KpqC 4종 알고리즘 특성 분류 및 적용 방안 ver2

심민주, 송경주, 이민우, 서화정

25-07-15 **@아일랜드 리솜 리조트**

KpqC 4종 알고리즘 분석 (구현 관점)

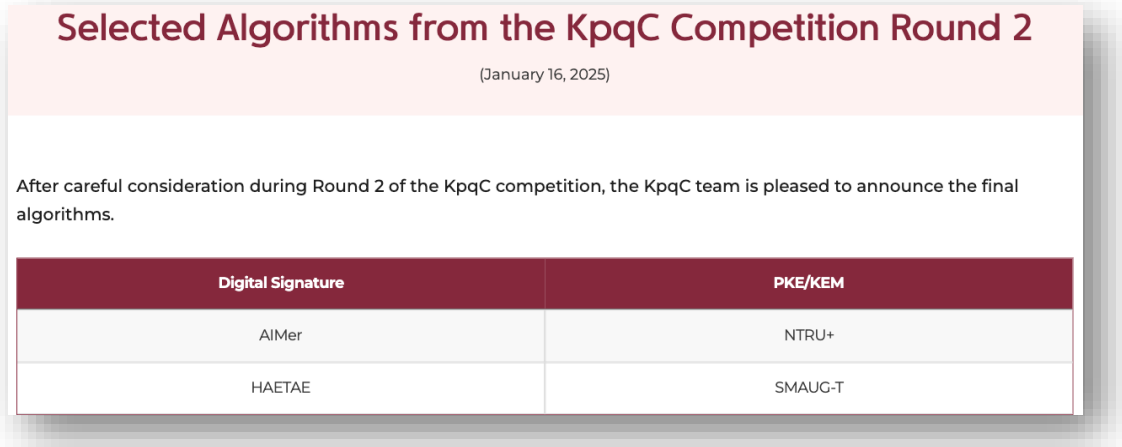
KpqC 4종 적용 프로토콜 관점 (TLS, PKI)

KpqC 4종 적용 서비스 관점 (CAN-FD)

KpqC 공모전

- **2022년 2월:** 국산 양자내성암호 공모전 개최

- **2025년 1월:** 격자기반 3종, 대칭키 기반 1종 선정



Selected Algorithms from the KpqC Competition Round 2

(January 16, 2025)

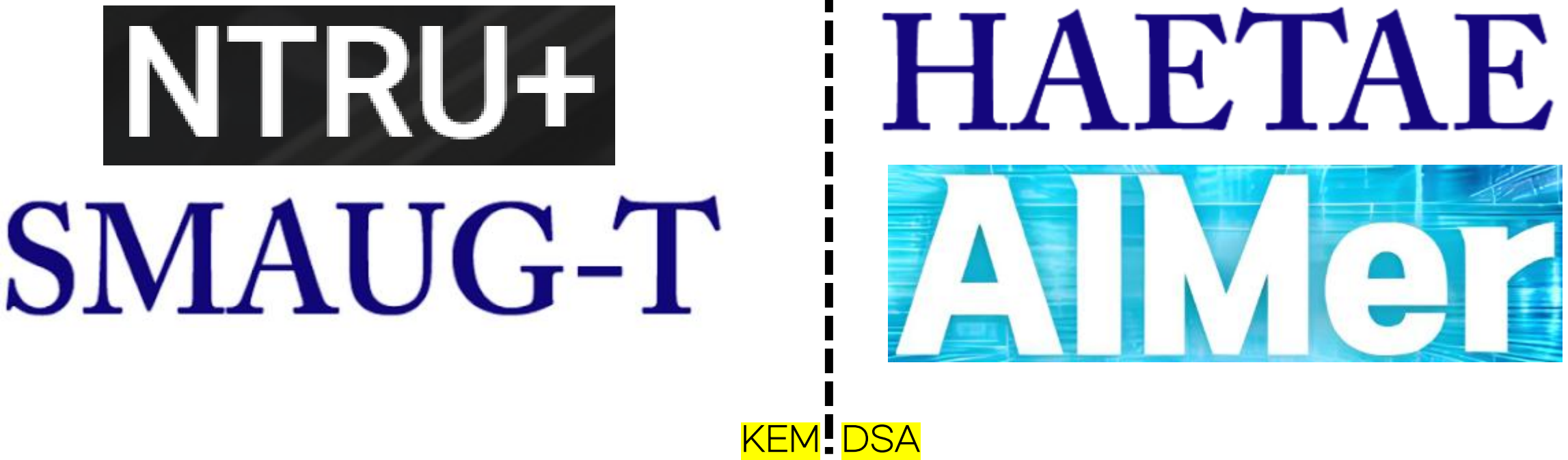
After careful consideration during Round 2 of the KpqC competition, the KpqC team is pleased to announce the final algorithms.

Digital Signature	PKE/KEM
AIMer	NTRU+
HAETAE	SMAUG-T

- **선정된 KpqC (예상) 후속 작업 (?):**
  - KpqC 알고리즘 표준화
  - 신규 알고리즘의 서비스 적용 방안 모색 (i.e., 마이그레이션) ← 금일 Talk의 주된 주제

선정된 KpqC 알고리즘 분류

- **기능: KEM 그리고 DSA**
  - 알고리즘 별 사용 용도가 명확히 분류됨



선정된 KpqC 알고리즘 분류

- **기반문제: 격자 그리고 대칭키**
  - 파라미터 크기, 계산 복잡도 → 통신 부하, 연산 속도, 메모리 사용량

NTRU+

SMAUG-T

HAETAE

AIMer

격자 | 대칭키

KpqC 알고리즘 버전 업데이트 사항

- **NTRU+**
  - 최신 업데이트 : 2025년 6월 26일 (커밋 96ff02b)
    github : https://github.com/ntruplus/ntruplus/tree/main
- **SMAUG-T**
  - 최신 코드 업데이트 : 2025년 5월 25일 (HEAD)
  - KAT 파일은 이후 업데이트되지 않음 → KAT 파일의 업데이트: 2024년 8월 21일 (커밋 c5acb07)
    github : https://github.com/hmchoe0528/SMAUG-T_public
- **AIMer**
  - 최종 업데이트 2024년 8월 1일
    github : https://github.com/samsungsds-research-papers/AIMer
- **HAETAE**
  - Round 2 제출 이후 공식 업데이트 없음

KpqCleanver2 프로젝트 최신 코드로 업데이트 완료
github : https://github.com/kpqc-cryptocraft/KpqClean_ver2

KpqC 알고리즘 별 파라미터 크기

- **파라미터 크기의 영향**
  - 네트워크 부하 (i.e., 프로토콜 상의 공개키, 암호문, 서명 전송)
  - 메모리에 대한 잦은 접근에 따른 연산 속도 저하 (e.g., register spill)
  - RAM 상에 파라미터를 적재가 어려운 경우 구현자체가 불가능

PKE/KEM (byte)

Scheme	PUBLICKEY	SECRETKEY	CIPHERTEXT
NTRU+KEM576	864	1,760	864
NTRU+KEM768	1,152	2,336	1,152
NTRU+KEM864	1,296	2,624	1,296
NTRU+KEM1152	1,728	3,488	1,728
NTRU+PKE576	864	1,760	864
NTRU+PKE768	1,152	2,336	1,152
NTRU+PKE864	1,296	2,624	1,296
NTRU+PKE1152	1,728	3,488	1,728
SMAUG-T1	672	832	672
SMAUG-T3	1,088	1,312	992
SMAUG-T5	1,440	1,792	1,376

Digital Signature (byte)

Scheme	PUBLICKEY	SECRETKEY	SIGNATURE
HAETAE2	992	1,408	1,474
HAETAE3	1,472	2,112	2,349
HAETAE5	2,080	2,752	2,948
AIMer128f	32	48	5,888
AIMer128s	32	48	4,160
AIMer192f	48	72	13,056
AIMer192s	48	72	9,120
AIMer256f	64	96	25,120
AIMer256s	64	96	17,056

KpqC 알고리즘 별 파라미터 크기

- 파라미터 크기의 영향
  - 네트워크 부하 (공개키, 암호문, 서명 전송)
  - 구현 시 메모리 접근에 따른 연산 속도 저하

Scheme	PUBLICKEY (bytes)
SMAUG-T5	1,440
NTRU+1152	1,728

Scheme	SECRETKEY (bytes)
SMAUG-T5	1,792
NTRU+1152	3,488

Scheme	CIPHERTEXT (bytes)
SMAUG-T5	1,376
NTRU+1152	1,728

Scheme	PUBLICKEY (bytes)
AIMer256	64
HAETAE5	2,080

Scheme	SECRETKEY (bytes)
AIMer256	96
HAETAE5	2,752

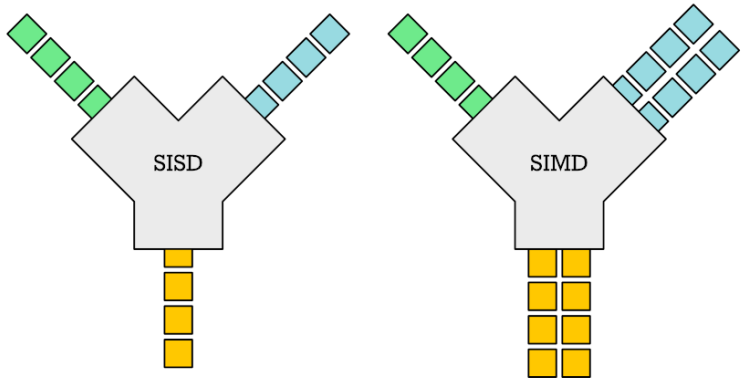
Scheme	SIGNATURE (bytes)
AIMer256s	17,056
AIMer256f	25,120
HAETAE5	2,948

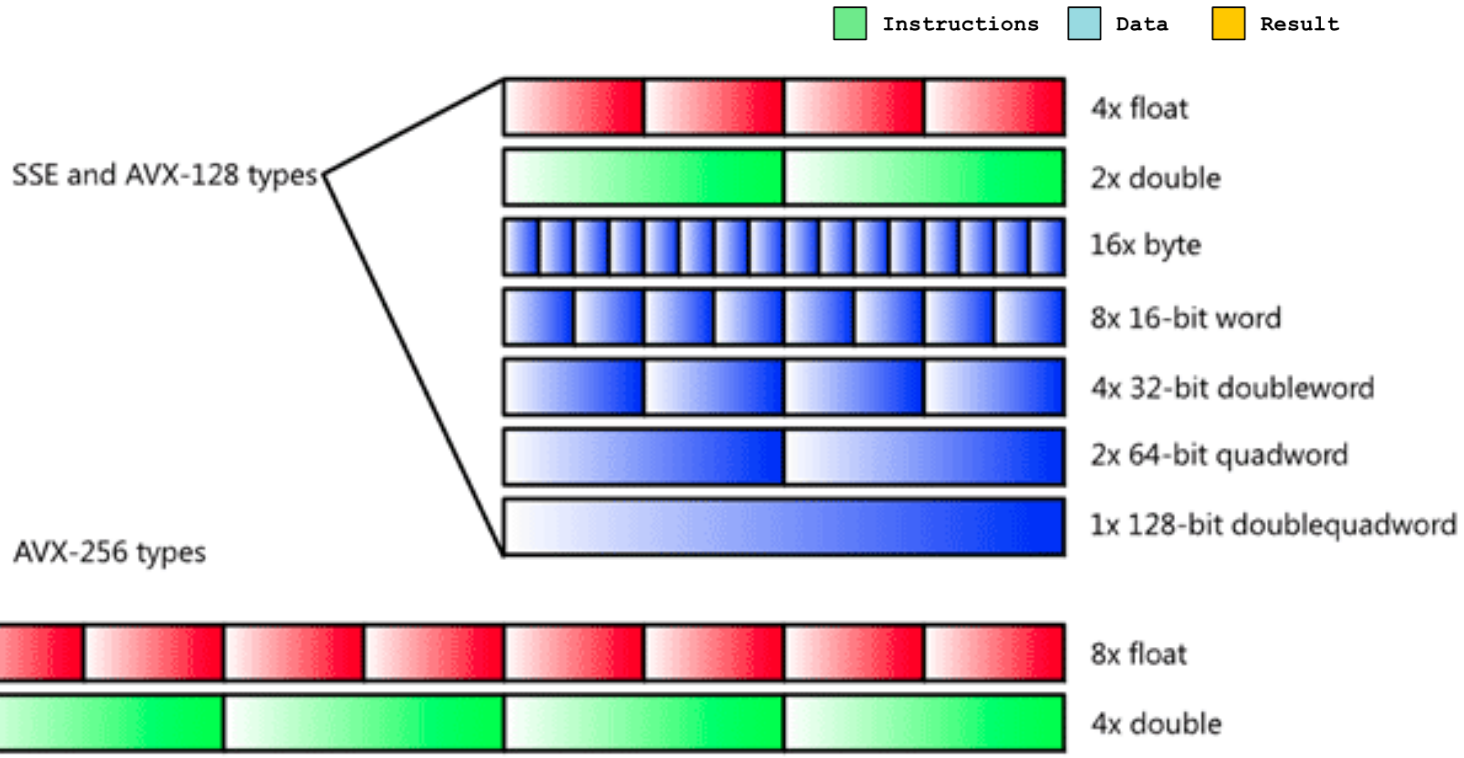
NTRU+KEM768	1,152	2,336	1,152
NTRU+PKE864	1,296	2,624	1,296
NTRU+PKE1152	1,728	3,488	1,728
SMAUG-T1	672	832	672
SMAUG-T3	1,088	1,312	992
SMAUG-T5	1,440	1,792	1,376

HAETAE2	992	1,408	1,474
HAETAE3	1,472	2,112	2,349
	2,080		
		32	
		32	
AIMer192f	48	72	13,056
AIMer192s	48	72	9,120
AIMer256f	64	96	25,120
AIMer256s	64	96	17,056

KpqC 알고리즘 성능 비교 (AVX2를 기준으로)

- **AVX2란?**
  - Intel에서 지원하는 SIMD 명령어 셋
  - 256-비트 벡터 레지스터를 지원
  - 암호화 명령어셋 지원
    (e.g., AES, GCM, SHA-3)





511 256	255 128	127 0
ZMM0	YMM0	XMM0
ZMM1	YMM1	XMM1
ZMM2	YMM2	XMM2
ZMM3	YMM3	XMM3
ZMM4	YMM4	XMM4
ZMM5	YMM5	XMM5
ZMM6	YMM6	XMM6
ZMM7	YMM7	XMM7
ZMM8	YMM8	XMM8
ZMM9	YMM9	XMM9
ZMM10	YMM10	XMM10
ZMM11	YMM11	XMM11
ZMM12	YMM12	XMM12
ZMM13	YMM13	XMM13
ZMM14	YMM14	XMM14
ZMM15	YMM15	XMM15
ZMM16	YMM16	XMM16
ZMM17	YMM17	XMM17
ZMM18	YMM18	XMM18
ZMM19	YMM19	XMM19
ZMM20	YMM20	XMM20
ZMM21	YMM21	XMM21
ZMM22	YMM22	XMM22
ZMM23	YMM23	XMM23
ZMM24	YMM24	XMM24
ZMM25	YMM25	XMM25
ZMM26	YMM26	XMM26
ZMM27	YMM27	XMM27
ZMM28	YMM28	XMM28
ZMM29	YMM29	XMM29
ZMM30	YMM30	XMM30
ZMM31	YMM31	XMM31

