저사양 프로세서 상에서의 경량 블록암호 SIMECK 최적 구현 동향

심민주[1], 이민우[1], 김동현[1], 윤세영[1], 서화정[1]
[1]한성대학교 it융합공학부
minjoos9797@gmail.com, minunejip@gmail.com, donghyun6469@gmail.com,
sebbang99@gmail.com, hwajeong84@gmail.com

Trend of Optimal Implementation of Lightweight Block Cipher SIMECK on Low-end Processors

Min-Joo Sim[1], Min-Woo Lee[1], Dong-Hyun Kim[1], Se-Young Yoon[1],
Hwa-Jeong Seo[1]
[1]Dept. of IT Convergence Engineering, Hansung University

요 약

사물인터넷에 성능이 향상됨에 따라 사물인터넷에 사용되는 저사양 프로세서들의 보안도 주목받고 있다. 이에 따라, 저사양 프로세서 상에서 안전하고 효율적으로 동작하는 경량 암호에 대한 개발과 최적 연구가 활발히 진행되고 있다. 경량 블록 암호 중 하나인 SIMECK은 경량 블록 암호인 SPECK과 SIMON의 이점만을 결합한 암호 알고리즘이다. 본 논문에서는 저사양 프로세서 상에서의 경량 블록암호 SIMECK 최적 구현 동향에 대해 살펴본다.

1. 서론

NIST 경량 암호 공모전 등 국내외에서 공모전 등을 통해 새로운 경량 암호에 대한 개발이 활발하게 진행되고 있다. 다양한 경량 암호에 대한 최적 구현 연구도 진행되고 있다. SIMECK은 경량 블록 암호로, 다양한 환경에서 최적 구현 연구가 진행되었다[1]. 따라서, 본 논문에서는 저사양 프로세서 상에서의 경량 블록암호 SIMECK 최적 구현 연구 동향을 살펴본다. 2장에서는 경량 블록 암호 SIMECK에 대해 알아본다. 3장에서는 저사양 프로세서 상에서의 SIMECK 최적 구현 연구 동향에 대해서 살펴본다. 마지막으로 4장에서 본 논문의 결론을 내린다.

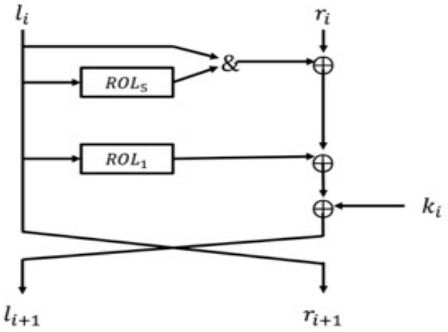
2. 경량 블록암호 SIMECK

CHES'15에서 발표된 SIMECK은 Feistel 구조인 경량 블록 암호이다[1]. SIMECK은 NSA(National Security Agency)에서 개발한 SPECK과 SIMON[2]의 장점만을 결합하여 개발되었다. SIMECK은 SPECK과 유사한 키 스케줄링을 사용하며, SIMON의 라운드 함수를 수정한 구조이다. SIMECK의 암호화에 대한 파라미터는 [표 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

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

SIEMCK은 XOR, Rotation, AND 연산으로 이뤄져있다. SIMECK의 라운드 함수는 [그림 1]과 같다.



[그림 1] Round function of SIMECK.

3. 저사양 프로세서 상에서의 경량 블록암호 SIMECK 최적 연구 동향

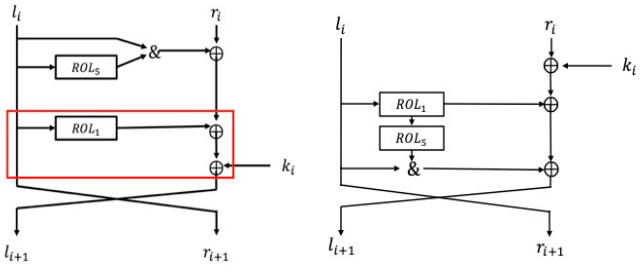
본 장에서는 저사양 프로세서인 8-bit AVR, 16-bit MSP430, 32-bit ARM과 RISC-V 상에서의 경량 블록암호 SIMECK 최적 구현 연구 동향에 대해서 살펴본다.

