Lattice Chat

양자내성 암호 기반 종단간 암호화 메신저

ML-KEM-768 (FIPS 203) · ML-DSA-65 (FIPS 204) · AES-256-GCM

seQure Lattice · 2026. 10. 02 · 컴퓨터과학전공 김정현 · 지도교수 : 박영훈 교수

목차

프로젝트 개발 배경
기능 소개
시스템 구조
ML-KEM 알고리즘 소개
경쟁 서비스 비교

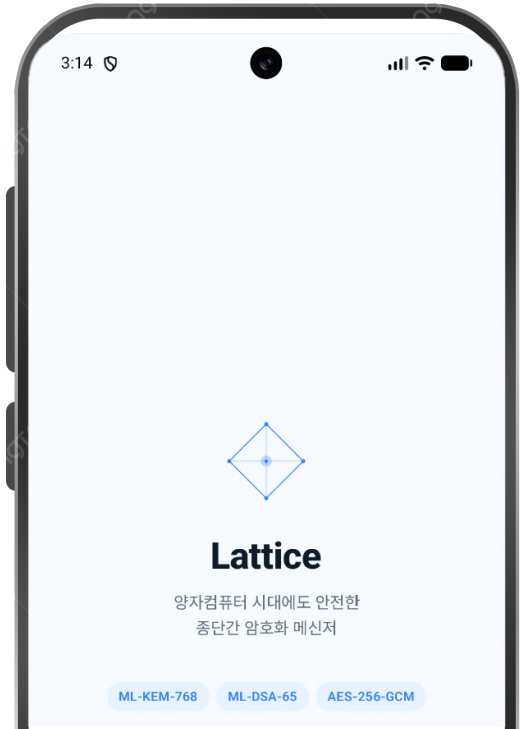
프로젝트 개발 배경

왜 지금 바꿔야 하는가

지금 쓰는 공개키 암호는 깨집니다

RSA · ECC → Shor's Algorithm

NIST(미국 표준기술개발연구소) → 2030년까지 기존 암호 사용 중단 권고



Harvest Now, Decrypt Later

'지금 오가는 암호문을 저장해 두고, 양자컴퓨터가 나온 뒤에 푼다.'

그래서 "아직 양자컴퓨터가 없으니 나중에"가 성립하지 않습니다.

오래 비밀이어야 하는 대화라면 지금 바꿔야 합니다.

프로젝트 핵심 주제

종단간 암호화 - 서버에 메시지 남지 않음

공개키 세션마다 교체- **Double Ratchet**

공개키/서명 알고리즘 → **양자내성암호 알고리즘(ML-KEM/ML-DSA)**

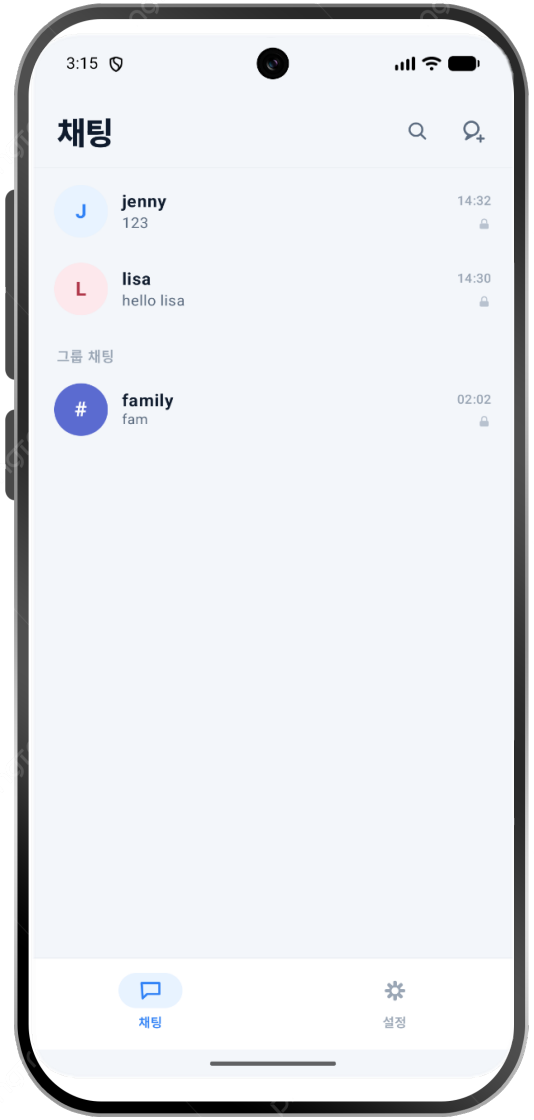
기술 스택



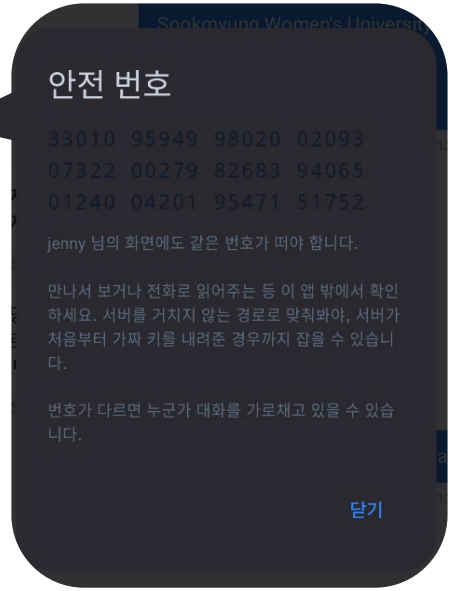
구성

- Android (Kotlin, Jetpack Compose)
- Spring Boot · PostgreSQL
- AWS EC2 · Nginx · Let's Encrypt

