32-bit RISC-V 프로세서 상에서의 경량 블록 암호 SPECK 카운터 운용 모드 최적 구현

심민주[1], 이민우[2], 송민호[2], 서화정[3]
[1]한성대학교 정보컴퓨터공학과 박사과정
[2]한성대학교 융합보안학과 석사과정
[3]한성대학교 융합보안학과 교수
minjoos9797@gmail.com, minunejip@gmail.com, smino0906@gmail.com,
hwajeong84@gmail.com

Optimized Implementation of Lightweight Block cipher SPECK Counter Operation Mode on 32-bit RISC-V Processors

Min-Joo Sim[1], Min-Woo Lee[2], Min-Ho Song[2], Hwa-Jeong Seo[3]
[1]Dept. of Information Computer Engineering, Hansung University
[2,3]Dept. of Convergence Security, Hansung University

요 약

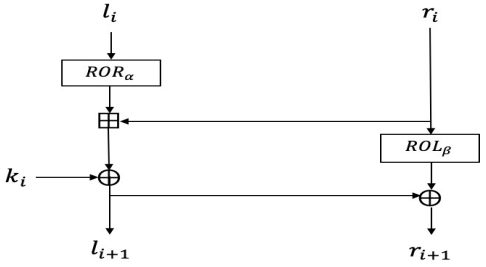
본 논문에서는 32-bit RISC-V 프로세서 상에서의 경량 블록 암호인 SPECK의 CTR 운용 모드에 대한 최적 구현을 제안한다. RISC-V 상에서의 SPECK 단일 평문과 2개의 평문에 대한 최적화와 고정된 논스 값을 사용하는 CTR 운용모드의 특징을 활용하여 일부 값에 대해 사전 연산을 하는 라운드 함수 최적화를 제안한다. 결과적으로, 레퍼런스 대비 제안된 기법은 단일 평문과 2개의 평문에 대해 각각 5.76배, 2.24배 성능 향상을 확인하였으며, 사전 연산 기법을 적용하지 않은 최적 구현 대비 사전 연산 기법을 적용하였을 때, 1% 성능 향상을 확인하였다.

1. 서론

SPECK은 2013년에 SIMON과 함께 발표된 경량 블록 암호이다[1]. SPECK에 대한 최적화 연구는 8-bit AVR 마이크로 컨트롤러, 32-bit ARM 프로세서 등 다양한 환경에서 활발히 진행되었다[2~3]. 32-bit RISC-V 상에서 SPECK에 대한 최적 구현 연구는 아직 수행되지 않았기 때문에 본 논문에서는 RISC-V 상에서 SPECK-CTR 최적화를 진행한다.

2. 관련 연구

2.1 SPECK 경량 블록 암호



[그림 1] Round function of SPECK.

SPECK은 2013년 NSA에서 개발한 Feistel 구조의 경량 블록 암호로, 소프트웨어 구현에 최적화되어 있다. SPECK의 라운드 함수는 [그림 1]과 같다. SPECK의 각각의 암호화에 대한 파라미터는 [표 1]과 같다.

[표 1] List of SPECK ciphers and their parameters.

Block Size	Key Size	Rounds	Word Size	α	β
32	64	22	16	7	2
48	72	22	24	8	3
	96	23			
64	96	26	32		
	128	27			
96	96	28	48		
	144	29			
128	128	32	64		
	192	33			
	256	34			

2.2 RISC-V Processors

RISC-V는 32-bit와 64-bit 구조를 제공한다. 32-bit 구조인 RV32I는 2개의 32-bit 레지스터를 제공한다[4]. $x0$ 레지스터는 항상 0의 값을 갖는 레지스터이고, 이외 $x1$~$x4$의 레지스터는 특수 용도에 사

용되는 레지스터로 지정되어 있다.

2.3 카운터 운용 모드

블록 암호 운용 모드 중 하나인 카운터(Counter, CTR) 운용 모드는 임의의 고정된 상수인 논스 값과 블록을 암호화할 때마다 1씩 증가하는 카운터를 결합한 값을 입력으로 사용하여 키 스트림을 만든다. 그리고 생성된 키 스트림과 평문을 XOR 연산한 결과가 암호문이 되는 특징을 갖고 있다.

