RISC-V 프로세서 상에서의 경량 블록 암호 SIMON과 SPECK 최적 구현

엄시우*, 권혁동*, 김현지*, 서화정**

*한성대학교 IT융합공학부 (대학원생)

**한성대학교 IT융합공학부 (교수)

Implementation of lightweight block cipher SIMON and SPECK on RISC-V processors

Si-Woo Eum*, Hyeok-Dong Kwon*, Hyun-Ji Kim*, Hwa-Jeong Seo**

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

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

요 약

경량 블록 암호 SIMON과 SPECK은 간단한 알고리즘으로 어느 플랫폼에서나 우수한 성능을 발휘할 수 있도록 다양한 블록 크기와 키 크기를 지원한다. 본 논문에서는 RISC-V 프로세서를 대상으로 Rotation과 Addition 연산의 최적 구현과 레지스터의 복사를 최소화한 최적 구현을 제안한다. 구현은 32-bit E31 RISC-V 코어가 포함된 HiFive1 Rev B 개발 보드를 대상으로 한다. 그 결과 Reference C 코드 대비 3~5배의 성능 향상을 확인할 수 있었으나, Fixslice 기법이 적용된 AES와 성능 비교하였을 때, SIMON의 경우 0.6배의 성능을 확인하였다.

I. 서론

사물인터넷이 발전하면서 계산 작업은 점점 더 작고 작은 장치로 내려갔다. 기존의 암호 알고리즘이 이러한 새로운 현실의 요구에 적합하지 않다라는 것이 인식되어 왔다. 경량 암호는 이러한 제한된 플랫폼에서 작동하도록 설계된 알고리즘이다.

기존의 경량 블록 암호는 특정 종류의 플랫폼을 대상으로 설계되는 것이 일반적이고, 고정된 블록 크기와 두 종류의 키 크기를 갖는 경향이 있다. 그로 인해 특정 플랫폼에서 우수한 성능을 달성할 수 있으나, 다른 플랫폼에서는 성능이 떨어지는 유연성이 부족한 점이 있다. 다양한 장치들이 통신을 주고받고 있는 현시대에서 유연성이 부족한 것은 단점이 될 수 있다. 따라서 특정 플랫폼에 맞춰 설계하는 것보다 간단한 알고리즘을 만들어 어느 플랫폼에서나 우수한 성능을 발휘하는 것이 더 낫다. 이를 위해 SIMON과 SPECK 알고리즘은 다양한 블록크기와 세 개의 키 크기를 지원한다[1].

RISC-V는 2010년부터 UC 버클리에서 개발 중인 새로운 컴퓨터 CPU 구조이다. 단지 학술용이나 연구용이 아닌 산업계의 상용화를 목표로 하고 있다. 가장 큰 영향력을 가지고 있는 ARM과 달리 라이선스를 지불하지 않고 무료로 사용할 수 있어 여러 단체가 RISC-V 컨소시엄에 포함되어 활동 중에 있다.

본 논문에서는 SIMON과 SPECK 알고리즘을 RISC-V 프로세서 상에서 최적 구현한 결과를 제시한다. 2장에서는 RISC-V 프로세서의 구조와 SIMON과 SPECK 알고리즘에 대해 확인한다. 3장에서는 제안하는 최적 구현 기법을 제시한다. 4장에서는 성능을 비교 분석하고, 마지막으로 5장에서는 본 논문의 결론을 내린다.

II. 관련 연구

2.1 RISC-V 프로세서의 구조 및 명령어

RISC-V의 32-bit 구조인 RV32I는 32-bit 레지스터 32개(x0~x31)를 제공한다. RISC-V 프로세서 상에서의 레지스터 용도는 [표 1]과 같다.

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

[표 1] Registers in RISC-V[2]

zero(x0) 레지스터는 다른 값을 넣을 수 없고 항상 0의 값을 가지고 있다. a0~a7 레지스터는 함수 인자와 반환 값을 가질 수 있다. sp와 s0~s11 레지스터는 callee saved 레지스터로 레지스터 사용 전 기존의 값을 보존시켜줘야 한다.

본 논문에서 활용한 RISC-V 프로세서상에서의 명령어 셋은 [표 2]와 같다.

Instruction	Description
ADD	Add
ADDI	Add immediate
OR	Inclusive or
XOR	Exclusive or
SLTU	Set less than unsigned
SLLI	Shift left logical immediate
SRLI	Shift right logical immediate

[표 2] Instruction for RISC-V[2]

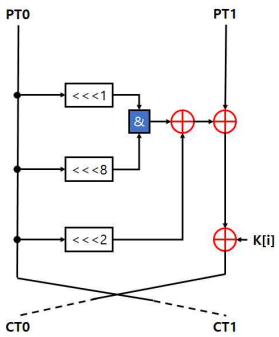
2.2 SIMON-SPECK알고리즘

경량 블록 암호 SIMON은 경량 블록 암호 SPECK과 함께 발표된 암호로, 2013년 미국 국가안보국(National Security Agency, NSA)에서 개발하였다. 2018년에는 RFID 에어 인터페이스 표준으로 선정되어 ISO/29167-21 표준 번호를 보유하고 있다[3]. SIMON과 SPECK은 다양한 환경을 지원하기 위해 [표 3]과 같이 많은 종류의 암호 규격을 제공한다.