KpqC 알고리즘 성능 비교 (AVX2를 기준으로)

- **테스트 환경:** Ubuntu 23.10.1, i5-8259U @2.30GHz, 16GB RAM, gcc 13.2.0, -O3
- **대체적으로 DSA가 KEM보다 연산 속도가 느리고 메모리 사용량이 많음**

Scheme	Keypair(avg)
AIMer256f	182,324
HAETAE5	920,726
NTRU+PKE1152	33,554
NTRU+KEM1152	36,731
SMAUG-T5 kem	51,740
SMAUG-T5 kem 90s	53,899

Scheme	Enc/Sign(avg)
HAETAE5	4,114,258
AIMer256f	4,128,866
NTRU+PKE1152	29,510
NTRU+KEM1152	31,394
SMAUG-T5 kem	41,453
SMAUG-T5 kem 90s	41,822

Scheme	Dec/Verify(avg)
HAETAE5	138,643
AIMer256f	3,999,737
NTRU+PKE1152	18,987
NTRU+KEM1152	19,441
SMAUG-T5 kem 90s	53,680
SMAUG-T5 kem	53,919

Algorithm	Stack+Heap (bytes)	Stack (bytes)	Heap (bytes)
AIMer-256f	650,592	650,584	8
HAETAE-V	235,808	235,800	8

Algorithm	Stack+Heap (bytes)	Stack (bytes)	Heap (bytes)
NTRU+1152	25,376	25,368	8
SMAUG-T5	40,448	40,440	8

KpqC 암호화 알고리즘에 대한 프로파일링

Algorithm	버전	AVX2 Enable
NTRU+KEM576	개발자 코드	O
SMAUG-T1	개발자 코드	O
AIMer128f	개발자 코드	O
HAETAE5	개발자 코드→KPQClean(ver2)	O

- gprof 툴을 활용하여 프로파일링 수행
- 전체 수행, 키생성, 서명/암호화, 검증/복호화로 세분화하여 프로파일링 수행

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
 31.25      0.05     0.05                             __KeccakF1600
 12.50      0.07     0.02                             KeccakP1600_AddBytes_LaneAlignedLoop
 12.50      0.09     0.02                             loop3
  6.25      0.10     0.01                             KeccakP1600_Permute_24rounds
  6.25      0.11     0.01                             _loop_inner
  6.25      0.12     0.01                             _loop_poly_crepmod3
  6.25      0.13     0.01                             _looptop_poly_tobytes
  6.25      0.14     0.01                             _looptop_start_345
  6.25      0.15     0.01                             loop1
  6.25      0.16     0.01                             loop2
```

