블록암호 PIPO에 대한 구현 동향

김현지*, 서화정*†

*한성대학교 (대학원생)

*†한성대학교 (교수)

Implementation Trends for Block Cipher PIPO

Hyun-ji Kim*, Hwa-jeong Seo*†

*Hansung University(Graduate student)

*†Hansung University(Professor)

요 약

PIPO는 ICISC'20에서 제안된 경량 블록암호이며, 본 논문에서는 PIPO에 대한 다양한 플랫폼에서의 구현 및 최적 구현 동향을 살펴본다. 저전력 프로세서 RISC-V, CPU, GPU, QPU에서의 병렬 구현 및 최적 구현들이 제시되었다. 병렬 구현의 경우, ARM의 벡터레지스터, Intel의 AVX-2, AVX-512 등과 같은 SIMD 연산을 사용하였으며, GPU 상에서의 최적 병렬 구현의 경우 PTX 인라인 어셈블리를 활용하였다. 또한, 자바스크립트 및 웹 어셈블리를 활용하여 웹 환경에서 사용가능하도록 하는 구현물도 존재한다. 마지막으로, 양자 컴퓨터 상에서 동작할 수 있도록 양자회로로 설계된 PIPO 구현도 있으며, 이는 양자 자원을 효율적으로 사용하도록 설계되었다. 이 외에도 부채널 공격에 대응하기 위한 마스킹 구현 등이 있다.

I. 서론

PIPO는 ICISC'20에서 제안된 경량 블록암호이다. PIPO와 같이 제한된 환경에서 사용하기 적합한 경량 암호 알고리즘들이 다수 개발되고 있으며, 많은 양의 데이터를 암호화 하기 위해서는 암호 알고리즘의 최적화가 필요하다. 본 논문에서는 PIPO 블록암호에 대한 다양한 플랫폼에서의 구현 및 최적 구현 기법 등의 연구 동향에 대해 살펴본다.

II. 관련 연구

2.1 PIPO 블록 암호

PIPO는 8비트 AVR 환경에서 다른 64비트 경량 블록암호를 능가하는 경량 블록암호이다[1]. 표 1은 PIPO의 파라미터이며, 64비트의 입출력과 128비트 및 256비트의 키 (64/128 및 64/256)를 가지며, 키의 길이에 따라 13라운드와 17라운드로 구성된다. 또한, SPN (Substitution Permutation Network) 구조로 설계되었으며, 라운드 키 덧셈 연산을 수행하는 AddRoundKey와 비선형 치환 연산인 S-box 연산을 수행하는 S-layer, 그리고 선형 회전 연산을 수행하는 R-layer가 라운드마다 반복된다.

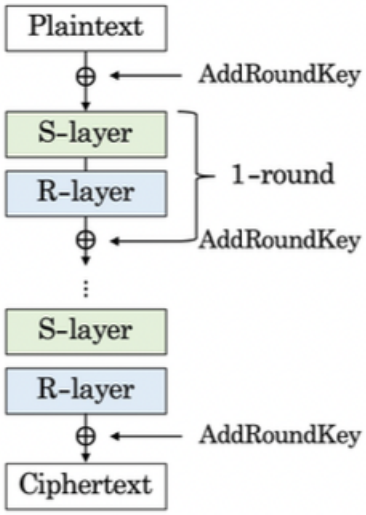

그림 1 PIPO 암호화 구조도

표 1 PIPO의 파라미터

Block size	Key size	Round
64	128	13
64	256	17

S-layer에서는 11개의 비선형 연산과 23개의 선형 비트 연산을 사용한 비트 슬라이싱 구현과 룩업 테이블을 이용하여 치환 연산을 수행하는 TLU(Table Look-up) 구현이 제공되며, PIPO는 비트슬라이싱 방식이 효율적으로 활용될 수 있도록 설계되었다. R-layer에서는 선형 회전 연산을 행 단위로 수행하며, 두 번째 행부터 7, 4, 3, 6, 5, 1, 2 비트 회전 연산을 수행한다.