기능 소개

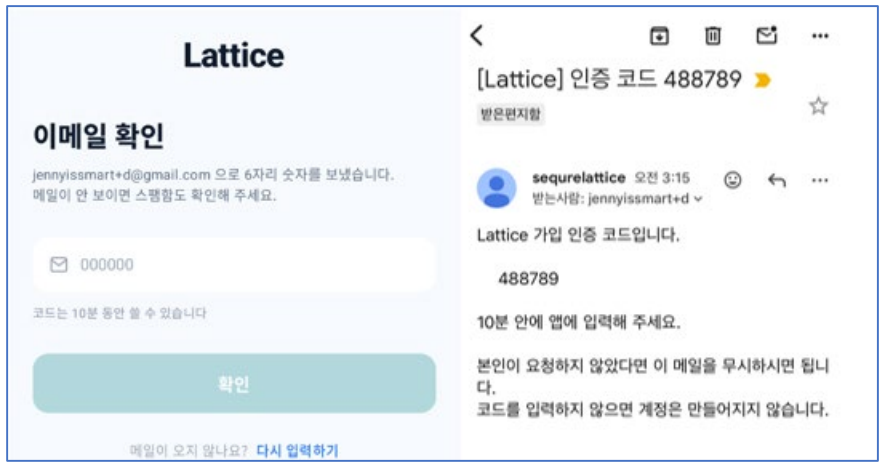

3:15
채팅
jenny
123
14:32
lisa
hello lisa
14:30
그룹 채팅
family
fam
02:02
채팅
설정


12:36
jenny
종단간 암호화 · ML-KEM-768
ML-KEM-768 · AES-256-GCM · ML-DSA-65 서명 검증
PQC based End-to-End messenger Algorithm based on: ML-KEM/ ML-DSA/AES
Sookmyung Women's University Computer Science Graduation Project, fall semester 26
Junghyun Kim
오전 12:34
양자내성암호 기반 종단 간 암호화 메신저
오전 12:35
숙명여자대학교 컴퓨터과학전공 졸업 프로젝트
26년도 가을학기
오전 12:35
Hello it's Jay.
오전 12:35
hello.
오전 12:36
오전 12:36
Double Ratchet 세션 활성화 — 매 메시지 키 자동 교체
암호화 메시지 입력

