64-bit ARMv8상에서의 KpqC 후보 알고리즘 GCKSign 고속 구현

https://youtu.be/rrItLHD9oI0

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

관련연구

제안 기법

성능 평가

결론

KpqC 공모전

- 2021년 말부터 국내 양자내성암호 표준 선정을 위한 KpqC 공모전이 진행 중
- 2022년 12월 1라운드 후보 알고리즘 선정
  - 7개의 공개키와 9개의 전자서명

KpqC 일정	
2021.11.25	공모전 공지
2022.02.18	공모전 제출 마감
2022.03.15	1차 평가
2023.12	1 라운드 결과 발표 2 라운드 후보 목록 공개
2024.03	2라운드 결과 발표
2024.09	KpqC 표준 알고리즘 발표

Type	Code-based	Lattice-based	Multivariate Quadratic-based	Hash-based	Zero-knowledge
Encryption/Key-establishment	IPCC	NTRU+	-	-	-
	Layered ROLLO	SMAUG			
	PALOMA	TiGER			
	REDOG				
Digital Signature	Enhanced pqsigRM	**GCKSign**	MQ-Sign	FIBS	AIMer
		HAETAE			
		NCC-Sign			
		Peregrine			
		SOLMAE			

KpqC 공모전

- 2021년 말부터 국내 양자내성암호 표준 선정을 위한 KpqC 공모전이 진행 중

NIST PQC 표준화 알고리즘에 대한 최적화 연구가 활발히 진행 중

KpqC 공모전 후보 알고리즘에 대한 최적화 연구도 활발히 진행되어야 함!

		REDOG	TIGER			
Digital Signature	Enhanced pqsigRM		GCKSign	MQ-Sign	FIBS	AIMer
			HAETAE			
			NCC-Sign			
			Peregrine			
			SOLMAE			

2022.03.15	1차 평가
2023.12	1 라운드 결과 발표 2 라운드 후보 목록 공개
2024.03	2라운드 결과 발표
2024.09	KpqC 표준 알고리즘 발표

GCKSign

- 격자 기반 전자 서명 알고리즘
- Discrete Gaussian distribution에 대한 샘플링 불필요
  - NIST PQC 표준으로 선정된 Falcon과 CRYSTALS-Dilithium 보다 간단하고 효율적
- Target-Modified One-wayness(TMO) 문제를 개량하여 사용
- 서명과 키 크기를 줄이기 위해 오류 함수 분석과 Central Limit Theorem(CLT) 사용
- **서명 체계의 단순성으로 인해 빠르고 효율적인 구현 가능**

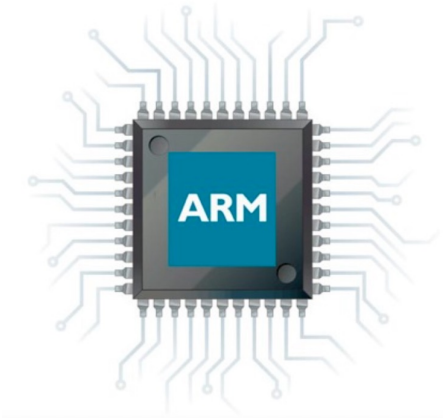
Scheme	n	m	q	η	Public key $(byte)$	Secret key $(byte)$	Signature $(byte)$
GCKSign II	256	4	$\approx 2^{54}$	1	1,760	288	1,952
GCKSign III	256	4	$\approx 2^{60}$	1	1,952	288	2,080
GCKSign V	512	3	$\approx 2^{44}$	1	3,040	544	3,104

(m : dimension of pk, q : modulus, η : secret polynomial range)

ARMv8프로세서

- ARM(**A**dvanced **R**ISC **M**achine)
  - ISA(Instruction Set Architecture) 고성능 임베디드 프로세서
  - 1958년 Acorn사에서 개발한 ARM1에서 시작
    추후 ARM Holdings를 설립하여 개발 시작