III. 구현 동향

본 논문에서는 PIPO 블록암호의 구현 동향에 대해 살펴본다. 표 2는 여러 플랫폼 및 기법이 적용된 PIPO 구현에 대한 비교표이다. [2]에서는 32-bit를 지원하는 RISC-V 중 RV321 모델을 사용하여 32-bit 레지스터에 효율적인 8-bit 단위 R-layer 연산을 구현하고 메모리 최적화와 속도 최적화 측면에서 병렬 구현을 제시하였다. 또한, [3]에서는 64-bit ARM (A10x fusion 프로세서)을 사용하여 8개 및 16개의 평문에 대한 정렬을 최소화한 병렬 구현을 제시하였다. [4]는 웹상에서의 활용이 가능하도록 다양한 언어를 활용한 구현을 제시하였으며, [5]는 CPU 및 GPU 상에서의 최적 구현을 위한 병렬 처리 기법을 제안하였다. [6]에는 양자 컴퓨터 상에서의 PIPO 구현이 포함되어 있으며, [7]은 8-bit AVR 마이크로컨트롤러 상에서의 PIPO에 대한 마스킹 구현을 제시하였다.

표 2 PIPO에 대한 여러 구현물의 플랫폼 및 방식 비교

	Platform	Method
[2]	RISC-V	Parallel
[3]	ARM	
[4]	Web (i.e. Chrome, Safari)	Web implementation
[5]	Intel core i9	Parallel
[6]	Quantum computer (Simulator)	Quantum circuit

[2]에서 제시된 4평문 병렬 구현을 위해서는 S-layer 및 KeyRoundKey를 위해 레지스터 정렬 과정이 필요하다. 이러한 과정을 통해 동일 연산을 진행하는 state끼리 동일 레지스터에 모 은 후 연산을 진행하며, 필요한 사이클이 더 적 은 스택을 정렬 과정에 사용하였다. 그러나 이러 한 상태로 R-layer의 회전 연산을 수행할 경우, 다른 state와 값이 섞이게 되므로 AND 연산을 활용하여 다른 평문과 섞인 부분의 값을 0으로 바꿔준다. AddRoundKey에서는 속도 최적화를 위한 방법과 메모리 최적화를 위한 방법이 있다. 메모리 최적화는 라운드키 생성 과정을 통해 1 바이트씩 불러와진 라운드키를 32-bit로 늘린 후 연산을 수행하고, 속도 최적화 구현은 라운드키 생성 시 늘려주는 과정을 거친다. 이 경우에는 키 저장 공간이 증가하지만, 암호화 과정에서는 XOR 연산만 수행하면 되기 때문에 빠른 암호화 가 가능하다. 속도 최적화의 경우 59.93, 메모리 최적화의 경우 84.85 cpb를 달성하였다. [3]에서는 ARM 프로세서에서 제공하는 병렬 연산을 위한 벡터 명령어를 최소한으로 사용하였으며, 8평문 및 16평문 구현을 위해 각각 4개, 8개의 벡터 레지스터(128-bit)를 사용하여 병렬 구현하였다. 병렬 구현을 위해 동일 연산을 위한 state를 동일한 벡터 레지스터에 모은 후 정렬과 정을 거쳤으며, 정렬 최소화를 위해 라운드 키도 평문에 맞게 정렬한다. 이를 통해 전체 라운드에 대한 암호화를 수행하기 전과 끝난 후에 정렬을 한 번씩만 수행하도록 하였다. 또한, R-layer에 서는 회전 연산을 위해 shift 연산을 활용하였다. 결과적으로 8평문 64/128, 64/256 타입에 서 각각 8.2 cpb, 10.2 cpb의 성능을 보였으며, 16평문 64/128, 64/256 타입에서는 각각 3.9 cpb, 4.8 cpb의 성능을 달성하였다. 이는 기존 레퍼런스 구현[1] 대비 64/128, 64/256 타입에 대해 8평문 병렬 구현은 약 76.3%, 77.2%, 16 평문 병렬 구현은 약 88.7% 89.3% 더 향상된 성능을 달성하였다. [4]에서는 자바스크립트(Javascript), 웹어셈블리(WebAssembly)와 같은 웹 기반 언어를 사용하여 PIPO 64/128, 64/256에 대한 비트슬라이싱 및 TLU를 구현하였다. 또한, for문을 사용한 일반 루프 (Looped)와 for문을 사용하지 않은 루프 풀기 (Unrolled) 구현물을 제시하였다. Google

Chrome, Mozilla Firefox, Opera, Microsoft Edge 등의 다양한 웹 브라우저와 윈도우즈, Linux, Mac, iOS, 안드로이드 등의 다 양한 OS에서의 성능평가를 수행하였다. 각 브라우저별로 약간씩 다른 특징을 보였으나 Firefox 를 제외하고는 성능차이가 크지 않았다.

[5]에서는 CPU 환경과 GPU 환경에 최적화된 두 종류의 구현을 제시하였다. 최적화를 위해 CPU에서는 AVX-2, AVX-512 명령어를 사용하여 32 평문 및 64 평문에 대해 병렬 구현하였으며, GPU 환경에서는 CUDA를 활용하였다. 특히, GPU 아키텍처의 특성을 고려하여 데이터 정렬 및 결합 기법과 PTX 인라인 어셈블리를 활용하여 병렬 처리를 제안하였다. Intel Core i9-11900K (3.50GHz)에서 실험하였으며, AVX-2, AVX-512를 활용한 구현은 레퍼런스 코드에 비해 각각 839.64%, 985.46%의 성능 향상을 보였으며, RTX 2080Ti에서의 GPU 구현은 최대 1110.08Gbps의 처리량을 보여주었다. [6]에서는 PIPO에 대한 최적화된 양자 회로를 제시하였다. 특히, 8-bit S-box를 위한 최적화된 양자회로를 설계하였으며, IBM에서 제공하는 양자 시뮬레이터 ProejctQ에서 이에 대한 Grover search 알고리즘의 양자 자원을 평가하였다. 제안된 PIPO 양자 회로에서는 평문 및 마스터 키에 대한 큐비트만 할당되며, 64/128를 위해 192 큐비트가 사용되며, 64/256에는 320큐비트가 사용되었다. AddRoundKey 과정에서는 XOR연산을 위해 평문과 라운드키에 대해 CNOT 게이트를 사용하였고, 5-bit S-box 및 3-bit S-box에서는 CNOT 게이트, Tofffoli 게이트 및 X 게이트를 사용하여 구현되었으며, 추가 큐비트가 필요하지 않았다. 특히, 3큐비트 S-box 최적화를 위해 겹치는 부분을 제거하여 X 게이트 사용을 최소화하였다. 또한, 5-bit S-box는 제대로 동작하기 위해 추가적인 3 큐비트가 필요하지만, 새로운 S-box (S-box1, reverse S-box1, S-box2) 사용하여 추가 큐비트가 필요하지 않도록 하였다. S-box1은 전체 S-box의 출력에 필요한 3-bit를 생성하도록 최적화되어있으며, 이후 reverse S-box1를 거쳐서 S-box1에 입력되기 전의 상태로 되돌린다. 그 다음으로 S-box2를 적용하여 5-bit의 출력을 얻는다. 이처럼 5-bit Sbox의 역할을 두 개의 S-box와 reverse 연산을 활용하여 최적화 하였다.

IV.결론

본 논문에서는 블록암호 PIPO에 대한 구현 동향을 살펴보았다. 효율적인 암호화를 위한 병렬처리 기법 및 다양한 언어로의 구현물이 존재하며, 저전력 프로세서, CPU, GPU 그리고 QPU (양자 컴퓨터)를 위한 최적 구현이 다수 연구되었다. 이외에도 부채널 공격에 대응하기 위한 8-bit AVR 프로세서 상에서의 마스킹 구현[7] 등 다양한 구현 방법론이 제시되었다.

V.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, 100%).

[참고문헌]

[1] Kim, Hangi, et al. "PIPO: A lightweight block cipher with efficient higher-order masking software implementations." International Conference on Information Security and Cryptology. Springer, Cham, 2020.

[2] Eum, Si-Woo, et al. "Optimized parallel implementation of Lightweight blockcipher PIPO on 32-bit RISC-V." Proceedings of the Korea Information Processing Society Conference. Korea Information Processing Society, 2021.

[3] Eum, Si Woo, et al. "Optimized implementation of block cipher PIPO in parallel-way on 64-bit ARM Processors." KIPS Transactions on Computer and Communication Systems 10.8 (2021): 223-230.

[4] Lim, Se-Jin, et al. "Implementation and performance evaluation of PIPO lightweight block ciphers on the web." Journal of the Korea Institute of Information and Communication Engineering 26.5 (2022): 731-742.
[5] Choi, Hojin, and Seog Chung Seo. "Efficient Parallel Implementations of PIPO Block Cipher on CPU and GPU." IEEE Access 10 (2022): 85995-86007.
[6] Jang, Kyungbae, et al. "Grover on PIPO." Electronics 10.10 (2021): 1194.
[7] Kim, Hyunjun, et al. "Masked Implementation of PIPO Block Cipher on 8-bit International Information Springer, Cham, 2021. AVR Microcontrollers." Conference on Security Applications.