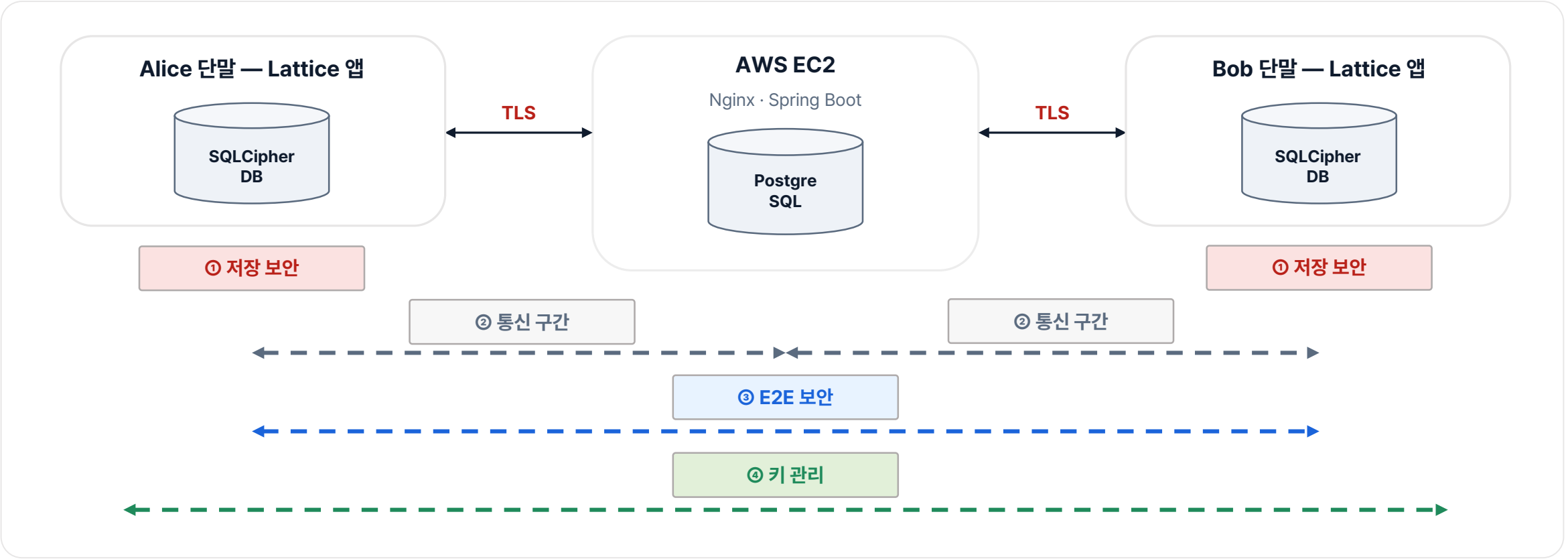

안전 번호
33010 95949 98020 02093
07322 00279 82683 94065
01240 04201 95471 51752
jenny 님의 화면에도 같은 번호가 떠야 합니다.
만나서 보거나 전화로 읽어주는 등 이 앱 밖에서 확인하세요. 서버를 거치지 않는 경로로 맞춰봐야, 서버가 처음부터 가짜 키를 내려준 경우까지 잡을 수 있습니다.
번호가 다르면 누군가 대화를 가로채고 있을 수 있습니다.
닫기


새 채팅
상대방 아이디로 채팅을 시작합니다.
jenny
검색
jenny
채팅 시작하기 →


방 코드로 입장
방장에게 받은 8자리 코드를 입력하세요
예: ABC123XY


Lattice
이메일 확인
jennyissmart+d@gmail.com 으로 6자리 숫자를 보냈습니다.
메일이 안 보이면 스팸함도 확인해 주세요.
000000
코드는 10분 동안 쓸 수 있습니다
확인
메일이 오지 않나요? 다시 입력하기
[Lattice] 인증 코드 488789
받은편지함
sequrelattice 오전 3:15
받는사람: jennyissmart+d
Lattice 가입 인증 코드입니다.
488789
10분 안에 앱에 입력해 주세요.
본인이 요청하지 않았다면 이 메일을 무시하시면 됩니다.
코드를 입력하지 않으면 계정은 만들어지지 않습니다.

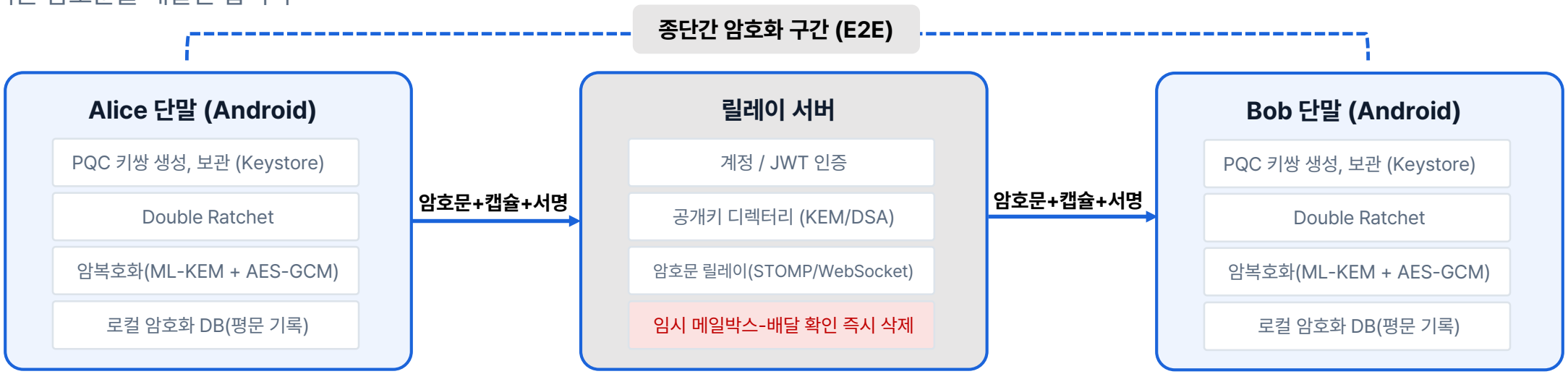
보안 아키텍처



① 저장 보안	SQLCipher · Android Keystore · 화면 캡처 차단 — 로컬 DB에 저장되는 데이터/키 정보 보안
② 통신 구간	TLS 1.2+ (Let's Encrypt) · 평문 통신 차단
③ E2E 보안	ML-KEM-768 · AES-256-GCM
④ 키 관리	기기에서 생성 · TOFU 고정 · 안전번호 · 더블 래칫 (즉시 폐기) — 가짜 키를 주는 중간자를 막음

시스템 구조 - E2E 데이터 보안

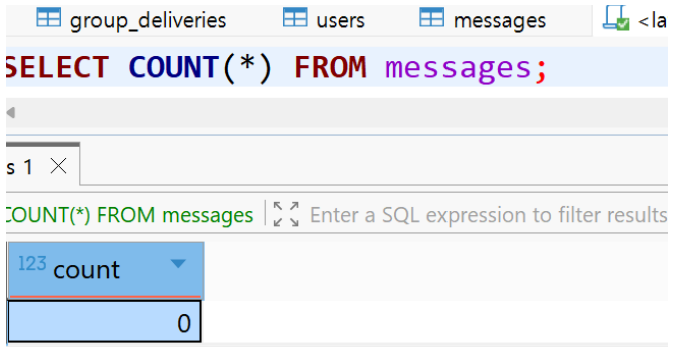
서버는 암호문을 배달만 합니다



서버가 전부 해킹되어도 얻는 것은 메타데이터와 아직 배달되지 않은 암호문 뿐입니다.