```
readme:1:grep -rnw . -e 'loop3'
grep: test_enc: binary file matches
grep: test_key: binary file matches
asm/sha256.S:517:loop3:
asm/sha256.S:522:        jb      loop3
all.txt:8: 12.50      0.09     0.02                              loop3
grep: test_dec: binary file matches
key.txt:65:[3]     33.3    0.01    0.00                  loop3 [3]
key.txt:178:    [2] TEST_CCA_KEM_CLOCK      [6] fips202avx_shake256      [3] loop3
```

함수명과 실제 연산을 매칭시켜서 연산부하 추정

상위 3개의 함수에 대해서만 랭킹 수행

KpqC 암호화 알고리즘에 대한 프로파일링 (KEM)

- **NTRU+**

RANK	ALL		KEY		ENC		DEC	
1	SHA3	31.25%	etc	33.33%	SHA3	40.00%	SHA3	50.00%
2	SHA3	12.50%	SHA3	33.33%	SHA3	20.00%	SHA3	25.00%
3	SHA2	12.50%	SHA2	33.33%	NTT	20.00%	MUL	25.00%

SHA3가 KpqC 암호화 알고리즘에서 큰 부하를 차지하고 있음

- **SMAUG-T**

RANK	ALL		KEY		ENC		DEC	
1	SHA3	36.00%	SHA3	62.50%	SHA3	28.57%	SHA3	66.67%
2	NTT	24.00%	SHA3	12.50%	SHA3	28.57%	SHA3	11.11%
3	etc	-	Sampling	12.50%	NTT	28.57%	SHA3	11.11%

KpqC 암호화 알고리즘에 대한 프로파일링 (DSA)

- **AIMer**

RANK	ALL		KEY		SIGN		VERIFY	
1	SHA3	42.65%	SHA3	40.00%	SHA3	40.00%	SHA3	35.29%
2	SHA3	26.47%	AES	20.00%	SHA3	17.14%	SHA3	35.29%
3	MUL	10.29%	AES	20.00%	MUL	11.43%	MUL	14.71%

- **HAETAE: 다른 알고리즘과 달리 FFT 연산 부하가 높음**

RANK	ALL		KEY		SIGN		VERIFY	
1	FFT	50.77%	FFT	82.98%	Sampling	37.93%	Sampling	33.33%
2	SHA3	13.85%	Sampling	7.45%	SHA3	34.48%	SHA3	33.33%
3	Sampling	13.08%	FFT	5.32%	etc	-	SHA3	16.67%

주요 공통 구현 모듈 분석

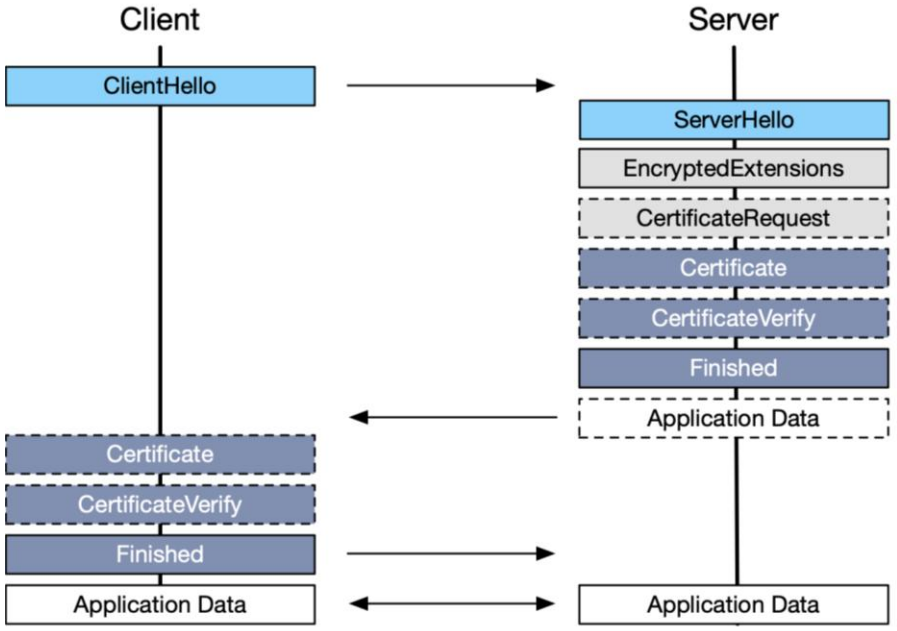
- 해시 모듈은 PQClean KeccakP1600x4 공통 사용
  - 프로파일링 상에서 확인한 바와 같이 SHA-3 구현 최적화가 중요

- 곱셈 및 리덕션의 경우 각 기반문제에서 적합한 기법들이 적용
- 루프 언롤링 및 mask 기반 분기 제거 기법은 4종 모두 적용

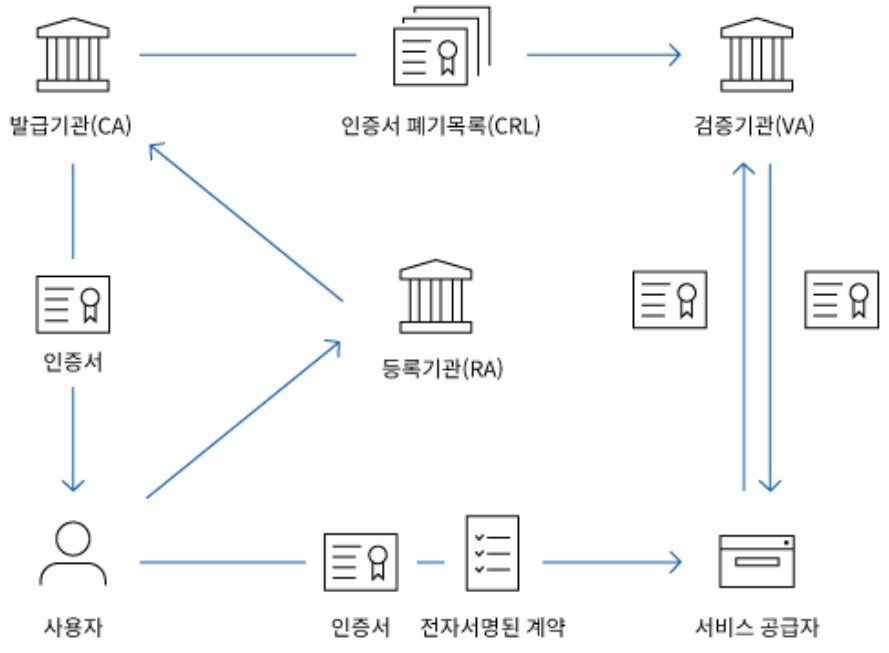
최적화 대상	AIMer128	HAETAE2	SMAUG-T1	NTRU+576
해시	4-way Keccak			
NTT	X	O		
리덕션	Fast Reduction	Montgomery & Barrett	Shift based	Montgomery

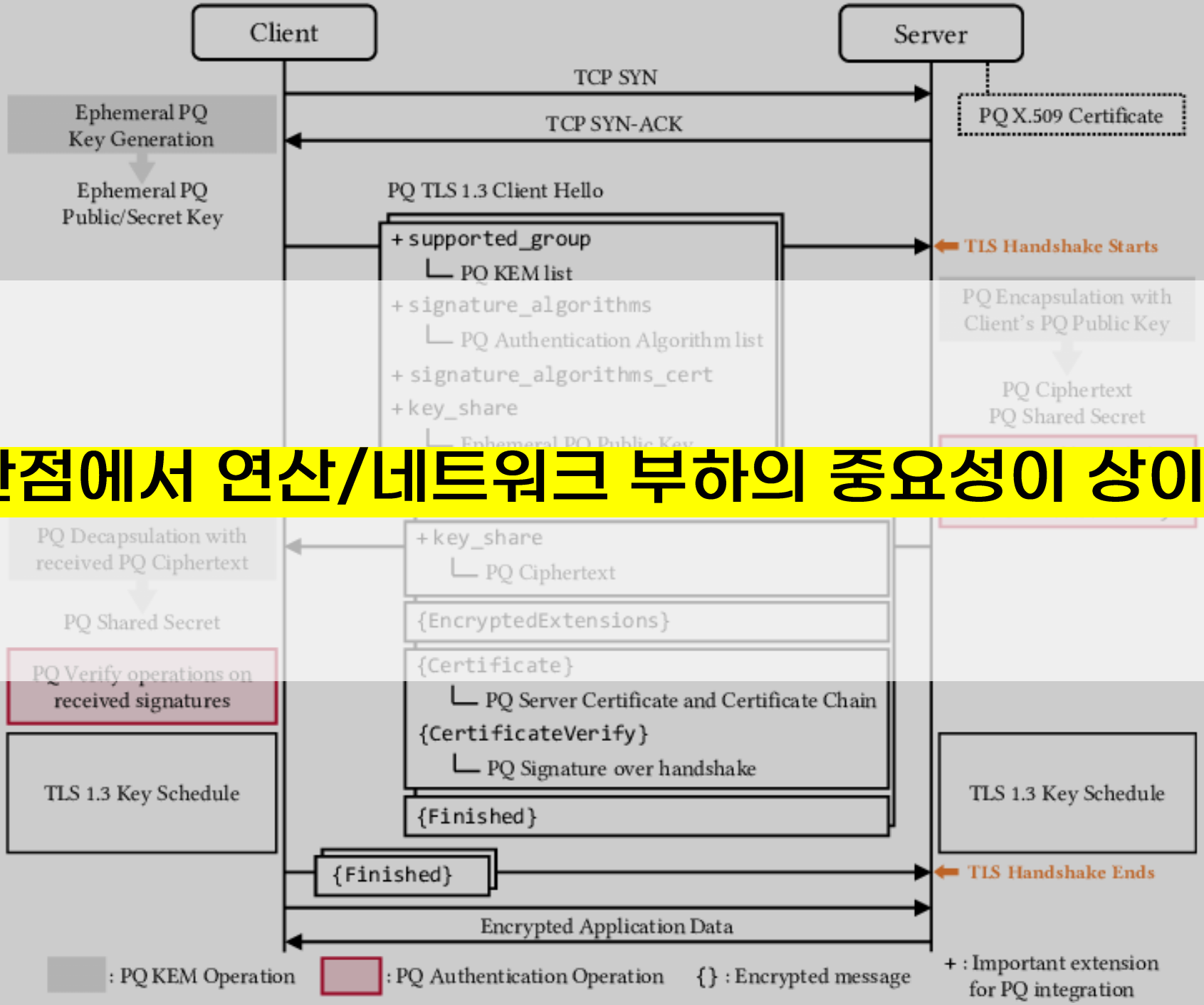
KpqC (혹은 공개키 암호)가 가장 많이 활용될 프로토콜

- TLS
  - TLS 1.3: 2018년 표준화된 최신 암호화 프로토콜, TCP 위에서 동작
  - 클라이언트-서버 간 인증, 암호화, 무결성 제공
  - 보안 강화 및 지연 시간 감소

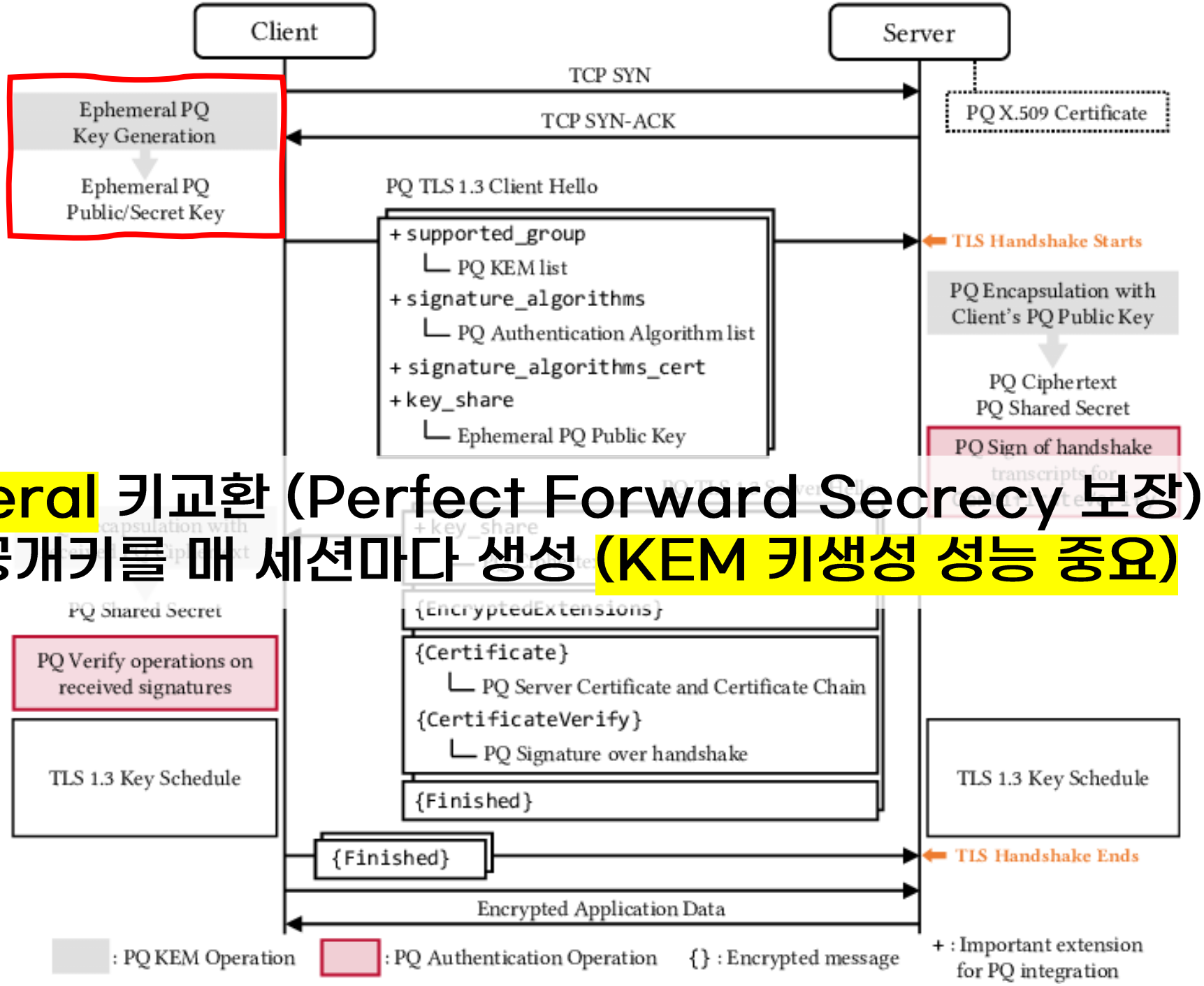


- PKI
  - PKI는 공개키 암호와 디지털 인증서를 기반으로 신원 인증과 안전한 통신을 제공하는 인프라
  - 웹 트래픽 보호, 이메일·코드 서명 등 다양한 보안 서비스에 활용됨



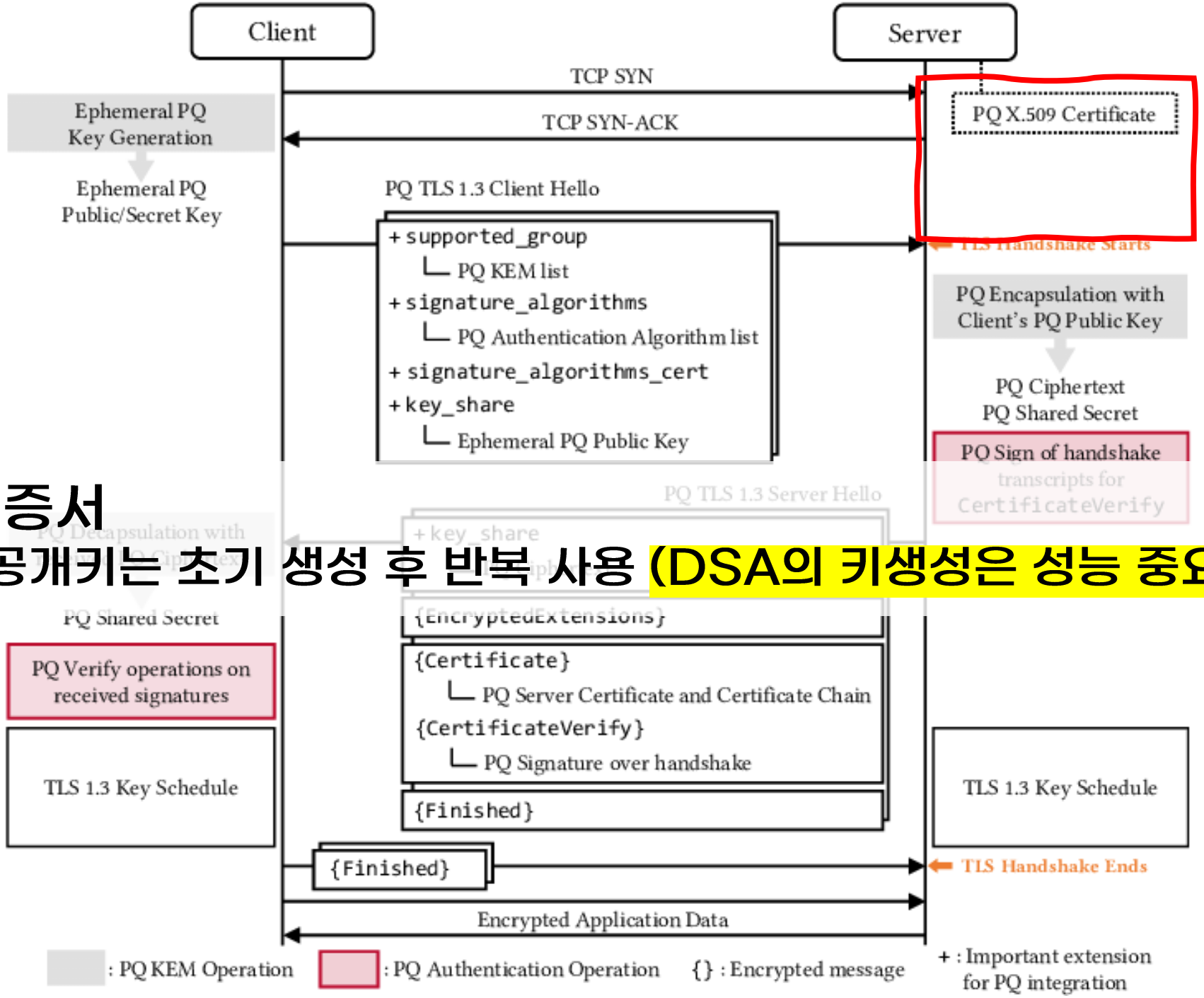


TLS 관점에서 연산/네트워크 부하의 중요성이 상이하게 나타남



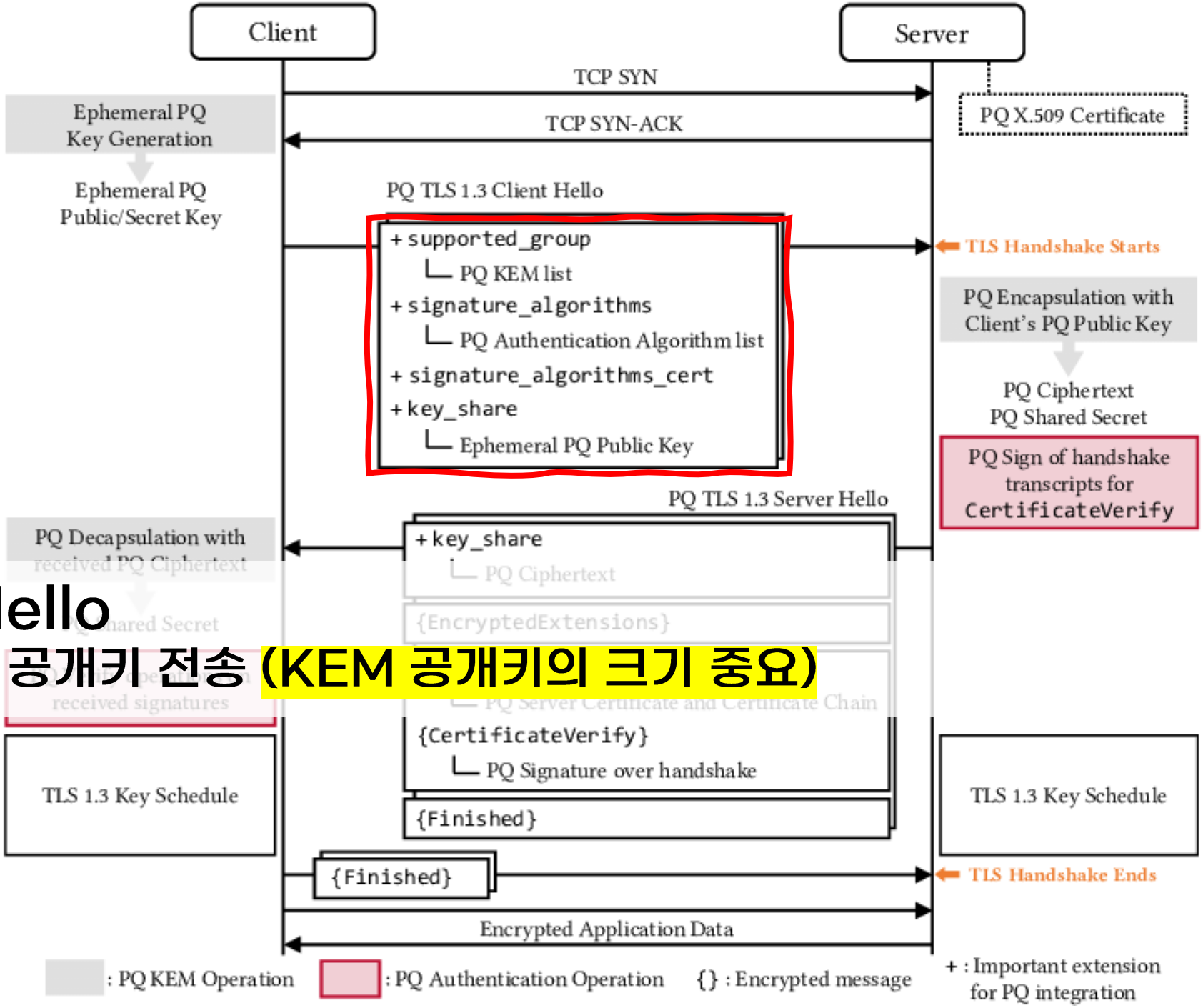
Ephemeral 키교환 (Perfect Forward Secrecy 보장)

- 임시 공개키를 매 세션마다 생성 (KEM 키생성 성능 중요)



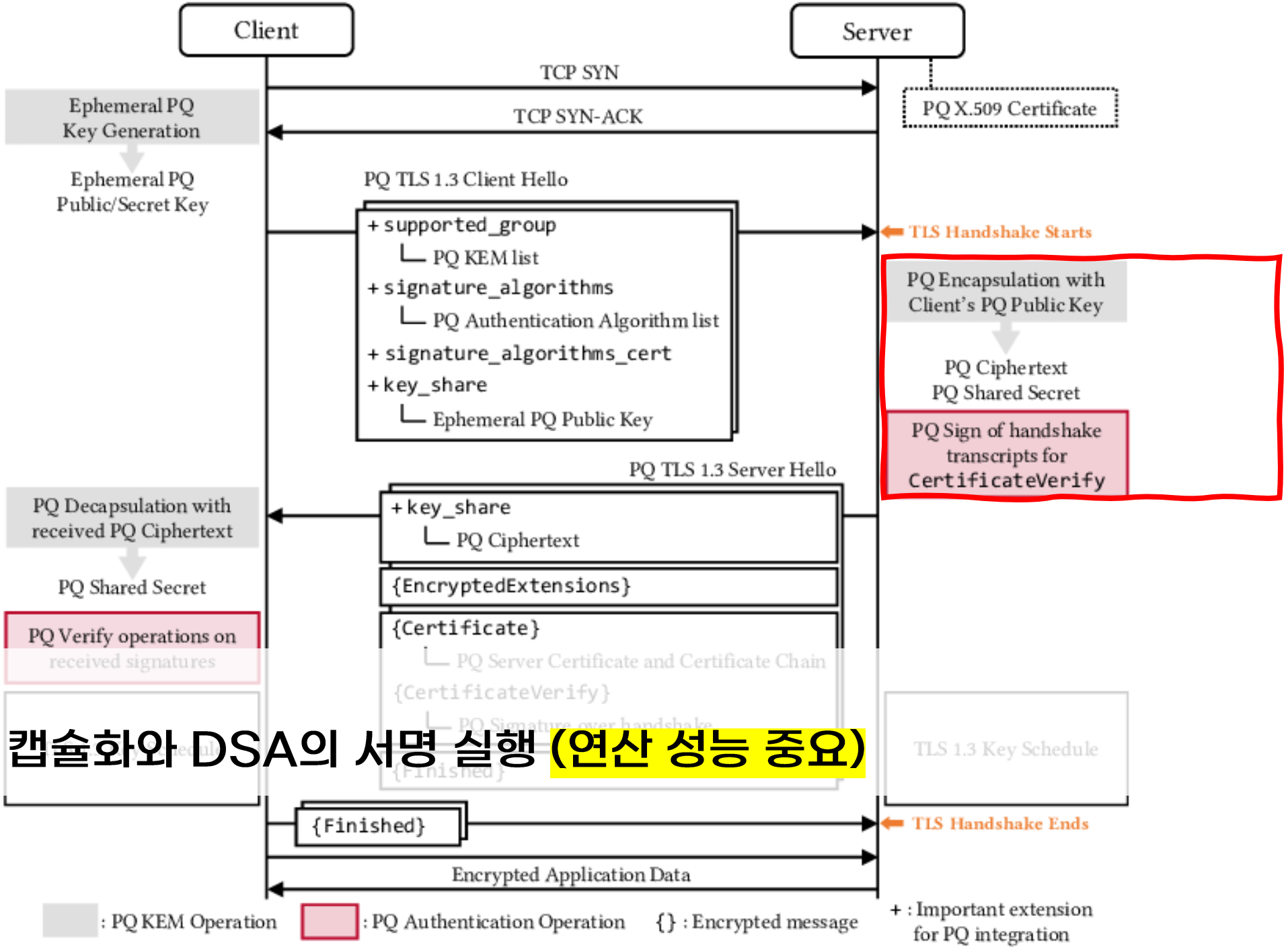
X.509 인증서

- 서명용 공개키는 초기 생성 후 반복 사용 (DSA의 키생성은 성능 중요도 낮음)



Client Hello

- KEM의 공개키 전송 (KEM 공개키의 크기 중요)

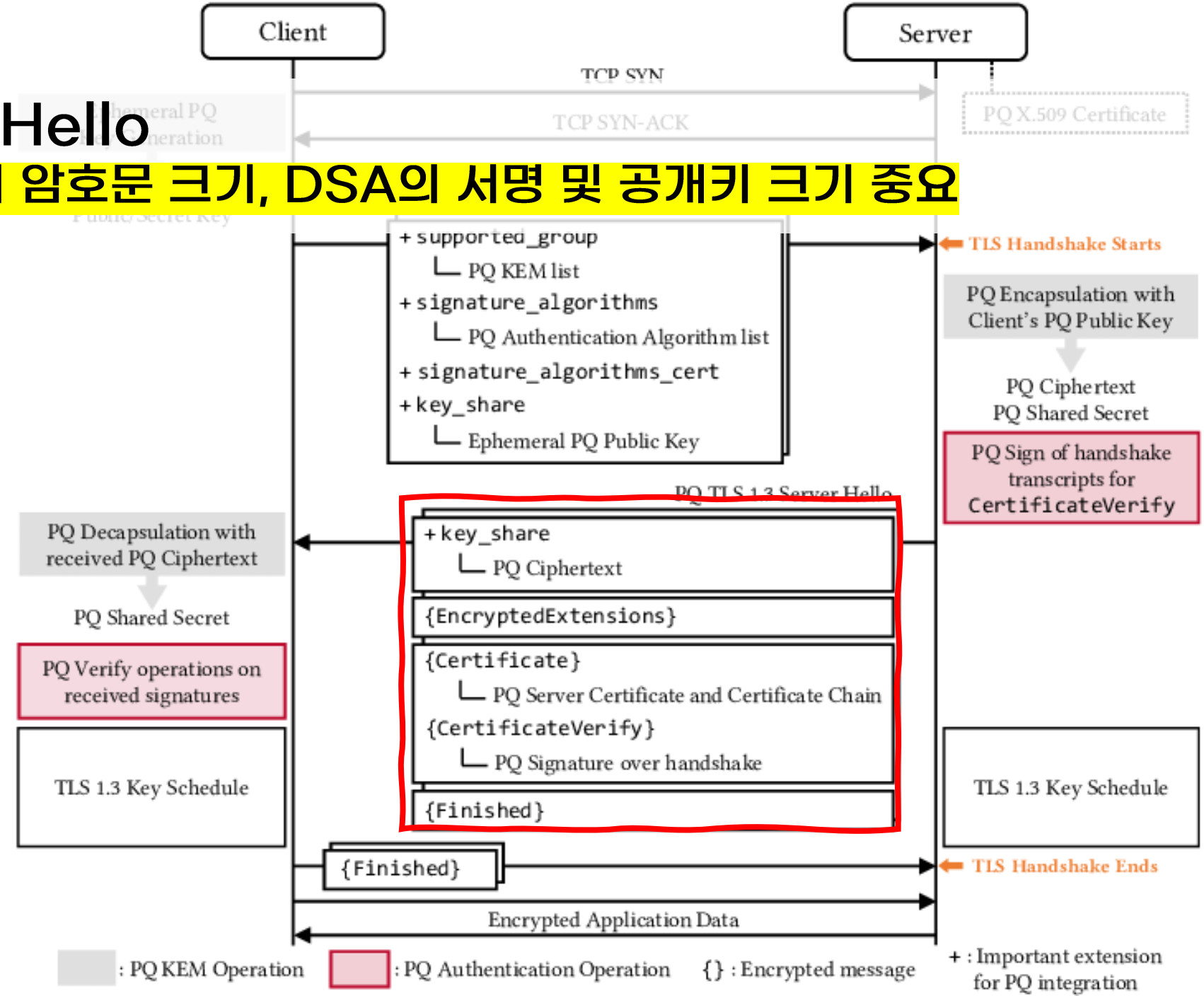


서버연산

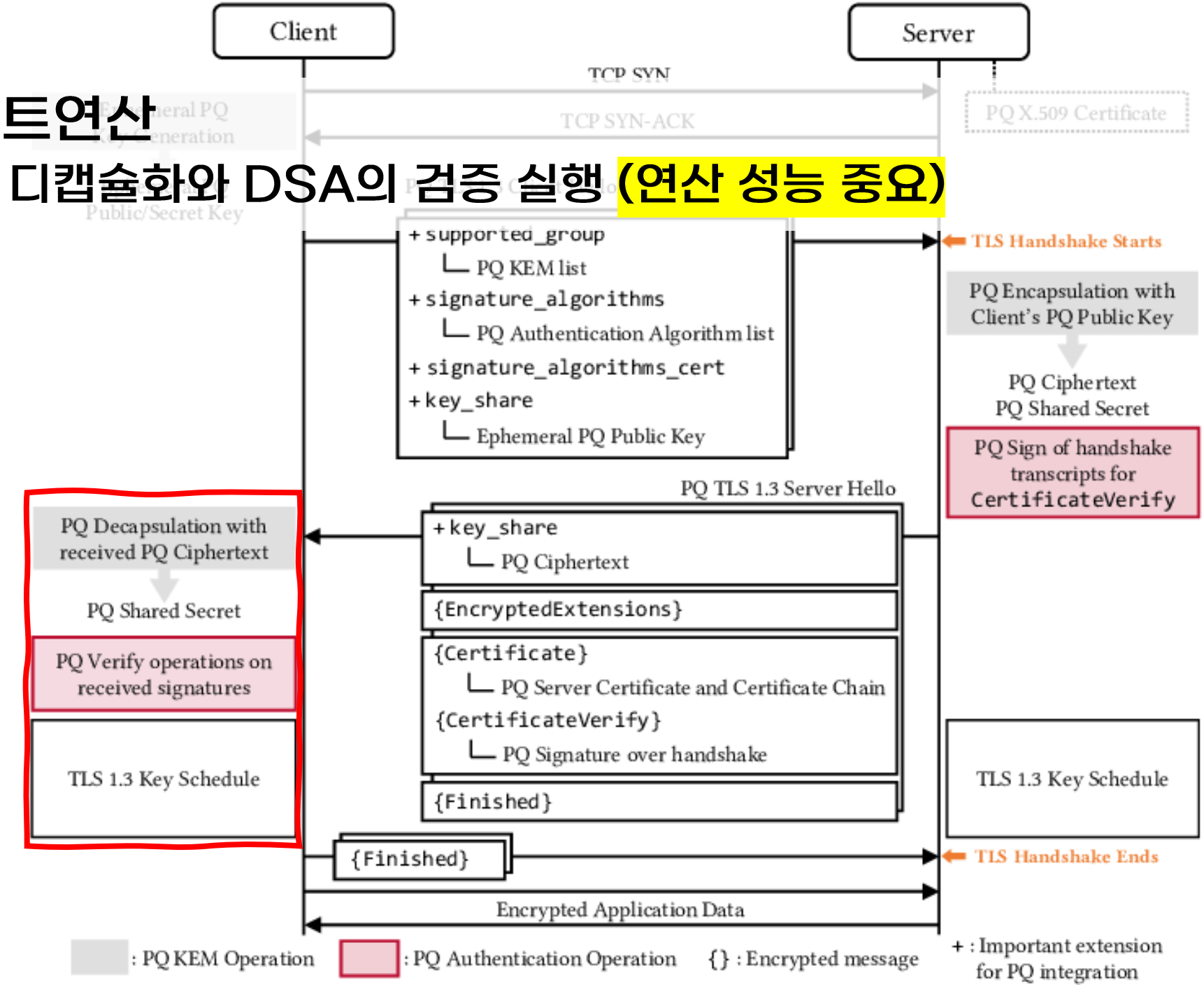
- KEM의 캡슐화와 DSA의 서명 실행 **(연산 성능 중요)**