3.1 8-bit AVR 상에서의 최적 구현

[3]은 8-bit ATmega128 기반인 STK600 보드에서 최적 구현을 진행하였다. 효율적으로 로테이션 연산을 구현하기 위해 데이터 블록을 나누어 8비트 레지스터에 위치시키고, 별도의 레지스터에 쉬프트 연산을 해야 할 비트 값을 저장한다. 효율적으로 로테이션 연산을 하기 위해 로테이션 명령어와 쉬프트 명령어를 활용한다. 이때, 오른쪽 로테이션 연산을 한 후에는 XOR 연산을 수행한다. 그리고 왼쪽 로테이션 연산을 한 후, 캐리 연산이 포함된 덧셈을 수행한다.

그리고 [4]에서 설명한 것을 활용하여 효율적인 쉬프트 오프셋과 방향 알고리즘을 통해 효율적인 로테이션 연산을 제안하였다. 해당 알고리즘은 오프셋 값(o)을 4와 비교하여 4보다 큰 경우, 오프셋 값을 o-4로 설정하고, 방향을 변경합니다. 이 알고리즘을 통해 쉬프트 비트 값을 줄이고, 방향을 변경하여, 로테이션 연산을 줄인다.

[그림 2]의 빨간색 부분에 해당되는 연산들을 사전 연산을 하였다. 사전 연산 가능한 부분은 XOR 연산으로, XOR 연산은 지정된 연산 순서대로 연산할 필요가 없어 해당 부분에 대해 사전 연산을 진행하였다. 해당 연산을 사전연산을 하기 위해 SIMECK의 라운드 함수를 [그림 2]의 오른쪽과 같은 구조로 변경하였다.



[그림 2] (left) Pre-computed available part; (right) Re-organized Simeck block cipher encryption round function[3].

코드 최적화에서 속도에 대해 최적화하기 위해 loop unrolling을 사용하여 구현하였다. 이 기법을 통해 for문의 loop 비용을 줄였다.

위와 같은 기법을 적용한 성능 측정 결과는 다음과 같다. 최적화된 OTF(On-the-Fly) 속도는 평균 14.42배 빠르고, 최적화된 OTF 메모리는 레퍼런스 코드[5]보다 1.53배 적은 메모리를 사용한다. 그리고 속도와 메모리에 최적화 된 구현의 경우 [5]보다 평균 12.98배 빠름을 확인하였고, 1.3배 적은 메모리를 사용함을 확인하였다.

3.2 16-bit MSP430 상에서의 최적 구현

[6]은 16비트 환경에 적합한 SIMECK 최적 구현을 하였다. 특히, SIMECK-48/96의 워드 사이즈는 24비트로 이를 16비트와 8비트 조합으로 분리하여 최적 구현하였다. MSP430에는 8비트 크기의 데이터 연산에 대한 명령어(MOV.B, RLC.B 등)를 지원하여 효율적으로 연산하였다.

코드 사이즈 최적화는 loop rolling과 인덱스를 사용한다. 레퍼런스 코드[5]는 데이터 블록을 로드할 때 마다 인덱스 값을 연산한다. 하지만, [6]은 범용 레지스터에 데이터 블록을 저장한 후, Register mode addressing을 사용하여 데이터를 재사용하였다. 이를 통해, SIMECK-64/128의 최적 구현의 경우, 8-word 코드 사이즈를 줄였다.

속도 최적화는 loop unrolling을 사용하고, register addressing mode를 사용하여 인덱스 addressing mode의 시간을 줄였다. 이 기법을 통해 14 clock cycle을 절약하였다.

결과적으로 [5]와 비교하였을 때, 코드 사이즈 최적화 OTF는 평균 11.62%, 속도 최적화된 OTF는 평균 2.18배의 속도 향상을 보였다. 코드 크기 최적화 암호화는 평균 19.32% 더 작고 속도 최적화 암호화 속도는 평균 3.75배 빨라짐을 보였다.