서버 DB를 직접 열어보면

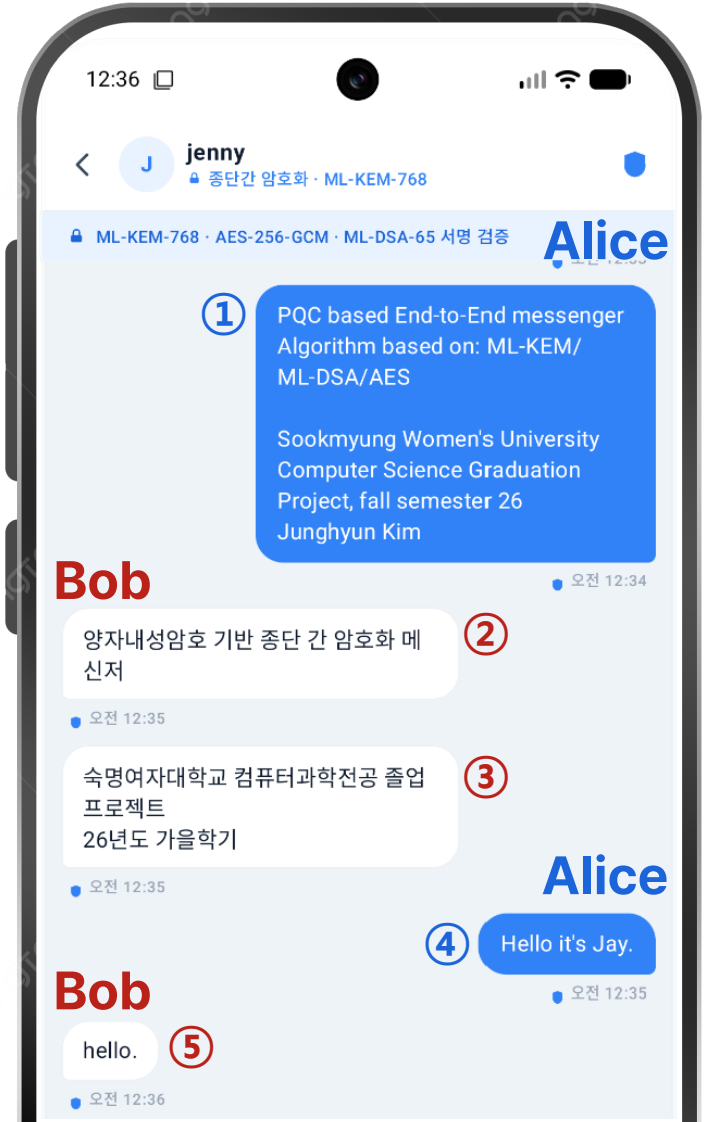
테이블	내용	읽힘
users	아이디 , 이메일 , 비밀번호 해시	해시만
user_public_keys	공개키	O
messages	미배달 암호문	X
group_messages	그룹 미배달 암호문	X
read_status	읽은 시각	O



서버는 읽는 즉시 암호문 삭제

ML-KEM과 Double Ratchet

KEM 으로 공유 비밀을 만들고 메시지마다 교체 - Forward Secrecy



서버

Alice의 ML-KEM공개키 🔑A

Bob의 ML-KEM공개키 🔑B

Alice

Bob의 공개키로 Encapsualte(**공유비밀 K**)

암호문: AES 키 암호화

✉️ 💊 🔑

Bob 🔑

(Capsule+암호문+서명+Alice의 새 래칫 공개키)

Bob의 개인키로 Decapsulate :

공유비밀 K->AES 키 유도->암호문 복호화

루트 래칫이 일어나는 시점

Alice 가 보냄 -> KEM O (키생성 + 캡슐화)

Bob이 받음 -> KEM O (역캡슐화)

Bob이 답장 -> KEM O (키생성 + 캡슐화)

Bob이 또 보냄 -> KEM X (해시만)