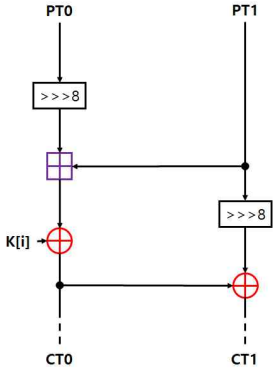
Block size(bit)/2n	Key Size (bit)/mn	Number of rounds	
		SIMON	SPECK
2 × 16 = 32	64	32	22
2 × 24 = 48	72	36	22
	96	36	23
2 × 32 = 64	96	42	26
	128	44	27
2 × 48 = 96	96	52	28
	144	54	29
2 × 64 = 128	128	68	32
	192	69	33
	256	72	34

[표 3] Parameters of SIMON and SPECK

SIMON, SPECK은 ARX(Addition, Rotation, XOR) 구조로 이루어져 있다. 다만 SIMON의 경우 Addition 연산 대신 AND 연산을 사용하여 설계되어 있다. SIMON과 SPECK은 Feistel 구조로, 암·복호화를 할 때 평문을 2개의 블록으로 나누어 반복적인 라운드 함수를 적용한다. 전체적인 구조는 [그림 1,2]와 같다.



[그림 1] SIMON Algorithm



[그림 2] SPECK Algorithm

III. 최적 구현 기법

본 논문에서는 Rotation, Addition 연산의 최적화와 레지스터 간의 복사를 최소화하여 구현하는 것을 목표로 진행하였다.

SIMON의 경우 1 round를 지나게 되면 평문 간의 자리를 바꾸는 연산이 필요하다. 본 논문에서는 레지스터간의 값 복사를 최소화하기 위해 Round 함수를 통과할 때 한번에 2 round를 동시에 진행한다.

3.1 Rotation 구현

SIMON과 SPECK의 라운드 함수에는 각각 3, 2 번의 로테이션 연산을 사용한다. 하지만 RISC-V에서는 로테이션을 할 수 있는 명령어가 없기 때문에 SRLI, SLLI, OR 연산을 활용하여 로테이션을 구현해주어야 한다. 로테이션 구현은 [표 4]와 같다.

32-Bit Rotation	
Rotaion left n	Rotation right n
SLLI t0, , n SRLI t1, pt, 32-n OR pt, t0, t1	SRLI t0, pt, n SLLI t1, pt, 32-n OR pt, t0, t1
64-Bit Rotation	
Rotation left n	Rotation right n
SLLI d0, pt0, n SRLI d1, pt1, 32-n OR d0, d0, d1 SRLI t0, pt0, 32-n SLLI t1, pt1, n OR d1, t0, t1	SRLI d0, pt0, n SLLI d1, pt1, 32-n OR d0, d0, d1 SLLI t0, pt0, 32-n SRLI t1, pt1, n OR d1, t0, t1

[표 4] 32-bit and 64-bit Rotation on RISC-V
(d : destination, pt : plain text, t : temp)

3.2 Addition 구현

SPECK 알고리즘에서 Addition 연산을 진행하게 된다. 32-bit의 경우 ADD 명령어를 통해 간단하게 구현할 수 있지만, 64-bit의 경우 32-bit 레지스터를 사용하는 RISC-V에서 하위 32-bit와 상위 32-bit의 ADD를 따로 진행해야 한다. 하지만 RISC-V에는 하위 32-bit에서 Carry가 발생하였을 때, 상위 32-bit로 전달해 주는 명령어가 없기 때문에 SLTU 명령어를 활용하여 64-bit Addition을 구현한다. SLTU 명령어는 op1, op2의 값을 비교하여 op2의 값이 op1의 값보다 작다면 목적지 레지스터에 1을 저장하는 명령을 수행한다.

64-Bit Addition
ADD d0, pt0_l, pt1_l //low 32-bit ADD SLTU t0, d0, pt0_l //Check Carry ADD d1, pt0_h, pt1_h //high 32-bit ADD ADD d1, d1, t0 //high 32-bit Carry ADD

[표 5] 64-bit Addition on RISC-V
(d : destination, t : temp, pt : plain text)

[표 5]는 64-bit Addition을 구현한 코드이다. d0에는 PT0(64-bit), PT1(64-bit)의 하위 32-bit ADD 결과 값이 저장된다. 이때 Carry가 발생하지 않았다면 d0의 값이 ADD에 사용된 pt0_l, pt1_l 둘 중 어느 값과 비교하여도 더 크고, Carry가 발생했다면 반대로 더 작기 때문에 레지스터간의 크기를 비교 할 수 있는 SLTU 명령어를 활용하여 Carry의 유무를 파악할 수 있다.