Server Hello

- KEM의 암호문 크기, DSA의 서명 및 공개키 크기 중요



클라이언트연산
- KEM의 디캡슐화와 DSA의 검증 실행 **(연산 성능 중요)**



TLS 관점에서 PQC 최적화 중요도

클라이언트	서버
KEM 키생성 KEM 디캡슐화 DSA 검증	(오프라인) DSA 키생성 KEM 캡슐화 DSA 서명

일반적으로 클라이언트가 성능이 낮은 만큼 클라이언트 쪽의 PQC연산이 효율적일수록 TLS 적용에 유리
다만 서버 쪽에서의 대용량 처리를 효율적으로 해야 하기 때문에 응용에 따라 다르게 해석 가능

- **네트워크 패킷 교환 관점 (작을수록 좋음)**
  - KEM 공개키
  - KEM 암호문
  - DSA 서명
  - DSA 공개키

TLS 관점에서 PQC 최적화 중요도

클라이언트	서버
KEM 키생성 KEM 디캡슐화 DSA 검증	(오프라인) DSA 키생성 KEM 캡슐화 DSA 서명

연산속도 중요 서비스 (컴퓨팅 파워 제약 사항)

- 네트워크 패킷 교환 관점 (작을수록 좋음)
  - KEM 공개키
  - KEM 암호문
  - DSA 서명
  - DSA 공개키

전송속도 중요 서비스 (통신회선 제약 사항)

TLS 관점에서 PQC 최적화 중요도

클라이언트	서버
KEM 키생성 KEM 디캡슐화 DSA 검증	(오프라인) DSA 키생성 KEM 캡슐화 DSA 서명

KEM 알고리즘	키생성	디캡슐화	총합
SMAUG-T5	51,740	53,919	105,659
NTRU+KEM1152	36,731	19,441	56,172

KEM 알고리즘	캡슐화	총합
SMAUG-T5	41,453	41,453
NTRU+KEM1152	31,394	31,394

DSA 알고리즘	검증	총합
HAETAE5	138,643	138,643
AIMer256f	3,999,737	3,999,737

DSA 알고리즘	서명	총합
HAETAE5	4,114,258	4,114,258
AIMer256f	4,128,866	4,128,866

TLS 실제 네트워크 전송 시 총 크기

계층	크기	비고 / 옵션
Ethernet Header	14 bytes	VLAN 태그 시 +4 bytes
IP Header	20 bytes (IPv4) 40 bytes (IPv6)	IPv4 옵션 포함 시 최대 60 bytes IPv6 확장 헤더 추가 기능
TCP Header	20 ~ 60 bytes	기본 20 bytes, 옵션 포함 시 확장
TLS Record Header	5-byte	Content Type(1) + Version(2) + Length(2)
TLS Record Data	순수 애플리케이션 데이터 크기: 최대 16,384 bytes (2^{14}) 암호화된 레코드 청크 제한: 최대 16,640-byte(2^{14}+256)	**Plaintext limit**: RFC 8446 Section 5.1 **Ciphertext limit**: RFC 8446 Section 5.2 암호화 확장: 최대 256 bytes (AEAD)

<실제 전송 시 총 패킷 크기>

Ethernet + IP + TCP + TLS 헤더 + TLS 데이터 = 최대 16,699 bytes

(IPv4 기준: 14+20+20+5+16,640)

*TLS Record Data는 암호화된 레코드 기준(16,640 bytes)로 계산

- **실제 제약 사항**
  - 이더넷 MTU (Maximum Transmission Unit): 1,500 bytes (표준)
  - 실제 TCP 페이로드: ~1,460 bytes
  - 16KB TLS 레코드: 약 11~12개 TCP 세그먼트로 분할
  - 분할로 인한 지연: 모든 세그먼트 수신 후 복호화

https://datatracker.ietf.org/doc/html/rfc8446

TLS 패킷 비교(KEM)

항목	TLS 레코드 크기	IP 페이로드 한도
의미	TLS가 처리하는 데이터 블록 크기	네트워크 패킷 내 전송 가능한 데이터 최대치
최대 크기	16,384 ~ 16,640 bytes	일반적으로 1,460 bytes (이더넷 기준)
관계	TLS 레코드가 전송될 때 분할됨	TLS 레코드 > IP 페이로드 → 분할 필요

- 모든 암호문 크기가 16,640 bytes를 넘지 않으므로, TLS 1.3 레코드 하나에 담아 전송 가능

알고리즘	보안레벨	공개키 PK (B)	암호문 CT (B)	TLS 패킷 크기 (B) (PK / CT)	패킷 수 (PK / CT)
MLKEM512	1	800	768	875 / 843	1/1
HQC-128	1	2,249	4,433	2,324 / 4,508	2/4
SMAUG-T1	1	672	672	747 / 747	1/1
NTRU+KEM576	1	864	864	939 / 939	1/1
MLKEM768	3	1,184	1,088	1,259 / 1,163	1/1
HQC-192	3	4,522	8,978	4,597 / 9,053	4/7
SMAUG-T3	3	1,088	992	1,163 / 1,067	1/1
NTRU+KEM768	3	1,152	1,152	1,227 / 1,227	1/1
NTRU+KEM864	3	1,296	1,296	1,371 / 1,371	1/1