알고리즘 구현 참고 논문

api.sequrelattice.com — 실제로 운영 중인 서비스



2018 IEEE European Symposium on Security and Privacy

CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM

Joppe Bos*, Léo Ducas†, Eike Kiltz‡, Tancrède Lepoint§, Vadim Lyubashevsky¶,
John M. Schanck‖, Peter Schwabe**, Gregor Seiler††, Damien Stehlé‡‡,
*NXP Semiconductors, Belgium. Email: joppe.bos@nxp.com
†CWI Amsterdam, The Netherlands. Email: ducas@cwi.nl
‡Ruhr-University Bochum, Germany. Email: eike.kiltz@rub.de
§SRI International, USA. Email: tancrede.lepoint@sri.com
¶IBM Research Zurich, Switzerland. Email: vad@zurich.ibm.com
‖University of Waterloo, Canada. Email: jschanck@uwaterloo.ca
**Radboud University, The Netherlands. Email: peter@cryptojedi.org
††IBM Research Zurich, Switzerland. Email: grs@zurich.ibm.com
‡‡ENS de Lyon, France. Email: damien.stehle@ens-lyon.fr

***Abstract*—Rapid advances in quantum computing, together with the announcement by the National Institute of Standards and Technology (NIST) to define new standards for digital-signature, encryption, and key-establishment protocols, have created significant interest in post-quantum cryptographic schemes.**

This paper introduces Kyber (part of CRYSTALS – *Cryptographic Suite for Algebraic Lattices* – a package submitted to NIST post-quantum standardization effort in November 2017), a portfolio of post-quantum cryptographic primitives built around a key-encapsulation mechanism (KEM), based on hardness assumptions over module lattices. Our KEM is most naturally seen as a successor to the NewHope KEM (Usenix 2016). In particular, the key and ciphertext sizes of our new construction are about half the size, the KEM offers CCA instead of only passive security, the security is based on a more general (and flexible) lattice problem, and our optimized implementation results in essentially the same running time as the aforementioned scheme.

We first introduce a CPA-secure public-key encryption scheme, apply a variant of the Fujisaki–Okamoto transform to create a CCA-secure KEM, and eventually construct, in a black-box manner, CCA-secure encryption, key exchange, and authenticated-key-exchange schemes. The security of our primitives is based on the hardness of Module-LWE in the classical and quantum random oracle models, and our concrete parameters conservatively target more than 128 bits of post-quantum security.

1. Introduction

There has been an increased interest in post-quantum cryptographic schemes triggered by recent advances in quantum computing [35] and the announcement by the National Institute of Standards and Technology (NIST) to define new standards for digital-signature, encryption, and key-establishment protocols [28]. Constructions based on the hardness of lattice problems are considered to be one of the leading candidates to replace the currently used schemes based on the believed hardness of the traditional number-theoretic problems such as integer factorization and discrete logarithms.

Lattice cryptography initially gained a lot of interest in the theoretical community due to the fact that the designs for cryptographic constructions were accompanied by security proofs based on *worst-case* instances of lattice problems. The first lattice-based encryption scheme was proposed by Ajtai and Dwork [1]. This scheme was later simplified and improved upon by Regev in [67], [68]. One of the major achievements of Regev's work was the introduction of an intermediate problem – the Learning With Errors (LWE) Problem – which was relatively simple to use in cryptographic constructions and asymptotically at least as hard as some standard worst-case lattice problems [61], [25].

The LWE assumption states that it is hard to distinguish from uniform the distribution $(\mathbf{A}, \mathbf{As} + \mathbf{e})$, where $\mathbf{A}$ is a uniformly-random matrix in $\mathbb{Z}_q^{m \times n}$, $\mathbf{s}$ is a uniformly-random vector in $\mathbb{Z}_q^n$, and $\mathbf{e}$ is a vector with random "small" coefficients chosen from some distribution. Applebaum et al. [6] showed that the secret $\mathbf{s}$ in the LWE problem does not need to be chosen uniformly at random: the problem remains hard if $\mathbf{s}$ is chosen from the same narrow distribution as the errors $\mathbf{e}$. Based on the idea from the NTRU cryptosystem [43] of working with elements over polynomial rings rather than over the integers, and following a series of works on this topic [58], [56], [63], [71], Lyubashevsky et al. [57] showed that it is also hard to distinguish a variant of the LWE distribution from the uniform one over certain polynomial rings, thus defining the Ring-LWE assumption.

The combination of all of the above results finally led

© 2018, Joppe Bos. Under license to IEEE.
DOI 10.1109/EuroSP.2018.00032

IEEE computer society

Authorized licensed use limited to: Sookmyung Womens University. Downloaded on June 08,2026 at 04:51:56 UTC from IEEE Xplore. Restrictions apply.

메인 참고 논문

- CRYSTALS-Kyber : a CCA-secure module-lattice-based KEM

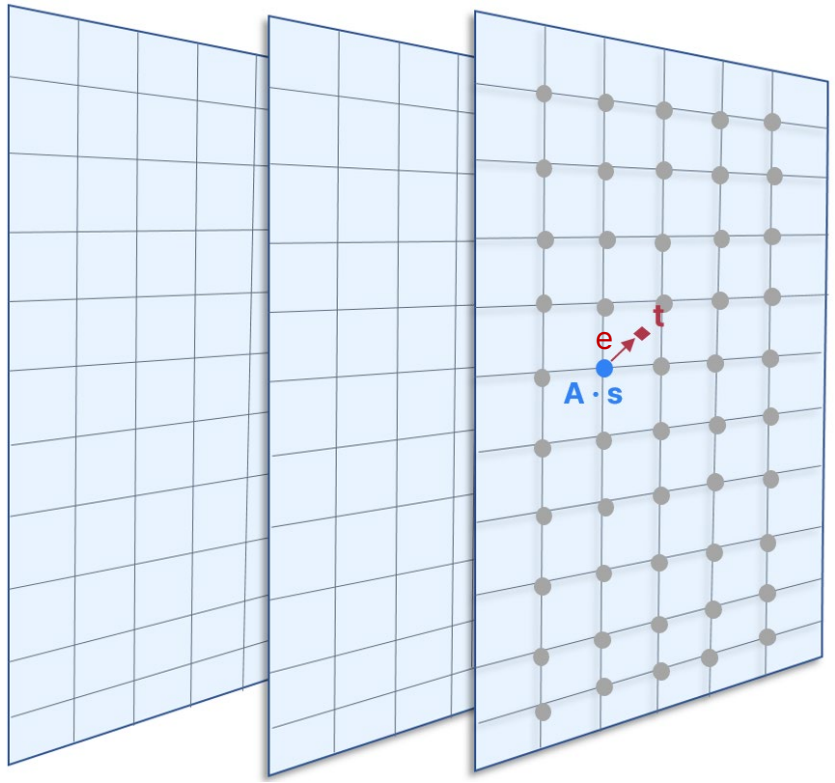
두 버전 제작

	ver1	**ver2**
ML-KEM	BouncyCastle	**논문 직접 구현 (CRYSTALS-Kyber)**
다항식 곱셈	NTT	NTT X
계산 과정	볼 수 없음	한 줄씩 관찰

ML-KEM 알고리즘

양자컴퓨터로도 풀리지 않는 격자 기반 알고리즘

$$t = A \cdot s + e$$



Kyber-768 파라미터

$n = 256$	$k = 3$	$q = 3329$	$\eta = 2$	$d_u = 10,\ d_v = 4$
다항식 계수 수	벡터 차원	계수 모듈러스	오차 범위 −2..+2	암호문 압축 비트

오차 e 가 없다면

A 와 t 를 아는 상태에서 연립방정식을 풀면 s 값을 알 수 있습니다.

오차 e 를 섞으면

가장 가까운 격자점 찾기 문제

RSA는 소인수분해로 환원되고, 쇼어 알고리즘으로 풀립니다. 격자 문제에는 그런 양자 알고리즘이 아직 없습니다.

ML-KEM 알고리즘

암호화와 복호화의 과정

공개키 $t = A \cdot s + e$

암호문 $u = A^T \cdot r + e1$

$v = t^T \cdot r + e2 + m$

받는 쪽이 $v - s^T \cdot u$ 를 계산합니다

$$
\begin{aligned}
v &= s^T \cdot A^T \cdot r + e^T \cdot r + e2 + m \\
-\, s^T \cdot u &= s^T \cdot A^T \cdot r + s^T \cdot e1 \\
\hline
&= m + (e^T \cdot r + e2 - s^T \cdot e1)
\end{aligned}
$$

결론적으로, $s^T \cdot A^T \cdot r$ 이 통째로 사라집니다

남는 것

m + 작은 오차

s, e, r, e1, e2 가 모두 −2..+2 이므로 남는 오차도 작은 값입니다.

오차 $< \frac{q}{4} = 832$

이면 항상 원래 비트를 반환합니다.

CRYSTALS-Kyber Algorithm 1. – 5.

참고 논문의 알고리즘

① Alg 1. KeyGen 키 생성

Bob 비밀 s·오차 e로 t = As + e 계산 → 공개키 pk · 개인키 sk

Algorithm 1 Kyber.CPA.KeyGen(): key generation

$\rho, \sigma \leftarrow \{0,1\}^{256}$
$\mathbf{A} \sim R_q^{k \times k} := \mathsf{Sam}(\rho)$
$(\mathbf{s}, \mathbf{e}) \sim \beta_\eta^k \times \beta_\eta^k := \mathsf{Sam}(\sigma)$
$\mathbf{t} := \mathsf{Compress}_q(\mathbf{As} + \mathbf{e}, d_t)$
return $(pk := (\mathbf{t}, \rho), sk := \mathbf{s})$

② Alg 4. Encaps 캡슐화

Alice 랜덤 m을 Enc로 감싸 c 생성 → 공유 비밀 K 도출

Algorithm 4 Kyber.Encaps($pk = (\mathbf{t}, \rho)$)

$m \leftarrow \{0,1\}^{256}$
$(\hat{K}, r) := \mathrm{G}(\mathrm{H}(pk), m)$
$(\mathbf{u}, v) := \mathsf{Kyber.CPA.Enc}\,((\mathbf{t}, \rho), m; r)$
$c := (\mathbf{u}, v)$
$K := \mathrm{H}(\hat{K}, \mathrm{H}(c))$
return (c, K)