IV. 성능 평가

RISC-V 구현은 확장을 사용하지 않고 RV32I 기반 ISA에 의존한다. 성능 측정은 32-bit E31 RISC-V 코어가 포함된 HiFive1 Rev B 개발 보드를 사용하였다.

Fixslice 기법을 적용한 AES[4]와 성능 비교를 진행한다. SPN 구조를 사용하는 AES와 ARX 구조를 사용하는 SIMON, SPECK을 비교하는 건 것은 형평성에 어긋날 수 있다. 하지만 암호 알고리즘 중에 가장 연구가 많이 되고, 최적화가 잘 되어있는 암호 중 하나인 AES와 성능 비교를 하기에는 충분하다.

[표 6]에서 볼 수 있듯이, Reference C 코드보다 모든 부분에서 약 3~5배 정도의 성능 향상을 확인 할 수 있었다. SIMON과 SPECK은 AES에 비해 알고리즘이 간단하기 때문에 AES보다 더 좋은 성능이 나올 것으로 예측하였다.

	Reference C	RISC-V
AES-128[4]	-	1,468
SIMON-64/96	2,247	681
SIMON-64/128	2,351	712
SIMON-128/128	7,164	2,073
SIMON-128/192	8,770	2,100
SIMON-128/256	7,580	2,193
SPECK-64/96	1,802	368
SPECK-64/128	1,869	381
SPECK-128/128	3,686	788
SPECK-128/192	3,798	812
SPECK-128/256	3,911	835

[표 6] Performance comparison result (Unit: cycles)

SPECK-128의 경우 키 길이와 상관없이 AES-128보다 좋은 성능을 확인 할 수 있다. 하지만 SIMON-128은 AES-128에 비해 성능이 0.7배 정도 차이 나는 것을 확인할 수 있다.

SIMON의 알고리즘은 간단하지만 ROUND 반복 횟수가 많고, Fixslice 기법이 적용된 AES-128은 Rotaion 연산을 사용하지 않는 반면 SIMON-128의 경우 Rotation을 408번 정도 연산하기 때문에 이로 인한 성능 차이가 발생했을 것으로 생각한다. 실제로 Bitslice 기법을 활용한 AES에서 Rotation 연산을 144번 사용할 때, RISC-V에 Rotation 명령어가 있다는 가정하에 성능 평가를 하게 되면 기존보다 7%의 성능 향상을 확인할 수 있었다[5].

V. 결론

본 논문에서는 RISC-V 상에서 경량 블록 암호 SIMON과 SPECK의 최적 구현을 제안한다. Rotation, Addition 연산 최적화에 초점을 맞춰 진행하였으며, 결과 Reference C 코드 대비 3~5배 정도의 성능 향상을 확인할 수 있었다. 하지만 Fixslice 기법이 적용된 AES-128의 비해 SIMON의 성능이 낮게 나온 것도 확인할 수 있었다. 향후 연구 과제로 다양한 경량 블록 암호에 대해 RISC-V상에서의 최적 구현을 제안하고자 한다.

VI. Acknowledgment

이 성과는 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2020R1F1A1048478)

[참고문헌]

[1] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J.Smith, and L. Wingers, “The SIMON and SPECK lightweight block ciphers,” *in 52nd ACM/EDAC/IEEEDesign Automation Conference*, San Francisco, pp. 1-6, 2015.

[2] SiFive, Inc.: SiFive E31 Core Complex Manual, 20G1, June 2020. https://sifive.cdn.prismic.io/sifive/52bb54e3-aab2-4aae-9f74-788332c3cd8d_sifive_E31_rtl_full_20G1.03.00_manual.pdf

[3] H. D. Kwon, K. B Jang, H. J. Kim, H. J. 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*, Vol. 25, No. 4: 588~594, Apr. 2021.

[4] Alexandre Adomnicai, Thomas Peyrin, “Fixslicing AES-like Ciphers New bitsliced AES speed records on ARM-Cortex M and RISC-V”, *IACR Transactions on Cryptographic Hardware and Embedded Systems*, ISSN 2569-2925, Vol. 2021, No. 1, pp. 402 - 425.

[5] Ko Stoffelen, “Efficient Cryptography on the RISC-V Architecture”, *International Conference on Cryptology and Information Security in Latin America, LATINCRYPT 2019*: Progress in Cryptology - LATINCRYPT 2019 pp 323-340