MLKEM1024	5	1,568	1,568	1,643 / 1,643	2/2
HQC-256	5	7,245	14,421	7,320 / 14,496	6/11
SMAUG-T5	5	1,440	1,376	1,515 / 1,451	2/2
NTRU+KEM1152	5	1,728	1,728	1,803 / 1,803	2/2

TLS 패킷 비교(DSA)

일부 알고리즘은 서명 크기가 TLS 레코드 최대 크기인 16,640 bytes 초과
→ 레코드 분할 필요

알고리즘	보안레벨	공개키 PK (B)	서명 Sig (B)	TLS 패킷 크기 (B) (PK / Sig)	패킷 수 (PK / Sig)
Falcon-512	1	897	666	972 / 741	1 / 1
SPHINCS+128s	1	32	7,856	107 / 7,931	1 / 6
SPHINCS+128f	1	32	17,088	107 / 17,163	1 / 13
AIMer128s	1	32	4,160	107/4,235	1 / 4
AIMer128f	1	32	5,888	107/5,963	1 / 5
MLDSA44	2	1,312	2,420	1,387 / 2,495	1 / 2
HAETAE2	2	992	1,474	1,067/1,549	1 / 2
MLDSA65	3	1,952	3,293	2,027 / 3,368	2 / 3
HAETAE3	3	1,472	2,349	1,547 / 2,424	2 / 2
AIMer192s	3	48	9,120	123 / 9,195	1 / 7
AIMer192f	3	48	13,056	123 / 13,131	1 / 10
SPHINCS+192s	3	48	16,224	123 / 16,299	1 / 12
SPHINCS+192f	3	48	35,664	123 / 35,739	1 / 26
MLDSA87	5	2,592	4,595	2,667 / 4,670	2 / 4
HAETAE5	5	2,080	2,948	2,155 / 3,023	2 / 3
Falcon-1024	5	1,793	1,280	1,868 / 1,355	2 / 1
SPHINCS+256s	5	64	29,792	139 / 29,867	1 / 22
SPHINCS+256f	5	64	49,856	139 / 49,931	1 / 36
AIMer256s	5	64	17,056	139 / 17,131	1 / 13
AIMer256f	5	64	25,120	139 / 25,195	1 / 18

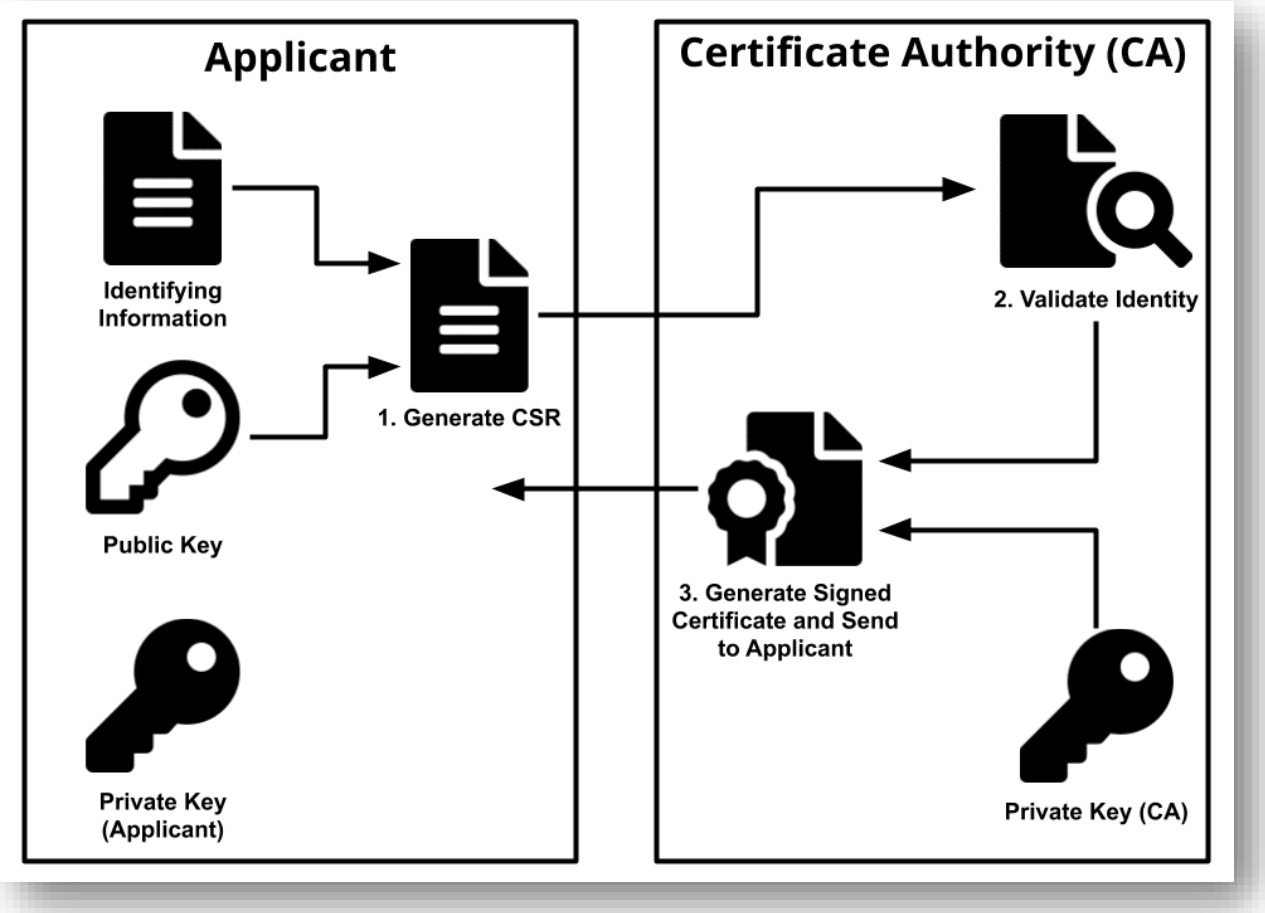
PQC TLS Handshake 알고리즘 조합

보안레벨	KEM	DSA	패킷 수 (CT/SIG)
L1	MLKEM-512	FALCON-512	1/1
	MLKEM-512	AIMer-128s	1/4
	SMAUG-T1 / NTRU+KEM576	FALCON-512	1/1
	SMAUG-T1 / NTRU+KEM576	AIMer-128s	1/4
L3	MLKEM-768	MLDSA-65	1/3
	MLKEM-768	HAETAE-3	1/2
	SMAUG-T3 / NTRU+KEM768	MLDSA-65	1/3
	SMAUG-T3 / NTRU+KEM768	HAETAE-3	1/2
L5	MLKEM-1024	MLDSA-87	2/4
	MLKEM-1024	HAETAE-5	2/3
	SMAUG-T5 / NTRU+KEM1152	MLDSA-87	2/4
	SMAUG-T5 / NTRU+KEM1152	HAETAE-5	2/3

TLS 1.3에 KpqC 적용 어려움이 없을 것으로 보임
TLS 1.3을 사용하는 산업 분야에 KpqC 적용 가능

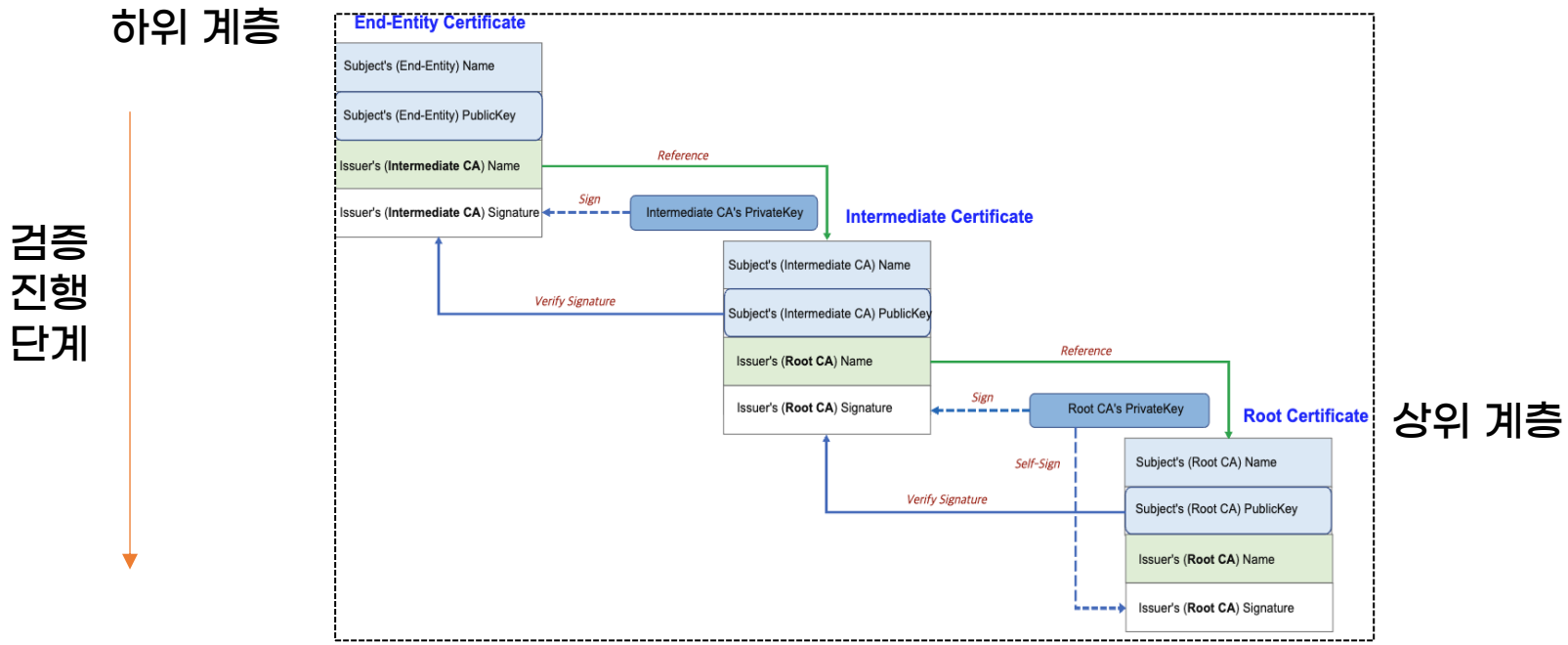
PKI 상세

- **CA (Certificate Authority) 역할**
  - 사용자는 CA의 공개키로 서명을 검증함으로써 해당 공개키가 신뢰할 수 있음을 확인하고 안전한 통신수행



PKI 상세

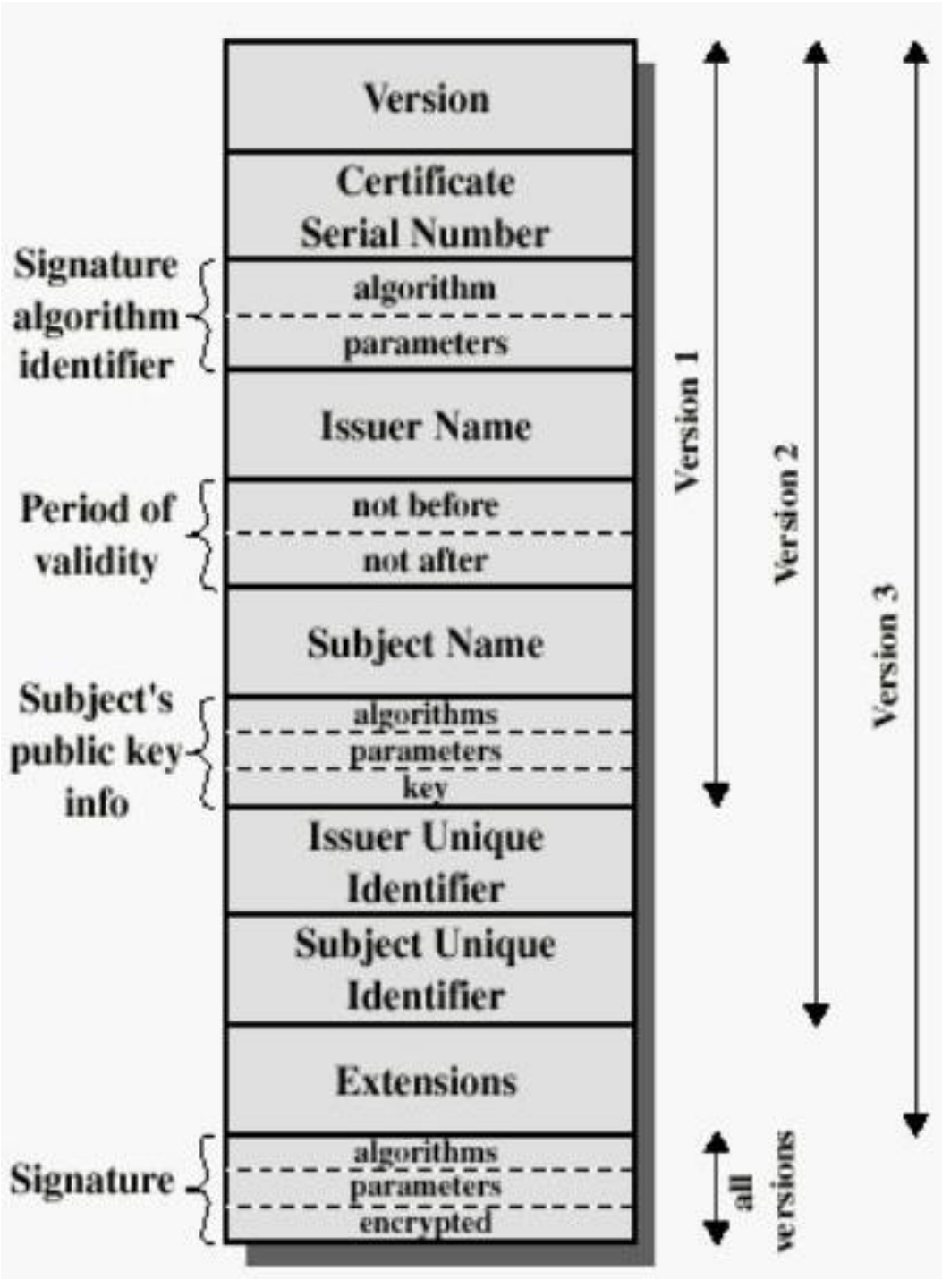
- **Chain of Trust**
  - PKI는 루트 인증서까지 상위 인증기관의 서명을 단계적으로 검증하는 Chain of Trust 구조 사용
  - Root CA (최상위 인증기관) : 자체 서명된 인증서를 탑재
  - Intermediate CA: Root CA로부터 서명을 받아 End-Entity 인증서를 발급/관리



Chain of trust

- **X.509**
  - 디지털 인증서 형식에 대한 국제 표준 (ITU-T X.509)
  - SSL/TLS, 전자서명, 이메일 암호화에서 사용
  - 버전, 시리얼 번호, 서명 알고리즘, 발행자, 유효기간 포함
  - 서명 값(Signature Value)은 CA가 인증서 데이터에 대해 개인키로 서명한 결과로 인증서의 신뢰성을 검증하는 핵심 요소

- **현 X.509의 문제점**
  - X.509 인증서를 PQC 기반으로 전환 필요
    - 호환성 문제로 단독 사용은 어려움
  - 이를 해결하기 위해 Legacy와 PQC 알고리즘을 함께 포함하는 하이브리드 인증서가 제안



X.509 하이브리드 인증서 (1 / 3)

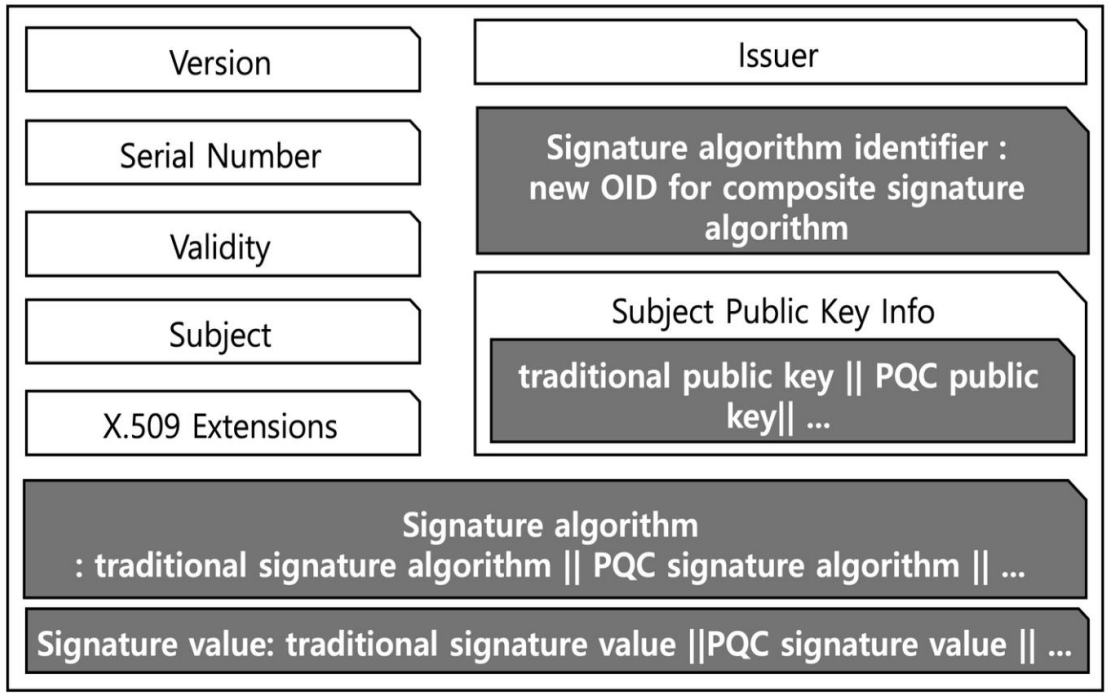
- Composite 인증서
  - 하나의 X.509 인증서 안에 둘 이상의 공개키 정보(RSA/ECC + PQC)와 복합 디지털 서명을 포함하는 방식

