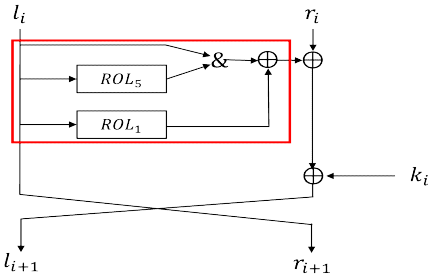
RISC-V는 32-bit 구조인 RV32I와 64-bit 구조인 RV64I가 있다[10]. RV32I는 32-bit 32개의 레지스터를 제공하며, 레지스터 용도는 [Table 2]와 같다.

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	

[Table 2] Purpose of registers in RISC-V processors.

III. 제안 기법

본 논문에서는 단일 평문과 2개의 평문 병렬 구현에 대한 SIMECK-32/64, SIMECK-64/128에 대한 CTR 모드 최적 구현을 한다.



[Fig 2] Omitable part due to Nonce value.

3.1 SIMECK-32/64

CTR 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 이전까지인 1라운드의 일부만 가능하다. 따라서, 기존 [Fig 1]과 같은 구조를 가진 SIMECK에 대해 [Fig 2]와 같이 XOR 연산

에 대해 순서가 바뀌어도 동일한 연산 결과를 갖는 가능한 특징을 활용한다.

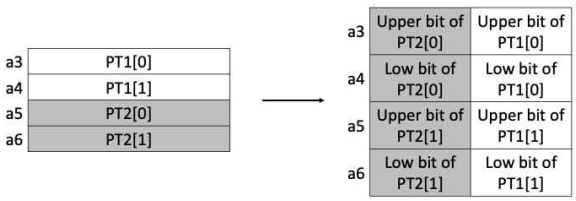
일반적으로 카운터 값은 32-bit를 사용하지만, SIMECK-32/64는 평문 길이가 32-bit이기 때문에 블록 하나의 크기인 16-bit를 카운터 값으로 사용하였다.

왼쪽 블록에는 고정 값인 nonce 값이 위치해 있어 [Fig 2]의 네모 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 진행해주기 때문에 단일 평문과 2개의 평문을 병렬 구현은 사전 연산을 통한 생략된 연산이 동일하다.

3.2 SIMECK-64/128

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.



[Fig 3] Register internal alignment in SIMECK-64/128.

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 나뉘어져 있기 때문에 로테이션 연산을 구현을 하기 위해서 더 많은 연산이 필요하다.

그 결과, 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 블록 이동 생략

SIMECK은 두 개의 블록에 대해 연산이 진행된다. SIMECK의 라운드 수는 모두 짝수이기 때문에 모든 연산이 끝난 블록의 위치는 처음과 동일하다. 이러한 라운드 함수의 특징을 활용해 라운드 함수 마지막에 진행되는 블록의 이동을 생략하였다. 결과적으로, 한 라운드에서 두 개의 블록 이동을 위한 2번의 연산을 생략하였다.

IV. 성능 평가

본 장에서는 SiFive 사의 HiFive1 Rev B 플랫폼인 RISC-V를 활용하여 제안 기법에 대한 성능을 비교한다.

SIMECK-32/64	[8]	our work
1-pt	655	646
2-pt	635	625
SIMECK-64/128	[8]	this work
1-pt	612	594
2-pt	1,560	1,540

[Table 3] Evaluation result on RISC-V.

32-bit RISC-V 프로세서 상에서의 SIMECK-CTR 최적 구현에 관한 기존 연구가 없기 때문에, 32-bit RISC-V 상에서의 SIMECK 최적 구현 Sim et al.[8]과 성능 비교를 하였다. 성능 비교 결과는 [Table 3]과 같다. 본 논문에서 제안한 기법이 적용된 최적 구현은 기존 연구 대비 1% 성능 향상을 확인하였다.

V. 결론

본 논문에서는 32-bit RISC-V 프로세서 상에서 SIMECK-CTR 최적 구현을 제안한다. 제안 기법은 고정된 nonce 값을 활용하는 CTR 운용

모드의 특징을 활용하여 사전 연산을 통한 연산 생략 구현을 제안한다. SIMECK-32/64와 SIMECK-64/128 각각 단일 평문과 2개의 평문이 병렬된 구현에 대해 적합한 사전 연산을 하여 연산을 생략하였다. 그리고 SIMECK의 라운드 함수 특징을 활용하여 블록의 이동을 생략하였다. 그 결과, 4개의 구현물에 대해 기존 SIMECK 최적 구현 연구 대비 1%의 성능 향상을 확인하였다. 향후 과제로 다른 경량 블록 암호의 CTR 운용 모드에 대한 최적 구현을 제안한다.

VI. Acknowledgment

이 성과는 2022년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2020R1F1A1048478, 100%).

[참고문헌]

[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, pp. 307-329. 2015.

[2] Park, Taehwan, et al. "Efficient implementation of Simeck family block cipher on 8-bit processor." Journal of information and communication convergence engineering 14.3 (2016): 177-183.

[3] Park, Taehwan, et al. "Efficient implementation of simeck family block cipher on 16-bit MSP430." 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN). IEEE, 2017.

[4] Park, Taehwan, Hwajeong Seo, and Howon Kim. "Fast implementation of simeck family block ciphers using avx2." 2018 International Conference on Platform Technology and Service (PlatCon). IEEE, 2018.

[5] Park, Taehwan, et al. "Parallel Implementation of Simeck Family Block Cipher by Using ARM NEON." 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN). IEEE, 2018.

[6] Jang, Kyung-bae, et al. "Parallel Implementation of SPECK, SIMON and SIMECK by Using NVIDIA CUDA PTX." Journal of the Korea Institute of Information Security & Cryptology 31.3 (2021): 423-431.

[7] Park, Taehwan, et al. "Efficient Parallel Simeck Encryption with GPGPU and OpenCL." 2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW). IEEE, 2018.

[8] M. Sim, S. Eum, H. Kwon, H. Seo, "Optimized Parallel Implementation of Lightweight Block Cipher SIMECK on 32-bit RISC-V Processor", Conference on Information Security and Cryptography Summer 2022.

[9] Kwon, Hyeok-Dong, et al. "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 (2021): 588-594.

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

[11] SiFive, Inc. "The RISC-V Instruction Set Manual Volume I: User-Level ISA Document Version 2.2". May 7, 2017. https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf