64-bit ARMv8상에서의 KpqC 후보 알고리즘 HAETAE 고속 구현

심민주*, 이민우*, 김상원*, 서화정*†

*한성대학교 (대학원생)

*† 한성대학교 (교수)

High-Speed Implementation of KpqC candidate algorithm HAETAE on 64-bit ARMv8 Processor

Min-Joo Sim*, Min-Joo Lee*, Sang-Won Kim*, Hwa-Jeong Seo*†

*Hansung University(Graduate student)

*† Hansung University(Professor)

요 약

HAETAE는 국내에서 진행하고 있는 양자내성암호 표준화 공모전 Round1에 선정된 격자 기반 전자서명 알고리즘이다. HAETAE는 2023년 7월에 시작된 NIST PQC 전자서명 추가 후보군 알고리즘에 선정되었다. 따라서, 본 논문에서는 ARMv8 프로세서 상에서의 HAETAE를 고속 구현하였다. 곱셈 연산의 일부 모듈과 AES 암호화를 ARM 명령어로 구현하여 연산 속도를 빠르게 하는 것에 중점을 두었다. HAETAE 암호 알고리즘에 구현 결과를 적용하였을 때, 서명 생성 알고리즘의 경우, 1.5배, 서명 검증 알고리즘의 경우, 1.22배 성능 향상을 확인하였다.

I. 서론

NIST(National Institute of Standards and Technology)에서는 양자컴퓨터가 빠르게 발전함에 따라 이를 대비하기 위해 양자내성암호 표준화 공모전(PQC)이 진행되고 있다[1]. 국내에서도 독자적인 양자내성암호 알고리즘을 개발하기 위해 KpqC 공모전이 2021년 말부터 진행 중이다[2]. NIST PQC 공모전 표준화 및 후보군 알고리즘에 대한 최적화 연구가 활발히 진행되고 있다[3]. 이처럼 KpqC 공모전 후보 알고리즘에 대한 최적화 연구도 활발히 진행되어야 한다. KpqC 공모전의 후보 알고리즘 중 하나인 HAETAE는 2023년 7월에 개최된 NIST PQC 추가 전자서명 알고리즘 1라운드에 선정되었다. 따라서, 본 논문에서는 HAETAE를 64-bit ARMv8 프로세서 상에서의 고속 구현을 제안한다. 제안하는 기법은 HAETAE에서 사용되는 몽고메리 reduce 연산 모듈을 고속 구현하였다.

II. 관련 연구

2.1 HAETAE

HAETAE는 Learning With Error (LWE)와 Short Integer Solution (SIS)의 어려움에 기반한 격자기반 서명 알고리즘이다[4].

Scheme	HAETAE 120	HAETAE 180	HAETAE 26-
n	256	256	256
q	64,513	64,513	64,513
S	293.51	385	457.01
η	1	1	1
τ	39	49	60

[표 1] Parameters of HAETAE

HAETAE는 NIST PQC 표준인 CRYSTALS-DILITHIUM에서 영감을 받아 설계되었지만, rejection sampling에 대해 Bimodal distribution을 사용하며, Hyperball Uniform

Distributions를 사용한다는 특징이 있다.

2.2 64-bit ARMv8 Processor

ARM(Advanced RISC Machine)은 ISA(Instruction Set Architecture) 고성능 임베디드 프로세서이다[5]. ARMv8-A는 31개의 64-bit general 레지스터(x0~x30)과 128-bit vector 레지스터(v0~v31)를 지원한다. [표 2]는 본 논문에서 사용된 ARM 명령어를 요약한 것이다.

ASM	Description	Operation
MOV	Move	Xd←Xn
MVN	Bitwise NOT (shifted register)	Xd←Xn<<#shift
ASR	Arithmetic shift right (immediate)	Xd←Xn>>#shift
ORR	Bitwise OR (shifted register)	Xd←Xn|(Xm <<#amount or Xm >>#amount)
AND	Bitwise AND (shifted register)	Xd←Xn&(Xm <<#amount or Xm >>#amount)
ADD	Add	Xd←Xn+Xm
SUB	Subtract	Xd←Xn-Xm
RET	Return from subroutine	Return

[표 2] Summary of instruction set used in implementation for ARMv8 architecture.

III. 제안 기법

제안하는 기법은 HAETAE의 곱셈 연산의 일부 하위 모듈을 구현하였다. 그리고 ARM에서 제공하는 AES 암호화 가속 명령어를 사용하여 고속 구현하였다.

소스코드는 HAETAE[4]에서 배포한 코드를 사용하였다. 하지만, ARMv8 상에서의 HAETAE을 구현하기 위해 해당 코드의 OpenSSL 의존성이 제거된 KPQClean에서 제공하는 코드를 사용하였다[6].

HAETAE는 Finite field 상에서의 연산이 수행된다. [표 3]은 32비트 reduce 연산을 구현한 것의 일부이다. 모듈러 연산을 위한 2Q의 값을 MOV 명령어와 배럴 쉬프트 명령어인 ORR 명령어를 활용하여 구현하였다. ORR 명령어를 통해 16비트 값을 왼쪽 쉬프트 연산 한 후, OR 연산이 수행된다. Reduce 연산을 수행하기 위해서는 MUL, ASR, SUB, ADD 등의 명령어를 통해 구현하였으며, NOT 연산이 수행되기 때문에 NOT 명령어인 MVN을 통해 효율적으로 구현하였다.

```
.macro asm_reduce32_2q
    mk_DQREC
    mk_DQ
    mk_Q

    mul x5, x0, x4
    asr x5, x5, #32
    mul x6, x5, x7
    sub x5, x0, x6

    asr x9, x5, #31
    mul x10, x7, x3
    and x11, x9, x10
    add x5, x5, x12

    sub x8, x5, x7
    asr x8, x8, #31
    mvn x9, x8
    and x10, x9, x7
    sub x5, x5, x10

    sub x8, x5, x3
    asr x8, x8, #31
    mvn x9, x8
    and x10, x9, x3
    sub x5, x5, x10
.endm
```

[표 3] Source code of reduction with double Q implementation.

[표 4] 는 finite field 요소에 대해 standard representative r을 구하기 위한 freeze 연산이다. freeze 연산도 reduce연산과 유사하게 구현하였다.

AES 가속기는 ARMv8에서 제공하는 SIMD 명령어 집합으로, AES 암복호화 성능을 향상시키기 위해 제공된다.

```
.macro freeze
    mk_Q_rec
    mk_DQ
    mk_Q

    mul x5, x0, x4
    asr x5, x5, #32
    mul x6, x5, x3
    sub x5, x0, x5

    and x8, x7, x5, asr #31
    add x5, x5, x8

    sub x8, x5, x3
    asr x8, x8, #31
    mvn x9, x8
    and x10, x9, x3
    sub x5, x5, x10
.endm
```

[표 4] Source code of computation standard representative implementation.

AES 암호화는 HAETAE의 난수 생성 프로세스에서 사용된다. 본 논문에서는 AES 가속기를 활용하여 AES 암호화 고속 구현을 하여 AES 암호화의 전체적인 연산을 고속화 하였다. [표 4]는 AES 암호화 가속기 명령어이다.

ASM	Description
AESE	AES single round encryption
AESMC	AES mix columns

[표 4] AES encryption instructions supported by ARMv8.

IV. 성능평가

성능 측정은 ARMv8 아키텍처를 사용하는 Apple M1칩이 탑재된 2020년형 MacBook Pro 13에서 성능 측정을 하였다. xcode상에서 구현을 진행하였으며, 최적화 옵션은 -O3(i.e. fastest)를 사용한다. 성능 측정은 HAETA-120의 구현에 대하여 10,000번 반복 동작한 시간(ms)과 Clock cycle(cc)을 측정하였다. 성능 측정 결과는 [표 5]와 같다.

HAETAE -120		Keygen	Sign	Verify
Ref	ms	2,705	13,546	413
	cc	865,600	4,334,720	132,160
This work	**ms**	**2,702**	**9,030**	**339**
	cc	**864,640**	**2,889,600**	**108,480**

[표 5] Performance evaluation result of HAETAE algorithm.

모듈 고속 구현과 AES 고속 구현한 결과를 레퍼런스 코드와 비교한 결과는 키 생성 알고리즘의 경우, 1.00배, 서명 생성 알고리즘의 경우, 1.5배, 서명 검증 알고리즘의 경우, 1.22배 성능 향상을 확인하였다.

V. 결론

본 논문에서는 KpqC Round1에 진출한 격자 기반 전자서명 알고리즘인 HAETAE를 ARMv8 프로세서 상에서 고속 구현하였다. HAETAE의 곱셈 연산 내부 모듈에 대해 고속 구현을 하였으며, ARM에서 지원하는 AES 가속기를 활용하여 AES 암호화를 고속 구현하였다. 결과적으로, 곱셈 연산의 일부 모듈 고속 구현과 AES 암호화 알고리즘 구현 결과, 서명 생성 알고리즘에서는 1.5배, 서명 검증 알고리즘에서는 1.22배 성능 향상을 확인하였다. 향후 연구로, ARMv8 상에서 HAETAE의 모든 파라미터에 대한 NTT 곱셈 최적 구현 연구를 제안한다.

VI. Acknowledgement

This work was 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%) and 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, 50%).

[참고문헌]

[1] NIST Post-Quantum Cryptography, [Online] https://csrc.nist.gov/projects/post-quantum-cryptography.

[2] 양자내성암호연구단, [Online] https://kpqc.or.kr/

[3] Post-Quantum Cryptography: Digital Signature Schemes, [Online] https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

[4] Cheon, Jung Hee, et al. "HAETAE: Shorter Lattice-Based Fiat-Shamir Signatures." Cryptology ePrint Archive (2023).

[5] Armv8-a instruction set architecture. https://developer.arm.com/documentation/den0024/a/An-Introduction-to-the-ARMv8-Instruction-Sets. Accessed: 2023-08-16.

[6] KPQClean. Available online: https://github.com/kpqc-cryptocraft/KPQClean.Accessed: 2023-08-16.