  - 장점
    - **보안성 극대화:** 하나라도 안전하면 인증서 위변조 방지
    - **단일 객체 관리:** 하나의 인증서로 이중 알고리즘 운용

  - 단점
    - **호환성 없음:** 기존 클라이언트에서 미지원 → **업데이트 필요**
    - **인증서 대형화:** 키·서명 중복으로 **크기 증가**
    - **구현 어려움:** 모든 구성서명 검증 등 **검증논리가 복잡**

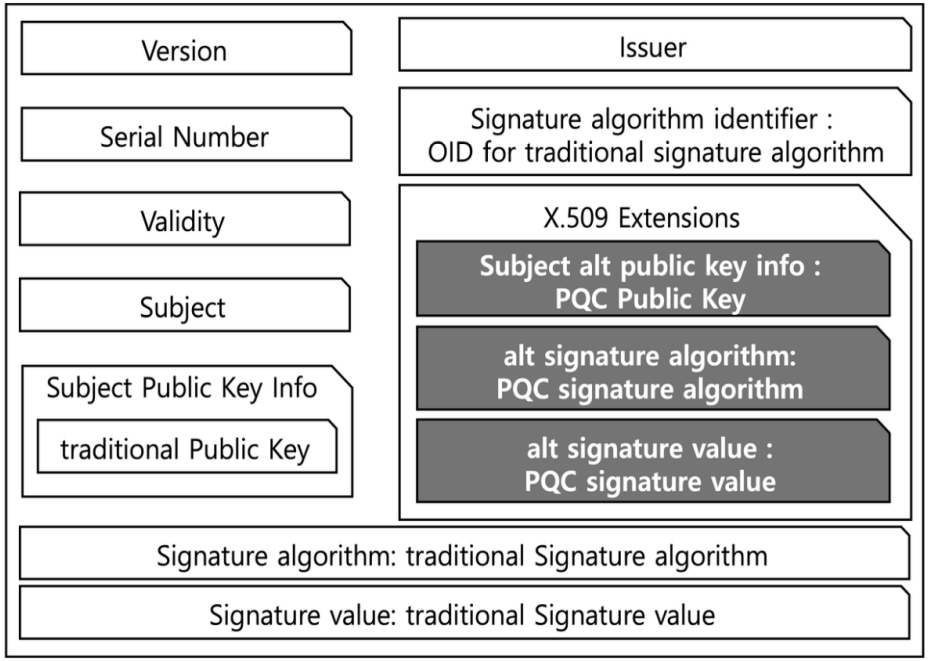
  - **IETF 초안 진행:** Composite Signature/KEM 초안 채택

  - **시범 구현:** OQS 등 OpenSSL 포크에서 테스트



X.509 하이브리드 인증서 (2 / 3)

- Hybrid 인증서
  - 기존 인증서 필드엔 Legacy 알고리즘 사용
  - v3 Extensions Field에 PQC 공개키·서명 알고리즘·서명값 저장
  - 인증서 자체는 표준 X.509 구조 유지



  - 장점
    - **완벽 호환:** 구형 시스템은 확장 무시 → 기존 방식으로 동작
    - **점진적 이행:** 하나의 인증서로 PQC 지원/미지원 모두 대응
    - **이중서명 안전:** 두 서명 중 하나가 안전하면 신뢰 유지
  - 단점
    - **구현 복잡:** 확장 필드 처리 및 이중 서명 검증 로직 추가 필요
    - **경로검증 수정:** PQC 검증 위해 PKI 소프트웨어 업그레이드 요구
    - **인증서 크기 증가:** 두 개 키/서명으로 인증서 용량 증가
  - **제품 구현:** EJBCA 등에서 지원 시작

X.509 하이브리드 인증서 (3 / 3)

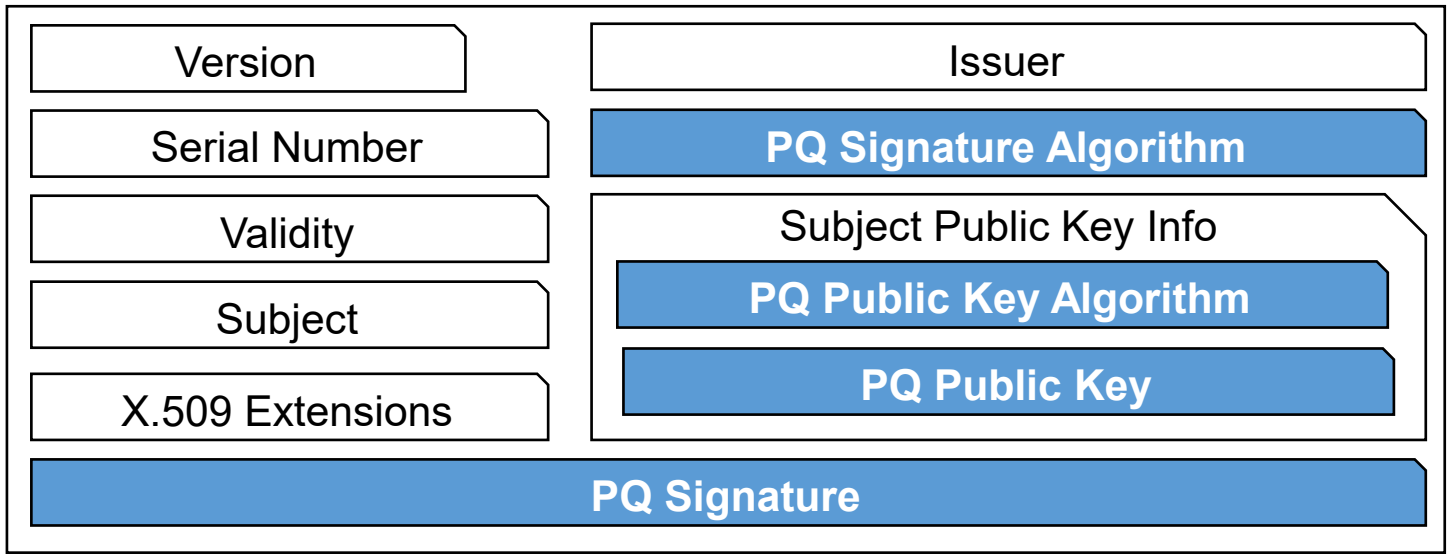
- **Chameleon 인증서**
  - **기본 인증서**(Base Certificate)와 **델타 인증서**(Delta Certificate) 한 쌍으로 구성
  - 기본 인증서는 기존의 RSA/ECDSA 등의 **전통적인 서명 알고리즘**으로 서명된 인증서
  - 델타 인증서는 **양자 내성(PQC) 서명 알고리즘**으로 서명된 인증서
  - **기본 인증서에는 델타 인증서를 가리키는 비필수 확장 필드(Delta Certificate Descriptor)가 포함**
    - 이 확장 필드에는 델타 인증서의 PQC 공개키, 델타 인증서의 서명 값 등이 요약되어 저장
    - 필요 시 이를 이용해 델타 인증서를 재구성할 수 있음

  - 전통 알고리즘만 지원하는 환경에서는 Base Cert만 검증
  - PQC를 지원하는 환경에서는 Delta Cert까지 활용해 “양자 안전성”을 보장
  - 아직 표준화 진행 논의 중

  - 장점
    - **상황별 선택적 사용 :** 클라이언트나 서버가 **둘 중 어느 알고리즘을 쓸지**, 혹은 **둘 다 확인할지** 동적으로 결정 가능
  - 단점
    - **인증서 크기와 연산 부담 증가:** 하나의 인증서가 **둘 이상의 공개키·서명 정보를 동시에 포함**하기 때문에, 인증서 파일 사이즈가 커지고, TLS 핸드셰이크 등에서의 **검증 연산 비용**도 증가할 수 있음.
    - **구현 복잡도**

PQ X.509 테스트 수행

- X.509를 PQ X.509로 전환 시 PQ 서명 알고리즘, PQ 공개 키 알고리즘, PQ 공개키, PQ 서명을 포함한 핵심 요소 포함해야 함



- 테스트 환경
  - PQ 공개키와 서명은 크기가 커서 인증서 크기에 영향을 미침
  - 인증서 효율성과 보안을 모두 고려해 Legacy DSA 알고리즘과 PQC를 결합한 Composite 인증서 테스트
  - 기존 X.509에서 PQ X.509로 전환 시 크기, 알고리즘 특성 등 고려사항 분석 필요
  - KpqC와 NIST PQC 표준을 비교 분석하여 적합성 평가 진행

PQ X.509 테스트 수행

- **Open Quantum Safe (OQS) 프로젝트**
  - 양자알고리즘 기반 프로토콜·애플리케이션 통합을 지원하는 프로젝트
  - 구성에는 liboqs 라이브러리와 통합 프로토타입이 포함
  - 2025년 4월 17일: liboqs 0.13.0 버전 출시
- **liboqs**
  - liboqs는 양자 내성 암호 알고리즘을 모아 놓은 오픈 소스 라이브러리
  - 다양한 KEM 및 전자서명 알고리즘에 대한 API 제공
  - OpenSSL, SSH, VPN 등 프로토콜과의 통합 테스트에 활용
  - 실제 시스템에 PQC 알고리즘을 적용하는 데 사용

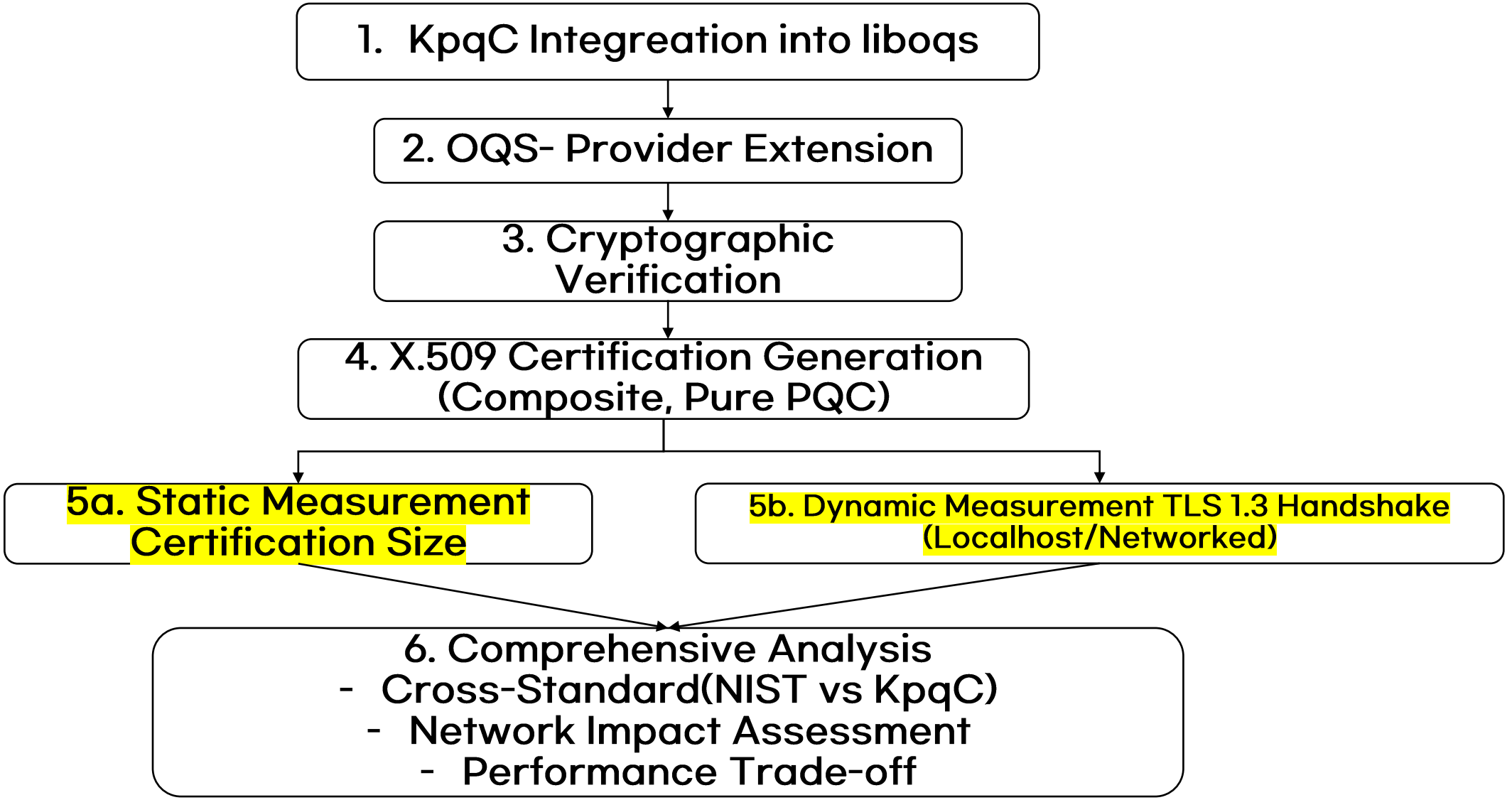
liboqs

- 특징
  - 공통 API 데이터 구조 : OQS_KEM, OQS_SIG 구조체 사용
    - 공통 구조체: 알고리즘, 키 크기, 보안 수준, 함수 포인터
    - OQS_KEM_keypair, OQS_KEM_encaps
  - 다수의 구현체 통합
    - 공식 구현 참조, PQClean 구현 등 코드 통합
  - 테스트 및 벤치마크 지원
    - 각 알고리즘에 대한 KAT, 속도 벤치마크 코드 포함