③ Alg 5. Decaps 역캡슐화 / 위조 검사

Bob Dec로 m′ 복원 → 다시 Enc해 c와 비교
(같으면 K, 다르면 가짜 K)

Algorithm 5 Kyber.Decaps($sk = (\mathbf{s}, z, \mathbf{t}, \rho), c = (\mathbf{u}, v)$)

$m' := \mathsf{Kyber.CPA.Dec}(\mathbf{s}, (\mathbf{u}, v))$
$(\hat{K}', r') := \mathrm{G}(\mathrm{H}(pk), m')$
$(\mathbf{u}', v') := \mathsf{Kyber.CPA.Enc}\,((\mathbf{t}, \rho), m'; r')$
if $(\mathbf{u}', v') = (\mathbf{u}, v)$ **then**
  return $K := \mathrm{H}(\hat{K}', \mathrm{H}(c))$
else
  return $K := \mathrm{H}(z, \mathrm{H}(c))$
end if

↳ Alg 2. Enc 공개키 암호화

Encaps·Decaps 안에서 호출 : 공개키 t로 m을 오차 속에 숨김

Algorithm 2 Kyber.CPA.Enc($pk = (\mathbf{t}, \rho), m \in \mathcal{M}$): encryption

$r \leftarrow \{0,1\}^{256}$
$\mathbf{t} := \mathsf{Decompress}_q(\mathbf{t}, d_t)$
$\mathbf{A} \sim R_q^{k \times k} := \mathsf{Sam}(\rho)$
$(\mathbf{r}, \mathbf{e}_1, e_2) \sim \beta_\eta^k \times \beta_\eta^k \times \beta_\eta := \mathsf{Sam}(r)$
$\mathbf{u} := \mathsf{Compress}_q(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1, d_u)$
$v := \mathsf{Compress}_q\left(\mathbf{t}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v\right)$
return $c := (\mathbf{u}, v)$

↳ Alg 3. Dec 개인키 복호화

Decaps 안에서 호출 : v – sᵀu를 반올림해 m 복원

Algorithm 3 Kyber.CPA.Dec($sk = \mathbf{s}, c = (\mathbf{u}, v)$): decryption

$\mathbf{u} := \mathsf{Decompress}_q(\mathbf{u}, d_u)$
$v := \mathsf{Decompress}_q(v, d_v)$
return $\mathsf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$

읽는 순서

① → ② → ③ Bob과 Alice가 공유 비밀 K를 주고받는 흐름 (KEM)

↳ **Enc , Dec** 그 안에서 실제로 메시지를 감추고 꺼내는 함수 (PKE)

결과 두 사람이 같은 K를 얻고,

이 K로 AES-256-GCM 키를 만듭니다

실제 계산

실제 대화 시 찍힌 로그

```
D ┌─ Algorithm 5 — Decaps(sk, c)   역캡슐화  ·  상대: junghyunkim
D │ ① u := Decompress(c.u,10),  v := Decompress(c.v,4)   3.39 ms
D │     받은 암호문의 압축을 푼다 — 버린 비트는 안 돌아오지만 판정에 지장 없을 만큼만 작다  ·  u[0] = [1479, 562, 1216, 1453, 3157, 1739, 2477, 2945, …]
D │ ② v - s^T·u   4.42 ms
D │     곱셈 3회 · 내 비밀 s로만 되는 계산 — t·r 부분이 서로 지워져 '메시지 + 작은 오차'만 남는다  ·  값은 비밀이라 찍지 않음
D │ ③ m' := Compress(v - s^T·u, 1)   0.53 ms
D │     계수가 0에 가까우면 비트 0, 1665에 가까우면 비트 1 — 오차가 q/4(832) 미만이면 항상 맞는다  ·  값은 비밀이라 찍지 않음
D │ ④ (K̄', r') := G(H(pk) ‖ m')   6.25 ms
D │     보낸 쪽과 똑같은 계산 — m'이 맞다면 r'도 보낸 쪽의 r과 같은 값이 된다  ·  값은 비밀이라 찍지 않음
D │ ⑤ c' := Enc(pk, m', r')   9.56 ms
D │     복원한 m'·r'로 암호문을 처음부터 다시 만들어 본다 — 역캡슐화가 캡슐화보다 오래 걸리는 이유  ·  c'.v = [14, 3, 14, 10, 3, 13, 10, 7, …]
D │ ⑥ c == c' ?   0.48 ms
D │     일치 ✓ — 보낸 쪽이 절차대로 만든 암호문이다 (중간에 손대지 않았다는 뜻이기도 하다)
D │ ⑦ K := KDF(K̄' ‖ H(c))   0.15 ms
D │     보낸 쪽과 같은 32 B가 나온다 — 이제 이 값으로 대화를 암호화한다  ·  값은 비밀이라 찍지 않음
D └ 완료 — 실제 연산 24.86 ms  (출력은 읽히도록 천천히 내보냄)
```

