비밀 의존성 기반 분할을 통한 임베디드 TrustZone-A 환경의 경량 ML-KEM 아키텍처

2026 한국컴퓨터종합학술대회

노경민, 권나희, 박수현, 조성제
Computer Security & OS Lab.

2026.06.24

Contents

PART 01

Introduction

배경과 문제점

Background

양자컴퓨터가 오기 전에, 임베디드 기기는 지금 양자 내성 암호 (PQC)로 옮겨야 한다.

- 양자 컴퓨터는 Shor 알고리즘으로 현재 임베디드 기기들이 주로 사용하는 RSA와 ECC 등의 공개키 암호를 무력화시키는 것으로 알려짐.
- 양자 공격은 “Harvest Now, Decrypt Later” 라는 캐치프레이즈 아래에서 현재 진행형임.
- IoT 및 임베디드 기기는 전력망, 의료, 산업 제어 등에서 10~20년 동안 장기 운용됨. 따라서 기기 수명이 암호 수명보다 훨씬 길.
- 배포 후 펌웨어 갱신이 어려운 기기가 많아, 장기 민감 데이터를 지키기 위해서는 지금부터 PQC를 내장해야 함.

→ 2024년 8월 NIST가 ML-KEM을 양자 내성 대칭키 분배 암호의 표준으로 확정했음. 또한 ML-KEM 암호 이외에도 ML-DSA 등의 시그니처 기반 양자 내성 암호도 표준화하였음. 이 배경에서도 알 수 있듯이, 양자컴퓨터가 오기 전에 신속하게 PQC로 대체할 필요가 있음.

Problem

두 가지 배치 방식은 보안과 성능을 동시에 잡지 못한다.

Full-REE (Rich Execution Environment)

모든 연산을 Normal World에서 수행

- 성능: 빠르며, 추가 격리 비용이 없음.
- 보안: Normal World가 손상되면 비밀키가 노출되기 때문에 취약함.

Full-TEE (Trusted Execution Environment)

모든 연산을 Secure World에서 수행

- 보안: 보호 경계가 명확하고, Normal World에 비해 안전함.
- 성능: 모든 PQC 알고리즘 코드가 보호 경계 내부에 들어가기 때문에 TCB (Trusted Computing Base)가 증가하며, 경계를 넘나드는 등의 추가 비용이 발생해 성능이 낮아짐.

한계 Full-REE는 보안이 취약하고, Full-TEE는 성능을 잡지 못한다.

핵심 아이디어: 어떻게 문제를 해결할 것인가?

ML-KEM은 비밀 연산과 공개 연산으로 명확히 나뉜다.

ML-KEM은 비밀 시드, noise, 비밀 벡터 연산과 반복적인 공개 polynomial(NTT, matrix-A) 연산이 구조적으로 분리된다. 이 특성을 그대로 Secure World와 Normal World 분리에 활용한다.

기존 방식, Full-TEE

Secure World

ML-KEM 전체 연산
(비밀과 공개)

제안 방식, Split

Normal World

공개 연산
(NTT, matrix-A)

Secure World

비밀 연산
seed, s, e

비밀 의존 연산만 Secure World에 고립시키고 계산량이 큰 공개 연산은 Normal World로 이전한다.
보호 경계는 유지하면서 TEE 부담을 줄인다.

PART 02

Related Work

기존 연구가 이 문제를 어떻게 해결하는가

Related Work

기존 연구가 어떻게 문제를 해결했는가

1	**TEE 기반 암호 분할 및 TCB 최소화**	**ReZone, USENIX Sec '22** 비밀 연산을 Secure World에 격리하고, Secure World의 권한·코드 범위를 줄여 TCB를 최소화 → RSA/ECC 중심이라 ML-KEM의 대규모 다항식 연산·비밀 의존성을 담기 어려움
2	**임베디드 PQC 구현 및 성능 평가**	**Kyber on ARM64, IACR TCHES '21 · CRYSTALS-Kyber, EuroS&P '18** 표준 격자 기반 KEM(ML-KEM / FIPS 203)을 Cortex-A 등 임베디드에서 구현·최적화·평가 → 일반 실행 환경 중심이라 Normal World 손상 시 비밀 노출 문제는 미해결
3	**TEE 기반 PQC 배치 및 Full-TEE**	**Post-Quantum Algorithms on ARM TEE, LADC '24** 비밀키 보호를 위해 ML-KEM 같은 PQC 연산을 Secure World에 배치 (Full-TEE) → 공개 연산까지 TCB에 포함되어 Secure World 코드 규모와 실행 지연이 증가

Research Question

우리가 해결하고자 하는 연구 질문

기존 방식은 보안(Full-REE)과 경량성(Full-TEE) 중 하나만 만족한다. 둘을 함께 달성하려면 무엇을 어디에 둘지 정하는 분할 기준이 필요하다.