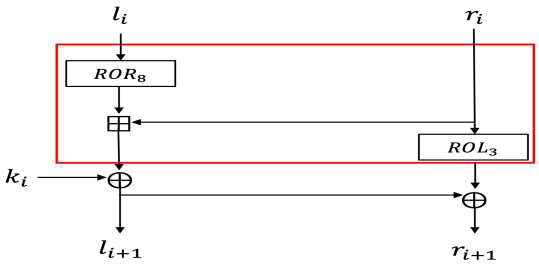
3. 제안 기법

본 논문에서는 단일 평문과 2개의 평문 병렬 구현에 대한 SPECK-64/128에 대한 CTR 운용 모드 최적화를 제안한다.

CTR 운용 모드는 고정된 논스 블록이 카운터 블록에 영향을 받기 전까지에 해당되는 연산에 대해 사전 연산이 가능하다. 따라서, 2개의 블록으로 연산이 수행되는 SPECK에 대한 사전 연산은 카운터 블록에 영향을 받기 이전인 1라운드의 일부에 대해서만 가능하다.

3.1 단일 평문 최적화

SPECK-64/128의 평문 길이는 64-bit로, 2개의 블록은 각각 32-bit로 나뉘기 때문에 하나의 블록은 카운터 값으로 사용하였다. SPECK-64/128의 하나의 블록 사이즈는 32-bit이기 때문에 32-bit 레지스터를 갖는 RISC-V의 명령어를 효율적으로 사용하여 구현하였다.

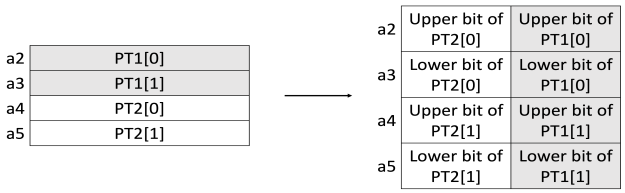


[그림 2] Part of omitted by precomputation.

[그림 2]의 빨간색 박스 부분에 해당되는 연산에 대해 사전 연산을 수행하였다. 결과적으로, 단일 평문에 대한 구현은 사전 연산을 통해 2번의 쉬프트 연산과 1번의 ADD 연산 생략하였다.

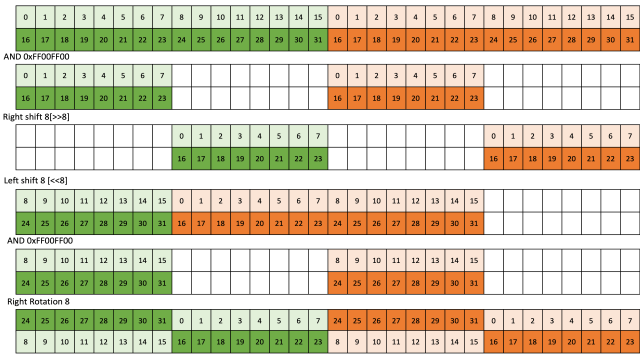
3.2 2개의 평문 최적화

2개의 평문에 대해 병렬 구현하기 위해서는 암호화 연산 전 하나의 레지스터에 [그림 3]과 같이 서로 다른 2개의 평문의 상위 비트와 하위 비트가 위치하게 내부 정렬을 수행하였다.



[그림 3] Register internal alignment in SPECK-64/128.

RISC-V에서는 로테이션 명령어를 따로 제공하지 않아 쉬프트 연산을 활용하여 구현하여야 한다. 하지만, 서로 다른 두 개의 16-bit 값이 연속으로 정렬된 하나의 레지스터에 대한 로테이션 연산을 수행하기 위해서는 값이 섞이지 않아야 한다.

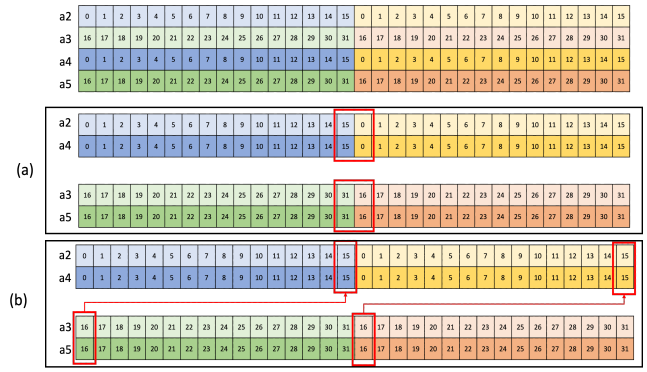


[그림 4] Process of Right Rotation 8 in SPECK-64/128.