```
kem->method_name = OQS_KEM_alg_ml_kem_512;
kem->alg_version = "FIPS203";

kem->claimed_nist_level = 1;
kem->ind_cca = true;

kem->length_public_key = OQS_KEM_ml_kem_512_length_public_key;
kem->length_secret_key = OQS_KEM_ml_kem_512_length_secret_key;
kem->length_ciphertext = OQS_KEM_ml_kem_512_length_ciphertext;
kem->length_shared_secret = OQS_KEM_ml_kem_512_length_shared_secret;
kem->length_keypair_seed = OQS_KEM_ml_kem_512_length_keypair_seed;

kem->keypair = OQS_KEM_ml_kem_512_keypair;
kem->keypair_derand = OQS_KEM_ml_kem_512_keypair_derand;
kem->encaps = OQS_KEM_ml_kem_512_encaps;
kem->decaps = OQS_KEM_ml_kem_512_decaps;
```

[지원 알고리즘 목록]	KEM	DSA
NIST PQC	**Kyber**(Round 3 ver), **ML-KEM**(FIPS 203 ver) **HQC**(Round4 ver)	**Dilithium**(Round 3 ver), **Falcon**, **ML-DSA**(FIPS 204 ver), **SPHINCS+**
Additional Round2	-	**MAYO, CROSS(ver 2), UOV, SNOVA(ver 2)**
상태 기반 해시	-	**LMS, XMSS**
NIST PQC Round 4	**Classic McEliece, BIKE**	-
기타	**FrodoKEM, NTUR-Prime**	-

KpqC Migration into TLS/PKI



PQ X.509 인증서 크기

- FALCON이 가장 작은 인증서 크기를 가지고 있음

Family	Scheme	Security level	PQC-only Certificate(bytes)	Composite Certificate(bytes)
Classical	secp256r1	1	385	-
NIST PQC	FALCON512	1	1,788	1,941
KpqC	AIMER128s	1	4,427	5,682
KpqC	AIMER128f	1	6,155	6,310
NIST PQC	SPHINCS+SHAKE128fsimple	1	17,382	17,536
KpqC	HAETAE2	2	2,702	2,857
NIST PQC	MLDSA44	2	3,977	4,120

Family	Scheme	Security level	PQC-only Certificate(bytes)	Composite Certificate(bytes)
Classical	secp384r1	3	447	-
KpqC	HAETAE3	3	4,057	4,276
NIST PQC	MLDSA65	3	5,506	5,713
KpqC	AIMER192s	3	9,404	9,624
KpqC	AIMer192f	3	13,340	13,559

Family	Scheme	Security level	PQC-only Certificate(bytes)	Composite Certificate(bytes)
Classical	secp521r1	5	521	-
NIST PQC	FALCON1024	5	3,304	3,596
KpqC	HAETAE5	5	5,264	5,554
NIST PQC	MLDSA87	5	7,464	7,742
KpqC	AIMER256s	5	17,356	17,647
KpqC	AIMer256f	5	25,420	25,711

P-256 인증서 + 하이브리드 TLS

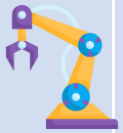
- 맥북-라즈베리파이 핸드쉐이크 200회 수행

Family	Scheme	Time(ms)	Overhead vs ECC
Classical	secp256r1(Baseline)	98.32	-
NIST PQC	secp256r1+mlekm512	130.14	+32.36%
KpqC	secp256r1+smaug_t1	130.15	+32.37%
KpqC	secp256r1+ntruplusKEM576	135.68	+38.00%

Family	Scheme	Time(ms)	Overhead vs ECC
Classical	secp384r1(Baseline)	109.90	-
NIST PQC	secp384r1+mlkem768	144.75	+31.71%
KpqC	secp384r1+ntruplusKEM768	146.70	+33.48%
KpqC	secp384r1+ntruplusKEM864	150.63	+37.06%
KpqC	secp384r1+smaug_t3	152.04	+38.34%

Family	Scheme	Time(ms)	Overhead vs ECC
Classical	secp521r1(Baseline)	125.19	-
KpqC	secp521r1+ntruplusKEM1152	166.04	+32.63%
NIST PQC	secp521r1+mlkem1024	167.41	+33.72%
KpqC	secp521r1+smuag_t5	176.40	+40.90%

산업 분야에서 사용하는 보안 프로토콜

산업 분야	리소스 / 실시간 제약 환경을 위한 보안 프로토콜
 우주 통신	• 위성/지상 간 데이터는 모두 암호화 (AES-GCM)하여 전송 (TM/TC 보호) • 지상망에서는 IPsec을 사용해 안전한 터널 구성
 자동차 내부 통신	• CAN 통신 보안: 차량 내부 제어기 간 통신을 해킹으로부터 보호 • 이더넷 보안: 차량용 고속 네트워크의 안전한 데이터 전송 보장
 자동차 외부 통신	• V2V/V2I 통신 보안: 차량 간 및 차량-인프라 간 실시간 정보 교환 보호 • Cellular-V2X 보안: 5G 기반 커넥티드카 통신의 안전성 확보
 드론 통신	• 드론 원격 조종 보안: 조종사와 드론 간 명령 전달을 암호화하여 해킹 방지 (비행 경로, 고도 조절) • 무선 주파수 보안: 드론과 조종기 간 무선 신호를 AES-GCM/CTR로 암호화하여 도청 차단
로봇 통신 	• 공장 로봇 보안: 제조 라인 내 로봇 간 데이터 통신을 암호화 → 생성 명령 및 작업 상태 보호 • 산업 자동화 통신 보안: 로봇과 제어 시스템 간 안전한 데이터 교환으로 오작동 및 침해 예방

각 산업 분야별 네트워크 성능 요구사항

산업 분야	주요 경로	RTT (Round-Trip Time) 범위	출처
우주 (LEO)	저궤도(LEO) 위성 ↔ 지상국	20 ~ 40 ms	[1]
우주 (GEO)	고궤도(GEO) 위성 ↔ 지상국	600 ms 이상	[1]
자동차 내부	EtherCAT/TSN 기반 산업용 이더넷	< 1 ms	[2]
자동차 외부 (V2V)	셀룰러-차량/사물 통신(DSRC/C-V2X) 직접 통신	< 10 ms	[3]
드론 (제어 C2)	컨트롤러 ↔ 드론 (FPV 조종)	5 ~ 30 ms	[4]
드론 (영상 스트림)	드론 ↔ 지상국 영상 링크	디지털 : 28 ~ 40 ms 아날로그 : 10 ~ 20 ms	[4]
로봇 (산업용 제어)	EtherCAT/PROFINET 등 필드버스	0.03 ~ 1 ms (PROFINET IRT/EtherCAT 기준)	[5]

- FPV : 드론에 장착된 카메라를 통해 실시간으로 조종자의 시야를 보면서 조종하는 기술
- EtherCAT : 이더넷 기반 실시간 산업용 통신 프로토콜
- PROFINET : 산업용 이더넷 통신 프로토콜
- TSN(Time-Sensitive Networking) : 이더넷 기반 실시간 통신을 가능하게 하는 IEEE 표준

[1] https://reliasat.com/satellite-communications-evolution-from-geo-to-leo/
[2] https://iebmedia.com/technology/industrial-ethernet/looking-inside-the-real-time-capabilities-of-industrial-ethernet/
[3] https://5gaa.org/content/uploads/2018/11/5GAA_P-190033_V2X-Functional-and-Performance-Test-Report_final-1.pdf
[4] Tomaszewski, Lechosław, Robert Kołakowski, and Paweł Korzec. "On 5G support of cross-border UAV operations." *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2020.
[5] Mayoral-Vilches, Víctor, et al. "ROS 2 on a Chip, Achieving Brain-Like Speeds and Efficiency in Robotic Networking." *arXiv preprint arXiv:2404.18208* (2024).

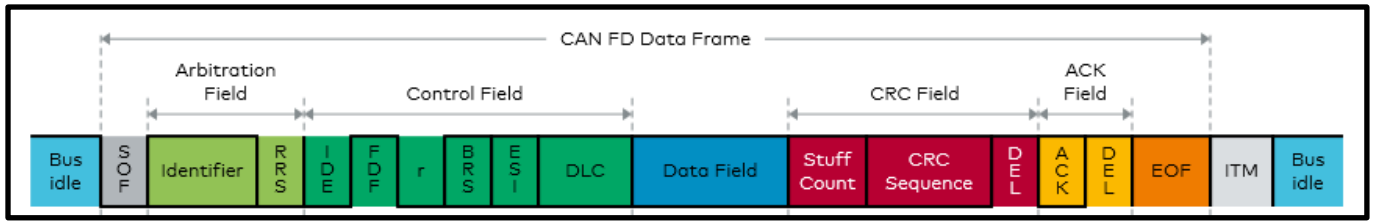
자동차 통신을 중심으로 PQC 적용 방안 모색

- 대부분의 내부 통신 BUS는 **실시간성을 위해 프레임 크기가 작음**
  - **대용량 payload가 필요한 PQC는 실용적인 적용이 어려움**

프로토콜	1-프레임 payload	지연 한계	현행 보안	KpqC 적용 가능성	비고	출처
CAN 2.0B	8 bytes	≤ 1 ms	X	X	클래식 CAN	[1]
CAN-FD	64 bytes	≤ 1 ms	X	조건부 적용 가능	차세대 CAN	[2]
LIN 2.x	8 bytes	≤ 10 ms	X	X	저속, 저가 네트워크 (창문·시트·조명 등 적용)	[3]
FlexRay (전자식 제동 통신)	254 bytes	≤ 2 ms	MAC(옵션)	조건부 적용 가능	과거 일부 고급 차량 적용 (보급률 추정치 ≈ 10 %)	[4]
100BASE-T1 (이더넷, ADAS)	최대 1,500 bytes	Class A ≤ 2 ms Class CDT ≤ 100 µs	TLS/ DTLS	적용 가능	신차 위주 적용 (보급률 추정치 ≈ 30%)	[5]

[1] https://www.iso.org/standard/86384.html
[2] https://tekeye.uk/downloads/can_fd_spec.pdf
[3] https://www.cs-group.de/wp-content/uploads/2016/11/LIN_Specification_Package_2.2A.pdf
[4] https://www.iso.org/obp/ui/#iso:std:iso:17458:-4:ed-1:v1:en
[5] Abdelgader, Abdeldime MS, and Wu Lenan. "The physical layer of the IEEE 802.11 p WAVE communication standard: the specifications and challenges." Proceedings of the world congress on engineering and computer science. Vol. 2. 2014.

CAN-FD 프로토콜



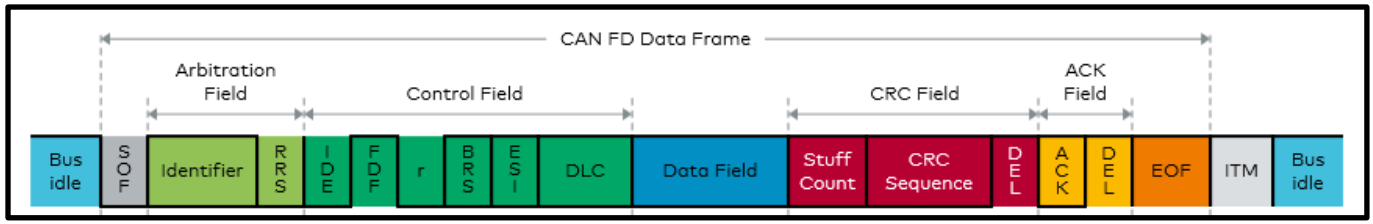
- CAN: ECU(전자제어장치) 간 실시간 통신 네트워크 표준(ISO 11898)
- **CAN-FD: Classic CAN(2.0B) 확장 버전**
  - **데이터 구간 속도 향상(BRS), 데이터 길이 확장(EDL)**

항목	CAN 2.0B	CAN-FD[1]
최대 데이터 길이	8 bytes	64 bytes
비트레이트	단일 속도	이중 속도 Nominal 구간(Arbitration + Control + ACK Field): 1 Mbps 이하 Data 구간(Data + CRC Field): 2, 5, 8 Mbps[2]
CRC	15/17-bit	데이터 ≤ 16 byte: 17-bit 데이터 > 16 byte: 21-bit
Bit-Stuffing 방식	고정(비트5:1)	가변(비트0~5: 1)

<PQC 적용 시 고려 사항>
데이터 크기 최대 64 Byte → PQC 데이터 전송 시 분할 필요(프레임 다량 발생)
가변 Stuff-bit: Worst-case 전송 지연 계산 필요

[1] https://tekeye.uk/downloads/can_fd_spec.pdf
[2] Zeltwanger, Holger. "CAN FD network design hints and recommendations." SAE International Journal of Passenger Cars-Electronic and Electrical Systems 9.2016-01-0060 (2016): 89-92.

CAN-FD 실제 전송 시 패킷 크기



- 전송하고자 하는 데이터가 20~64 bytes 일 경우 (1-프레임 안에 끝날 경우)

필드	길이 (bit)	비고 / 설명
SOF+ Arbitration + Control	23-bit	ID(우선순위 선정), FD 모드 플래그, DLC(데이터 길이 코드) 등
CRC + Stuff-counter + FSB	32-bit	데이터 ≥ 20 Byte면 CRC-21 사용
ACK + EOF + ITM	12-bit	ACK(수신 확인), EOF(프레임 종료 표시), ITM(프레임 간격 비트)
Data Field	0 ~ 512-bit	실제 사용자 데이터, DLC 값과 길이 매핑(E.g. DLC 15 -> 64 Byte)
가변 Stuff-bit	0 ~ 107-bit	**worst-case: 원본 5-bit당 1-bit(20%)[1] -> 약 107-bit** 원본: CRC 직전(Arbitration + Control ~ Data -> 535-bit)

<실제 전송 시 1-프레임 당 최대 패킷 크기>
고정 오버헤드(67-bits[2]) + 데이터 필드(512-bits) + 최대 Stuff-bit(107-bit)
≈ 86 Byte(686-bit)

(고정 오버헤드: SOF+ Arbitration + Control + CRC + Stuff-counter + FSB + ACK + EOF + ITM)

[1] https://www.can-cia.org/fileadmin/cia/documents/proceedings/2012_oertel.pdf
[2] https://web.archive.org/web/20151211125301/http://www.bosch-semiconductors.de/media/ubk_semiconductors/pdf_1/canliteratur/can_fd_spec.pdf

CAN-FD 패킷 비교(KEM/DSA)

- MTU(64 bytes)를 넘는 데이터
  → 64 bytes씩 분할하여 전송, 프레임당 최대 오버헤드 22 bytes 추가