Research Question

임베디드 TrustZone-A 환경에서 ML-KEM의 장기 비밀키와 shared secret을 Secure World 밖으로 노출하지 않으면서, Secure World의 TCB와 실행 부담을 함께 줄일 수 있는가?

PART 03

Split-based Architecture

위협 가정 위에서, 무엇을 어떻게 분리해야 하는가

위협 모델

Nomal World는 손상될 수 있다고 가정한다.

구분	대상
신뢰함 (TCB)	Secure Monitor, Trusted OS, Secure World 내부 암호 모듈, TrustZone-A 하드웨어 격리
신뢰하지 않음 (공격자 통제)	Normal World 전역(OS, 앱, 드라이버) 제어, Normal World 메모리와 코드 임의 읽기 쓰기, 공개 중간값 변조, 시스템 콜 흐름 교란, Secure World 입력과 포맷과 호출 순서 조작, rollback 재주입
범위 외 (보호 대상 아님)	TrustZone-A 하드웨어 결함, Secure Monitor와 OS 수준 취약점, 마이크로아키텍처 부채널

위협 모델

핵심 공격면은 SW 진입점, 입력과 순서의 재주입을 막는다.

공격	제안 구조의 대응
입력 교란 **ciphertext, seed, command ID 변조** **(변조 공격)**	Secure World가 모든 입력에 형식, 범위, 길이, 상태 검증을 강제하고, secret 경로에 영향을 주는 값은 Secure World 내부에서만 재생성
호출 순서, 패턴 조작 **SMC 순서 변경이나 반복 호출** **(호출 조작 및 DoS 공격)**	entry-point별 상태 기반 제어, secret 모듈은 독립 내부 상태를 유지해 유효 흐름 외 호출은 즉시 거부, secret branching은 Secure World 내부에서만
Rollback, 재주입 **과거 ciphertext나 상태 재사용** **(Replay 공격)**	비밀은 내부 TRNG로 생성하고 Secure World가 secret 상태를 외부로 노출하지 않아, 재주입이 oracle 역할을 못 하고 rollback이 구조적으로 무력화

연산 배치

세 연산을 비밀키 의존성에 따라 배치한다.

연산	Secure World (SW), 비밀	Normal World (NW), 공개
KeyGen	내부 TRNG로 비밀 seed와 noise 생성, secret 벡터 생성과 저장, 공개키 구성용 비밀 의존 값 생성	public seed 기반 matrix 확장, 공개 데이터 encoding과 packing, 공개키 전달
Encaps	shared secret 도출과 저장, session key handle 생성	공개키 파싱, 공개 입력 기반 ciphertext 생성과 packing
Decaps	비밀키 기반 복호화, ciphertext 검증, implicit rejection, shared secret 도출과 저장	ciphertext 전달, Decaps 요청, session key handle 사용

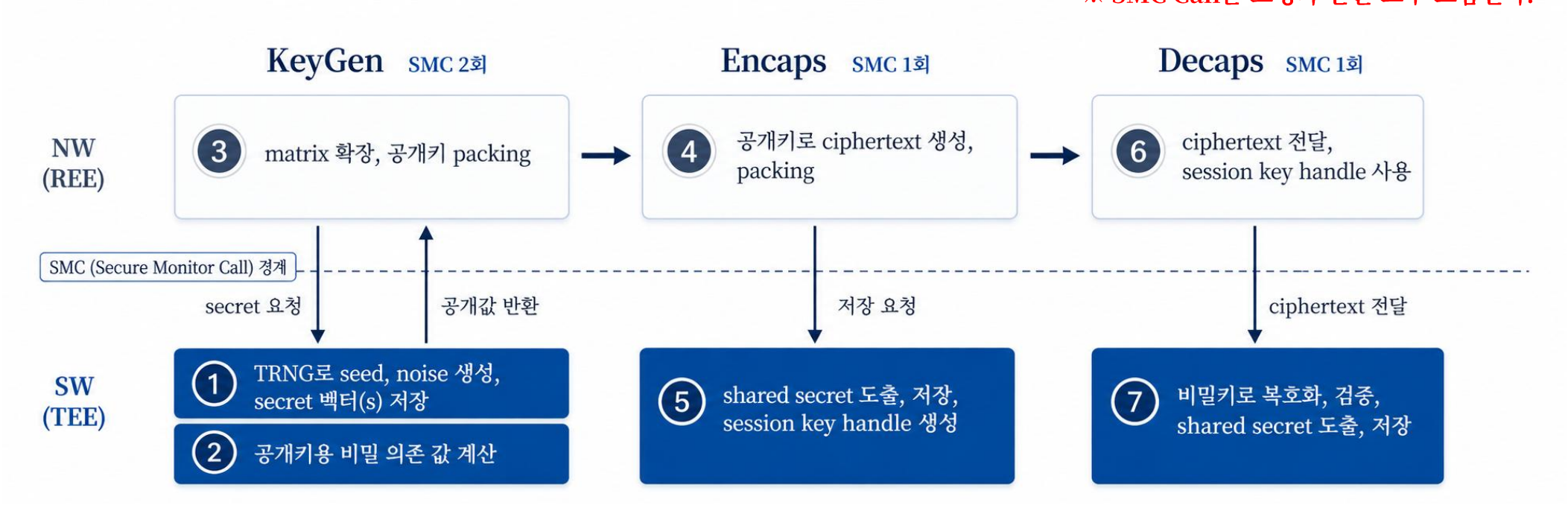
shared secret은 세션 보호용 비밀 출력이므로 Normal World로 반환하지 않고 Secure World 내부에 저장한다

비밀키 흐름

비밀키는 Secure World에서 생성되어 Secure World를 떠나지 않는다.