따라서, 쉬프트 연산, AND 연산, XOR 연산을 이용하여 [그림 4]와 같이 구현하였다. 쉬프트 연산을 할 때, 값이 섞이게 되기 때문에 AND 연산을 통해 섞이는 위치의 값을 0으로 변경 후, 섞이는 위치에 대한 값들을 temp로 사용하는 레지스터에 따로 저장한다. 그리고 쉬프트 연산 후, 따로 저장해 둔 값과 XOR 연산하여 로테이션 연산을 구현하였다.

[그림 5]의 (a)의 빨간색 박스와 같이 덧셈 연산을 수행하기 위해서는 로테이션 연산과 동일하게 값이 섞이지 않아야 하고, 덧셈 연산으로 인해 (b)의 빨간색 박스 부분에 캐리가 발생할 경우, 해당 값들을 적절하게 더해주어야 한다. 따라서, 캐리가 발생할 수 있는 (a)의 a3과 a5에 해당되는 두 개의 값을 따로 저장한다. 그리고 해당 위치의 값을 0으로 바꿔준 후, 덧셈 연산을 진행하였고, 따로 분리한 두 연산 결과가 XOR 연산하여 합쳐주는 기법을 적요한 덧셈 연산을 구현하였다.

결과적으로, 2개의 평문에 대한 병렬 구현은 사전 연산을 통해 6번의 XOR 연산, 6번의 쉬프트 연산, 15번의 AND 연산, 5번의 ADD 연산, 10번의 MV 연산, 1번의 STLU 연산 생략하였다.



[그림 5] Process of Addition Operation in SPECK-64/128.

4. 성능 평가

본 장에서는 SiFive사의 HiFive1 Rev B 플랫폼인 RISC-V를 활용하여 제안 기법에 대한 성능을 비교한다. 32-bit RISC-V 프로세서 상에서 SPECK에 대한 선행 연구가 존재하지 않기 때문에 SPECK-64/128에 대한 단일 평문과 2개의 평문 병렬 구현 최적 구현물과 사전 기법까지 적용된 구현물(*)에 대해 성능을 비교하였다.

제안 기법에 대한 Clock Cycles를 측정하여 성능을 비교한다. 성능 비교 결과는 [표 3]과 같다.

[표 3] Evaluation result on RISC-V.

SPECK-64/128	Clock Cycles	
Ref-c	1,873	
1-pt	our work	327
	our work*	**325**
2-pt	our work	1,768
	our work*	**1,670**

결과적으로, 레퍼런스 대비 제안된 기법은 단일 평문과 2개의 평문에 대해 각각 5.76배, 2.24배 성능 향상을 확인하였으며, 사전 연산 기법이 적용된 구현물은 적용되지 않은 구현물 대비 1% 성능 향상을 확인하였다.

5. 결론

본 논문은 32-bit RISC-V 프로세서 상에서의 SPECK 카운터 운용 모드 최적 구현을 최초로 제안한다. RISC-V 상에서 SPECK에 대한 선행 연구가 없기 때문에 단일 평문 최적화와 2개의 평문 병렬 최적화를 수행 후, 고정된 nonce 값을 활용되는 카운터 운용 모드의 특징을 활용하여 사전 연산을 통한 연산 생략 기법을 적용하였다. 결과적으로, 레퍼런스 대비 제안된 기법은 단일 평문과 2개의 평문에 대해 각각 5.76배, 2.24배 성능 향상을 확인하였으며, 사전 연산 기법이 적용된 구현물은 적용되지 않은 구현물 대비 1% 성능 향상을 확인하였다.

6. Acknowledgements

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, 50%) 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, 50%).

참고문헌

[1] 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.

[2] 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 Lightweight Cryptography for Security and Privacy: Third International Workshop, LightSec 2014, Istanbul, Turkey, September 1-2, 2014, Revised Selected Papers 3. Springer International Publishing, pp. 3-20, 2015.

[3] 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 Conference, WISA 2017, pp. 283-294, Aug. 2018.

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