- ARMv8 프로세서
  31개의 64비트 general 레지스터(x0~x30)와
  128-bit 32개 벡터 레지스터(v0~v31) 지원



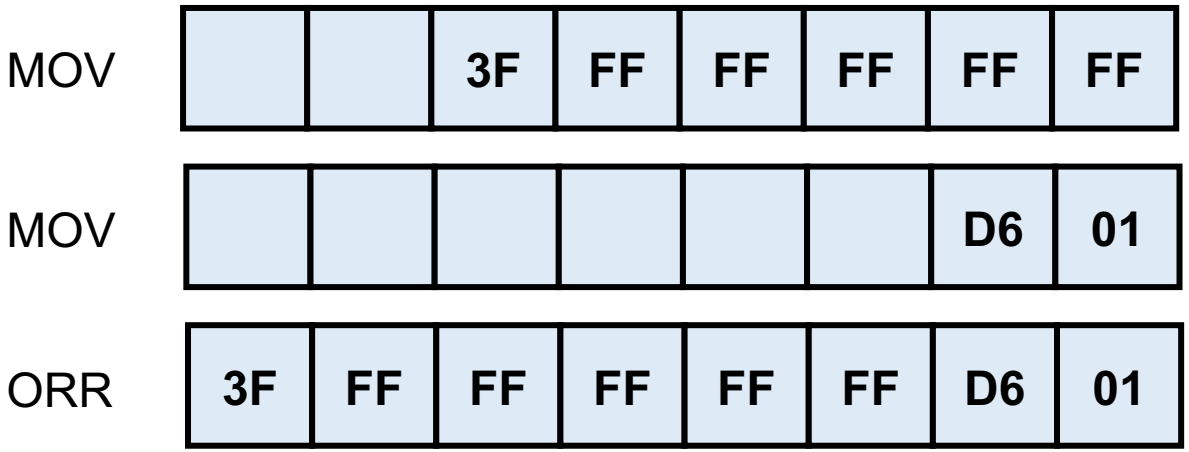
ASM	Description	Operation
MOV	Move	Xd←Xn
LSL	Logical Shift Left (immediate)	Xd←Xn<<#shift
ASR	Arithmetic shift right (immediate)	Xd←Xn>>#shift
ORR	Bitwise OR (shifted register)	Xd←Xn|(Xm <<#amount or Xm >>#amount)
AND	Bitwise AND (shifted register)	Xd←Xn&(Xm <<#amount or Xm >>#amount)
ADD	Add	Xd←Xn+Xm
SUB	Subtract	Xd←Xn-Xm
RET	Return from subroutine	Return

제안 기법

- ARMv8 상에서 GCKSign를 구현하기 위해 레퍼런스의 **OpenSSL 의존성 제거**
  - **PQClean에서 제공하는 AES와 SHA2**로 교체
- **몽고메리 연산과 NTT 곱셈 연산의 하위 모듈 고속 구현**
  - General 레지스터를 사용하여 곱셈 연산을 ARM 명령어를 활용하여 고속 구현
- ARM에서 제공하는 **AES 암호화 가속 명령어를 사용**한 AES 고속화

제안 기법

- **몽고메리 연산 구현**
- 모듈러 Q가 하나의 레지스터에 위치하도록 연산 수행
  - 하나의 레지스터에 MOV 명령어를 사용하여
  레지스터에 값을 지정할 수 있는 범위는 고정적



- 몽고메리 연산을 위한 모듈들을 ADD, SUB 명령어를 사용하여 효율적으로 구현

```
.macro mk_Q
    mov x3, #0x3FFFFFFFFF
    mov x2, #0xD601
    orr x3, x2, x3, lsl #16
.endm

.macro caddp
    and x12, x3, x0, asr #63
    add x0, x12, x0
.endm

.macro csubp
    mk_Q
    sub x0, x0, x3
    caddp
.endm
```

[표 3] Source code used for montgomery operation implementation.