실측 시간

단계	시간
키 생성	9.34 ms
캡슐화	15.28 ms
역캡슐화	24.86 ms

고려 사항

고전암호와 비교했을 때의 비용

KEM 연산은 대화 방향이 바뀔 때만 실행, 이어서 보낼 때는 해시함수만 계산되어 1ms 아래로 계산 가능

단말 연산 (에뮬레이터 실측)

상황	하는 일	시간
이어서 보냄	HKDF 2회 + AES-GCM	**< 1 ms**
방향 전환 (보내기)	키생성 + 캡슐화	**약 25 ms**
방향 전환 (받기)	역캡슐화 (재암호화 검증 포함)	**약 16 ms**

추가로 오가는 데이터

방향 전환 시 헤더	약 4.7 KB (공개키 + 캡슐, Base64)
이어서 보낼 때	**0 (본문 + 서명만)**

키 크기 비교

항목	RSA-2048	X25519	**ML-KEM-768**	**ML-DSA-65**
공개키	256B	32B	**1,568B**	**1,952B**
개인키	~1,190B	32B	**3,136B**	**4,032B**
암호문/서명	256B	32B	**2,048B**	**3,309B**
양자 안전	X	X	**O**	**O**

기존 상용 메신저 기능 비교

현재 공개된 자료를 기준으로 정리

구분	WhatsApp	Telegram 비밀대화	iMessage	Signal	Lattice Chat
1:1 종단간	기본 적용	비밀대화에서만	기본 적용	기본 적용	**기본 적용**
그룹 종단간	O	미지원	O	O	**O**
양자내성	미적용	미적용	PQ3	PQXDH	**ML-KEM-768**
구성 방식	고전	고전	**하이브리드**	**하이브리드**	PQC 단독
메타데이터	노출	노출	노출	**Sealed Sender**	노출
코드 공개	일부	클라이언트	비공개	전체	**전체**

정리

양자내성 암호를 논문에서 직접 구현하고,

앞으로 다가올 양자컴퓨터 시대에 대비한 암호 알고리즘을 적용한 메신저를 만들었습니다.

github.com/jennykim22

sequrelattice.com

[별첨] Double Ratchet

Signal 의 Double Ratchet 알고리즘 차용

🔒 🔐 Double Ratchet 세션 활성화 – 매 메시지 키 자동 교체

서버
Alice의 ML-KEM공개키(A0_pub)
Bob의 ML-KEM공개키(B0_pub)

Alice

새 키쌍 생성(A1_pub, A1_priv)
Encapsulate(B0_pub)->capsule, **공유비밀 K1**
└> 루트키1,
SendingChain1 (K1 폐기)
└> **msgKey**,
SendingChain2 (sendingChain1 폐기)
└> AES[“메시지”,**msgKey**]->**암호문** (msgKey 폐기)

✉️ 전송: [A1_pub, capule]+암호문+서명 →

Bob

Bob의 개인키(B0_priv)로 Decapsulate :
공유비밀 K1->msgKey 유도->암호문 복호화

[별첨] FO 변환

Fujisaki-Okamoto 변환 — 재암호화 검증과 묵시적 거부

m' 복원 → c' = Enc(pk, m', r') → c == c' ?

암호문 1바이트를 뒤집으면

공유 비밀의 지문	
보낸 쪽	A3 7F 2C 91 04 EE
그대로 받음	**A3 7F 2C 91 04 EE**
1바이트 변조	**5B 11 C8 40 9A 2F**

오류는 나지 않습니다. (키가 다를 뿐)

묵시적 거부

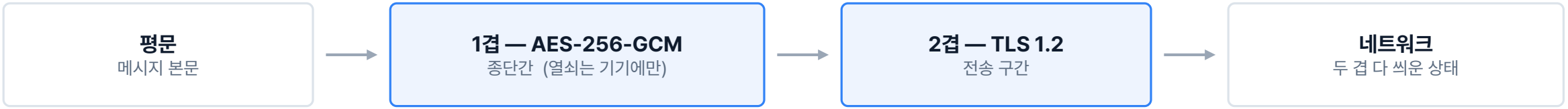
"잘못된 입력입니다"라는 안내는 공격 통로가 될 수 있습니다.

암호문을 조금씩 바꿔 보내며 그 반응만 모으면 개인키를 알아낼 수 있기 때문입니다.

그래서 개인키 안의 z 로 정답이 아닌 키를 만들어 조용히 돌려줍니다.

비교는 끝까지 합니다 : 조기 종료하면 걸린 시간만으로 몇 번째가 틀렸는지 알 수 있습니다

[별첨] 네트워크 암호화



서버(Nginx)는 2겹만 → 내부 1겹 : 종단간 암호화

TLS 적용 구간

적용 구간	무엇
HTTPS (443)	REST API — 가입 · 로그인 · 키 · 메일박스
WSS (443)	STOMP over WebSocket — 실시간 전달

인증서 발급 주체

	운영
발급	**Let's Encrypt**
확인 방식	도메인 통제권 (DV)
유효기간	90일 · 자동 갱신
신뢰 경로	시스템 CA