3.3 32-bit ARM 상에서의 최적 구현

[7]은 ARM NEON 최적화와 NEON의 파이프라인을 고려한 OpenMP SIMT(Single Instruction Multiple Thread) 속도 최적화하였다.

ARM NEON 최적화는 다음과 같다. NEON을 사용하여 데이터 블록을 효율적으로 로드하고 저장하였다. 로테이션 연산은 왼쪽 시프트 내장 함수와 오른쪽 시프트를 사용하고 오른쪽 시프트 연산 후 데이터 삽입을 지원하는 내장 함수(vsriq_n_u16, vsriq_n_u32)를 사용하여 코드 사이즈를 줄였다.

속도 최적화는 NEON 파이프라인을 고려하여 효율적으로 NEON 레지스터 동작 순서를 재설계하였다. 그리고, omp_set_num_threads(#) OpenMP 함

수를 사용하여 thread를 설정하였다. #은 thread 수를 의미한다.

결과적으로, 제안 기법을 통해 SIMON 최적 구현[8]과 비교하였을 때, SIMECK-32/64는 41.4%, SIMECK-64/128은 46.9% 성능 향상을 보였다.

3.4 32-bit RISC-V 상에서의 최적 구현

[9]는 SIMECK-32/64와 SIMEC-64/128에 대한 단일 평문 최적 구현과 2개의 평문 최적 구현을 하였다. SIMECK의 라운드 함수 마지막에서 수행되는 블록 이동 연산이 2라운드마다 다시 되돌아오는 특징을 활용하여 블록 이동 연산을 생략하였다.

단일 평문 구현의 경우, RISC-V는 로테이션 연산을 따로 지원하지 않기 때문에, 쉬프트 연산과 xor 연산을 통해 효율적으로 구현하였다. 그리고 병렬 구현의 경우, 암호화 시작 전 하나의 레지스터에 서로 다른 평문을 위치하게 레지스터 내부 정렬을 수행하였다. 그리고 32비트에 저장된 섞이지 않아 로테이션 연산을 효율적으로 구현하였다.

결과적으로, SIMECK-32/64의 단일 평문과 2개의 평문 병렬 구현은 레퍼런스 코드 대비 각각 503% 성능 향상과 1,038% 성능 향상을 보였다. SIMECK-64/128의 단일 평문과 2개의 평문 병렬 구현은 레퍼런스 코드 대비 각각 446%와 350% 성능 향상을 보였다.

4. 결론

본 논문에서는 저사양 프로세서 상에서의 경량 블록 암호 SIMECK 최적 구현 동향에 대해 살펴보았다. 다양한 환경에서 각 환경에 적합한 SIMECK 최적 구현 연구가 활발히 진행됨을 확인하였다. SIMECK-CTR에 대한 연구는 아직까지 진행되지 않음을 확인하였다. 따라서, 향후 연구로, 저사양 프로세서 상에서의 SIMECK-CTR 최적 구현 연구를 제안한다.

5. Acknowledgement

This work was 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).

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] 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.

[3] Park, Taehwan, et al. "Efficient implementation of Simeck family block cipher on 8-bit processor." Journal of information convergence 177-183.

[4] H. Seo, Z. Liu, J. Choi, T. Park, and H. Kim, “CompactImplementations of LEA block cipher for low-endmicroprocessors,” in Proceedings of 16th International Workshopon Information Security Applications, Jeju, Korea, pp. 28-40, 2015.

[5] S. Kolbl and A. Roy, “A brief comparison of SIMON and SIMECK,” 2015 [Internet]. Available: http://eprint.iacr.org/2015/706.pdf.

[6] 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.

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

[8] T. Park, H. Seo, G. Lee, K. Md. Al-Amin, Y. Nogami, and H. Kim,“Parallel implementations of simon and speck, revisited,” in International Workshop on Information Security Applications. Springer, 2017,pp. 1-12.

[9] 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.