제안 기법

- **NTT 곱셈의 하위 모듈 구현**
- NTT 하위 모듈에는 몽고메리 곱셈 연산 포함
- 모듈러 Q의 Inverse 값도 동일하게 하나의 레지스터에 위치하도록 MOV, LSL, ORR 명령어를 사용하여 구현
- **몽고메리 곱셈 연산**
  - MUL, SUB, ASR, AND, ADD 명령어를 사용하여 수행

```
.macro mk_Q_Inv
    mov x3, #0x4460
    lsl x3, x3, #48
    mov x4, #0x1c31
    orr x3, x3, x4, lsl #32
    mov x4, #0x6ee4
    orr x3, x3, x4, lsl #16
    mov x4, #0x2a01
    orr x4, x3, x4
.endm
.macro mont_ivt
    mul x10, x5, x4
    mul x11, x10, x3

    sub x11, x5, x11
    asr x11, x11, #63
    asr x11, x11, #1

    caddp
    sub x11, x11, x3

    and x12, x3, x11, asr #63
    add x0, x12, x11
.endm
```

[표 4] Source code used for NTT multiplication implementation.

제안 기법

- **AES 암호화 구현**
- AES-NI는 Intel에서 제안한 x86 명령어 집합의 확장
  - AES의 암호화와 복호화의 성능을 향상시키기 위해 제공
- GCKSign의 **키 생성, 서명 생성, 서명 검증 알고리즘에서 AES 암호화 사용**

→ ARMv8에서 지원하는 AES 암호화 명령어를 활용하여 AES 암호화 고속화

ASM	Description
AESD	AES single round decryption
AESE	AES single round encryption
AESIMC	AES inverse mix columns
AESMC	AES mix columns

[표 5] AES encryption instructions supported by ARMv8.

성능평가

- 성능 측정 환경
  - ARMv8 아키텍처를 사용하는 Apple M1chip이 탑재된 MacBook Pro 13(2020)에서 성능 측정
  - 반복횟수 : 10,000
  - 최적화 옵션 : -O2 Level

- 알고리즘별 성능 비교
  - 키 생성, 서명 생성, 검증 알고리즘에 적용한 결과
    - **곱셈 연산(몽고메리 + NTT 곱셈의 일부 하위 모듈) 고속 구현**

      서명 생성 알고리즘의 경우, 2.54배 성능 향상
    - **곱셈 연산 + AES 가속기 고속 구현**

      키 생성 알고리즘의 경우, 1.28배 성능 향상

      서명 생성 알고리즘의 경우, 2.70배 성능 향상

GCKSign -II		Keygen	Sign	Verify
Ref	ms	436	1,695	391
	cc	139,520	542,400	125,120
This work	**ms**	**434**	**678**	**399**
	cc	**138,880**	**216,960**	**127,680**
This work(A)	**ms**	**340**	**630**	**398**
	cc	**108,800**	**201,600**	**127,360**

[표 6] Performance evaluation result of GCKSign-II algorithm; Notation(A) indicates with optimization technique using AES-NI.

결론

- 본 논문에서는 KpqC Round 1 후보 알고리즘인 GCKSign 고속 구현
  - ARMv8 프로세서의 general register를 활용
  - GCKSign의 **몽고메리 연산**과 **NTT 곱셈의 일부 하위 모듈**에 대한 최적 구현
  - AES-NI를 활용하여 **AES 암호화 고속화**
  - 곱셈 연산 + AES 암호화 고속 구현을 알고리즘에 적용하였을 때 성능 결과
    - **키 생성 알고리즘의 성능 개선 1.28배**
    - **서명 생성 알고리즘의 성능 개선 2.70배**
- 향후 연구로, ARMv8 상에서의 KpqC 후보 알고리즘에 대한 최적 구현 연구 제안

Q & A