KeyGen → Encaps → Decaps 전 과정에서 Normal World로 건너가는 값은 공개키, ciphertext, session key handle뿐이다.
(전체 SMC 호출 4회)

※ SMC Call은 요청과 반환 모두 포함된다.



비밀 연산과 데이터 (Secure World 전용, 외부 노출 안 함)

공개 연산과 데이터 (Normal World 전달 가능)

PART 04

Implementation & Evaluation

실제 보드에서 측정한 지연, TCB, 호출 수

실험 환경

Raspberry Pi 3, OP-TEE 이용

하드웨어	Raspberry Pi 3 Model B · ARM Cortex-A53
TEE	OP-TEE
알고리즘	ML-KEM-512 (세 구조 동일 구현 기반)
비교군	Full-REE · Full-TEE · 제안 Split
측정 지표	Latency · TCB 크기(LoC · 모듈 수) · SMC Call 수
Latency 측정	KeyGen · Encaps · Decaps 각 2,000회 평균 수행 시간

Latency 측정 결과

Split은 Full-TEE보다 평균 25.5% 빠르고, Full-REE에 근접한다.

연산	Full-REE	Full-TEE	Split (제안)
KeyGen	0.919	1.561	**1.181**
Encaps	1.072	1.659	**1.104**
Decaps	1.336	1.881	**1.512**
평균	**1.109**	**1.700**	**1.266**

ML-KEM 연산별 Latency (ms, N = 2,000)

해석

- Split은 Full-TEE 대비 평균 Latency를 25.5% 줄였고, Full-REE 대비 추가 지연은 0.157 ms에 그침
- Encaps는 공개 연산 위주라 1.659에서 1.104 ms로 33.5% 개선됨
- KeyGen과 Decaps는 비밀 연산을 포함해 각각 24.3%, 19.6%로 개선 폭이 작음

TCB 측정 결과

비밀 의존 연산만 SW에 두어 TCB를 약 90.8% 줄였다.

항목	Full-TEE	Split (제안)
코드 규모 (LoC)	약 12,000	**약 1,100**
모듈 수	15+	**3**

SW Application 바이너리에 링크된 코드 기준 (주석과 공백 제외)

해석

- Full-TEE는 ML-KEM 라이브러리 전체를 SW에 포함해 12,000 LoC, 15개 이상의 모듈이 TCB에 들어감
- Split은 비밀키 의존 연산에 필요한 기능만 SW에 두어 1,100 LoC, 3개 모듈로 줄어듦
- LoC 기준 약 90.8% 감소로, 검증 부담과 공격 표면이 함께 작아짐

SMC Call 측정 결과

추가 비용은 SMC Call 호출 1회뿐, Shared Secret 보호를 위한 비용이다.

항목	Full-TEE	Split (제안)
KeyGen	1	2
Encaps	1	1
Decaps	1	1
총합	3	4

해석

- KeyGen은 비밀 상태 생성과 공개키 구성 지원이 분리되어 2회가 발생함
- Encaps는 ciphertext 공개 계산은 Normal World에서 하지만, shared secret을 Secure World에 저장하려 1회가 필요함
- Decaps는 Full-TEE와 동일하게 1회로, 총 호출은 1회만 늘어남
- 늘어난 1회는 shared secre을 Secure World에 유지하기 위한 보호 비용으로 해석할 수 있음

PART 05

Conclusion

기여 요약, 한계, 그리고 향후 연구

결론

비밀 의존성 분할로 보호 경계를 지키며 부담을 줄였다.

기여

- 비밀 의존 연산과 shared secret 생성은 Secure World에 유지하고 공개 연산은 Normal World로 위임
- Full-TEE 대비 TCB 약 90.8%, 평균 Latency 약 25.5% 감소
- 보호 경계를 유지하면서 시스템 가용성을 개선

한계 및 향후 연구

- ML-KEM 대상이라 다른 PQC는 별도 분할 분석이 필요
- side-channel과 fault injection 공격은 범위 밖
- 향후 다양한 PQC 확장과 실제 하드웨어 기반 부채널, 결함 대응 분석

비밀에 닿는 부분만 SW에 남기면, 임베디드 환경에서 ML-KEM을 안전하고 가볍게 배치할 수 있다

Q & A