알고리즘	보안 레벨	공개키 PK (byte)	암호문, 서명 (byte)	CAN 패킷 최대 크기 (byte) (PK / CT, PK / Sig)	프레임 수 (PK / CT, PK / Sig)
ML-KEM512	1	800	768	1,086 / 1,032	13 / 12
HQC-128	1	2,249	4,433	3,041 / 5,973	36 / 70
SMAUG-T1	1	672	672	914 / 914	11 / 11
NTRU+KEM576	1	864	864	1,172 / 1,172	14 / 14
Falcon-512	1	897	666	1,227 / 908	15 / 11
SPHINCS+128s	1	32	7,856	54 / 10,562	1 / 123
SPHINCS+128f	1	32	17,088	54 / 22,962	1 / 267
AIMer128s	1	32	4,160	54 / 5,590	1 / 65
AIMer128f	1	32	5,888	54 / 7,912	1 / 92
ML-DSA44	2	1,312	2,420	1,774 / 3,256	21 / 38
HAETAE2	2	992	1,474	1,344 / 2,002	16 / 24

CAN-FD 전송 지연 산출

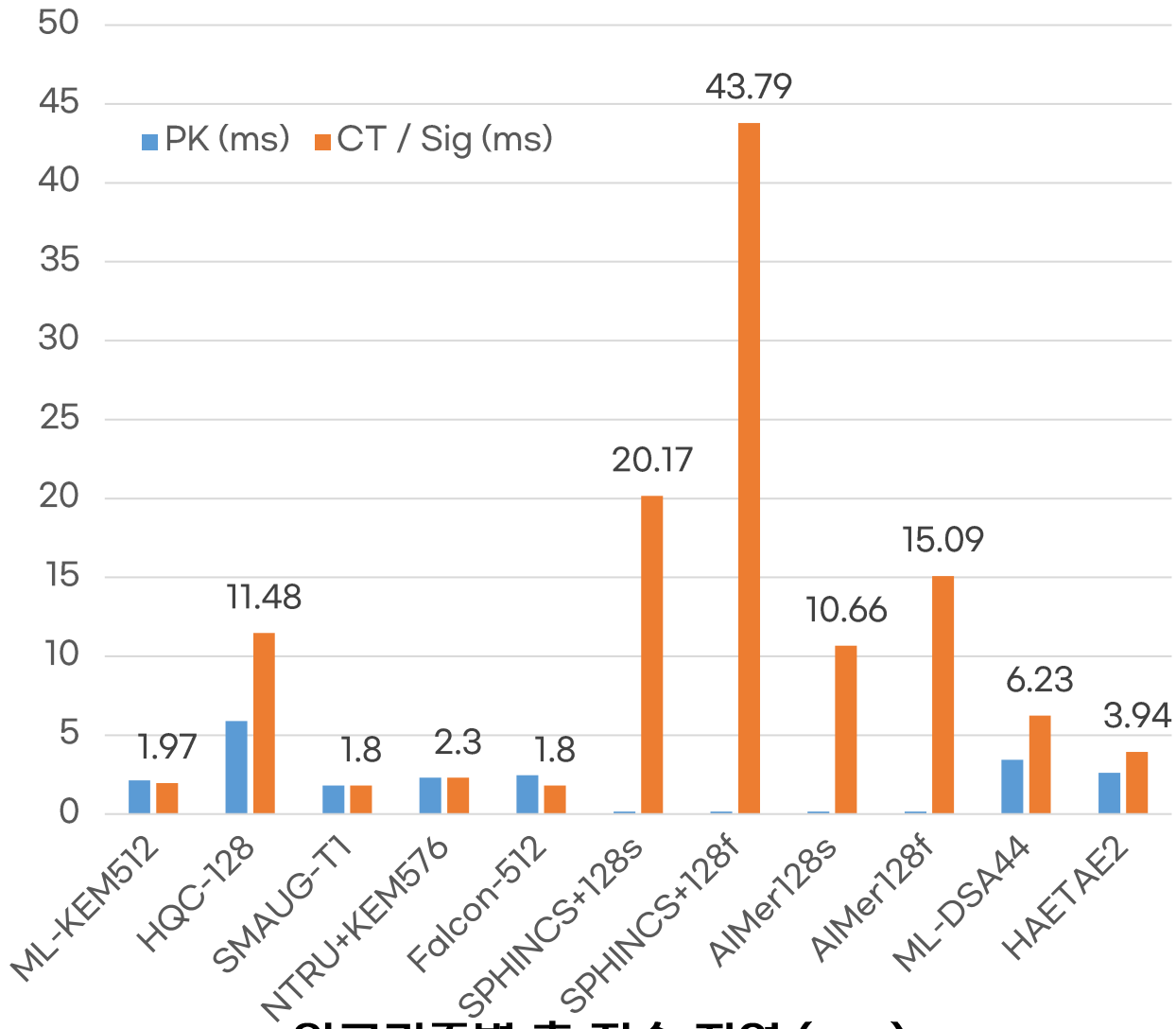
<1-프레임 지연 계산(worst case)>

① Nominal 구간 @1 Mbps
- Arbitration + Control = 22-bit → 22 µs
- ACK + EOF + IFS = 12-bit → 12 µs
- 22 + 12 = 34 µs

② Data 구간 @5 Mbps
- 데이터 64 B = 512-bit
- CRC + Stuff-counter + FSB = 32-bit
- 가변 Stuff-bit(≈ 107-bit)

(512 + 32 + 107) bit ÷ 5 Mbps ≈ 130 µs

③ 합계
34 µs + 130 µs = 0.164 ms



알고리즘별 총 전송 지연 (ms)

Hartwich, Florian. "Bit time requirements for can FD." *Proceedings of the 14th International CAN Conference*. 2013.

CAN-FD ECU 도메인 별 실시간성 및 PQC 적용 가능성 평가

- CAN 네트워크 상 ECU 도메인은 ASIL 등급[1]에 따라 구분 가능
  - **ASIL:** Automotive Safety Integrity Level(차량 안전 등급 체계)
  - **A → B → C → D 순으로 위험도 높음(더 엄격한 안전 조치, 검증 요구)**

주요 ECU 도메인	ASIL 등급	지연 한계[2]	ECU 주요 신호	PQC 적용 가능성
Power-train	D	≤ 1 ms	엔진 ECU, 변속기, 연료분사	불가능
Chassis / ADAS	C~D	≤ 2 ms	ABS (브레이크 잠김 방지), EPS (전자식 조향), 레이더, 카메라	일부 알고리즘 가능
Body / Comfort	A~B	≤ 50 ms	창문, 시트, 조명, 공조	대부분 알고리즘 가능
Diagnostics / OTA	A	≤ 100 ms	서비스 진단, 펌웨어 업데이트	

ASIL C,D 등급 도메인(브레이크,엔진 등)은 μs~ms 단위 실시간성 + 최고 무결성이 필수
PQC처럼 통신 부담이 큰 알고리즘은 사실상 부적합

[1] Gheraibia, Youcef, et al. "An overview of the approaches for automotive safety integrity levels allocation." Journal of failure analysis and prevention 18 (2018): 707-720.
[2] https://www.can-cia.org/fileadmin/cia/documents/publications/cnlm/june_2024/24-2_cnlm.pdf

CAN-FD PQC 적용 가능성

▲: 최적화 시 적용 가능성 있음
(비트레이트 향상, 프레임 재배치 등)

알고리즘	CT / Sig (Byte)	전송 지연 (ms)	Power-train (≤ 1 ms)	Chassis / ADAS (≤ 2 ms)	Body / Diagnostics (≤ 50 ms)
ML-KEM512	768	1.97	X	O	O
HQC-128	4,433	11.48	X	X	O
SMAUG-T1	672	1.8	X	O	O
NTRU+KEM576	864	2.3	X	▲	O
Falcon-512	666	1.8	X	O	O
SPHINCS+128s	7,856	20.17	X	X	O
SPHINCS+128f	17,088	43.79	X	X	O
AIMer128s	4,160	10.66	X	X	O
AIMer128f	5,888	15.09	X	X	O
ML-DSA44	3,256	6.23	X	X	O
HAETAE-2	2,002	3.94	X	▲	O

CAN-FD 버스에 PQC를 **전면 적용**하기에는 지연·대역폭 제약이 크기에 **불가능**
Power-train/Chassis 및 Body·Diagnostics에 PQC 적용을 도입하는 **하이브리드 전략**이 현실적

결론

- 20세기 마인드 (Pre-quantum):
  아이 (KpqC)는 낳기만 하면 (선정만 하면)
  알아서 큰다 (알아서 활용된다)?

- 21세기 마인드 (Post-quantum):
  아이 (KpqC)를 낳았으면 잘 키워야 한다. (이옥연 명예 회장님 말씀)
  → KpqC 선정된 이후 잘 성장할 수 있도록 지원하는 것이 중요!

아이 (KpqC)들이 행복하게 잘 클 수 있도록

영양가 있는 식사 (국가 단위 펀딩)도 제공하고
멋진 옷 (표준화)도 입혀주고
따뜻한 집 (서비스 하우징)에서 지낼 수 있도록

국내 암호 커뮤니티의 지속적인 관심이 필요!

감사합니다.

Appendix: KpqC 알고리즘 Clean 구현 Speed 측정 상세

성능 측정 환경

Intel	ARM
• OS: Ubuntu 23.10.1 • CPU: Intel i5-8259U (2.30 GHz) • RAM: 16GB • Compiler: gcc 13.2.0 • Optimization Level: -O3	• OS: macOs Sonoma 14.4.1 • CPU: Apple M2 (3.23 GHz) • RAM: 8GB • Compiler: Apple clang 15.0.0 • Optimization Level: -O3

Kyber, SMAUG-T, NTRU+ 키 크기 비교

Rank	Scheme	Public key
1	SMAUG-T1	672
	SMAUG-Timer	672
3	KYBER-512	800
4	NTRU+KEM576	864
	NTRU+PKE576	864
6	SMAUG-T3	1,088
7	NTRU+KEM768	1,152
	NTRU+PKE768	1,152
9	KYBER-768	1,184
10	NTRU+KEM864	1,296
	NTRU+PKE864	1,296
12	SMAUG-T5	1,440
13	KYBER-1024	1,568
14	NTRU+KEM1152	1,728
	NTRU+PKE1152	1,728

Rank	Scheme	Secret key
1	SMAUG-T1	832
	SMAUG-Timer	832
3	SMAUG-T3	1,312
4	KYBER-512	1,632
5	NTRU+KEM576	1,760
	NTRU+PKE576	1,760
7	SMAUG-T5	1,792
8	NTRU+KEM768	2,336
	NTRU+PKE768	2,336
10	KYBER-768	2,400
11	KYBER-1024	2,592
12	NTRU+KEM864	2,624
	NTRU+PKE864	2,624
14	NTRU+KEM1152	3,488
	NTRU+PKE1152	3,488

Rank	Scheme	Ciphertext
1	SMAUG-Timer	608
2	SMAUG-T1	672
3	KYBER-512	768
4	NTRU+KEM576	864
	NTRU+PKE576	864
6	SMAUG-T3	992
7	KYBER-768	1,088
8	NTRU+KEM768	1,152
	NTRU+PKE768	1,152
10	NTRU+KEM864	1,296
	NTRU+PKE864	1,296
12	SMAUG-T5	1,376
13	KYBER-1024	1,568
14	NTRU+KEM1152	1,728
	NTRU+PKE1152	1,728

Dililthium, FALCON, HAETAE 키 크기 비교

Rank	Scheme	Public key
1	FALCON-512	897
2	HAETAE2	992
3	Dilithium-II	1,312
4	HAETAE3	1472
6	FALCON-1024	1,793
7	Dilithium-III	1,952
8	HAETAE5	2080
10	Dilithium-V	2,592

Rank	Scheme	Secret key
1	FALCON-512	1,281
2	HAETAE2	1,408
3	HAETAE3	2,112
4	FALCON-1024	2,305
5	Dilithium-II	2,528
7	HAETAE5	2,752
9	Dilithium-III	4,000
10	Dilithium-V	4,864

Rank	Scheme	Signature
1	FALCON-512	666
2	FALCON-1024	1,280
3	HAETAE2	1,474
4	HAETAE3	2,349
5	Dilithium-II	2,420
7	HAETAE5	2,948
8	Dilithium-III	3,293
10	Dilithium-V	4,595

SPHINCS, AIMer 키 크기 비교

Rank	Scheme	Public key
1	AIMer128f	32
	AIMer128s	32
	SPHINCS+128	32
4	AIMer192f	48
	AIMer192s	48
	SPHINCS+192	48
7	SPHINCS+256	49
8	AIMer256f	64
	AIMer256s	64

Rank	Scheme	Secret key
1	AIMer128f	48
	AIMer128s	48
3	SPHINCS+128	64
4	AIMer192f	72
	AIMer192s	72
6	AIMer256f	96
	AIMer256s	96
8	SPHINCS+192	96
9	SPHINCS+256	128

Rank	Scheme	Signature
1	AIMer128s	4,160
2	AIMer128f	5,888
3	SPHINCS+128s	7,856
4	AIMer192s	9,120
5	AIMer192f	13,056
6	SPHINCS+192s	16,224
7	AIMer256s	17,056
8	SPHINCS+128f	17,088
9	AIMer256f	25,120
10	SPHINCS+256s	29,792
11	SPHINCS+192f	35,664
12	SPHINCS+256f	49,856

Kyber, SMAUG-T, NTRU+ 성능 비교 (Intel)

Rank	Scheme	Keypair(avg)
1	SMAUG-Timer	77,568
2	SMAUG-T1	79,536
3	Kyber512	110,653
4	NTRU+KEM576	130,594
5	NTRU+PKE576	134,416
6	SMAUG-T3	147,832
7	NTRU+KEM768	173,158
8	NTRU+PKE768	174,246
9	NTRU+KEM864	195,173
10	Kyber768	198,651
11	NTRU+PKE864	202,711
12	SMAUG-T5	221,543
13	Kyber1024	266,290
14	NTRU+PKE1152	310,068
15	NTRU+KEM1152	315,045

Rank	Scheme	encap(avg)
1	SMAUG-Timer	61,351
2	SMAUG-T1	61,515
3	NTRU+KEM576	67,355
4	NTRU+PKE576	69,784
5	NTRU+KEM768	107,650
6	NTRU+PKE864	109,906
7	NTRU+PKE768	113,481
8	NTRU+KEM864	114,314
9	Kyber512	125,375
10	SMAUG-T3	128,592
11	NTRU+PKE1152	141,300
12	NTRU+KEM1152	151,707
13	Kyber768	174,804
14	SMAUG-T5	215,704
15	Kyber1024	244,419

Rank	Scheme	decap(avg)
1	NTRU+KEM576	79,986
2	SMAUG-Timer	80,608
3	SMAUG-T1	82,346
4	NTRU+PKE576	83,498
5	NTRU+PKE768	125,727
6	NTRU+KEM768	126,515
7	NTRU+PKE864	138,923
8	Kyber512	143,529
9	NTRU+KEM864	145,264
10	SMAUG-T3	161,020
11	NTRU+PKE1152	175,211
12	NTRU+KEM1152	183,286
13	Kyber768	210,180
14	SMAUG-T5	264,539
15	Kyber1024	290,399

Dililthium, FALCON, HAETAE 성능 비교 (Intel)

Rank	Scheme	Keypair (avg)
1	dilithium2	278,954
2	dilithium3	482,104
3	dilithium5	734,609
4	HAETAE2	1,343,166
5	HAETAE3	1,777,115
6	HAETAE5	1,983,237
7	falcon-padded-512	37,897,367
8	falcon-512	37,959,474
9	falcon-1024	108,091,196
10	falcon-padded-1024	108,252,565

Rank	Scheme	Sign(avg)
1	dilithium2	450,483
2	dilithium5	1,463,427
3	HAETAE5	1,564,403
4	dilithium3	2,465,069
5	HAETAE3	3,954,599
6	HAETAE2	4,273,817
7	falcon-512	10,617,704
8	falcon-padded-512	10,959,368
9	falcon-1024	23,159,283
10	falcon-padded-1024	23,186,413

Rank	Scheme	verify(avg)
1	falcon-512	132,067
2	falcon-padded-512	133,669
3	HAETAE2	204,595
4	falcon-padded-1024	281,205
5	falcon-1024	281,777
6	dilithium2	281,907
7	HAETAE3	333,241
8	HAETAE5	431,209
9	dilithium3	440,805
10	dilithium5	737,849

SPHINCS, AIMer 성능 비교 (Intel)

Rank	Scheme	Keypair(avg)
1	AIMer128s	85,760
2	AIMer128f	86,858
3	AIMer192s	221,051
4	AIMer192f	221,939
5	AIMer256f	532,954
6	AIMer256s	536,662
7	sphincs-sha2-128f-simple	3,512,168
8	sphincs-sha2-192f-simple	5,106,785
9	sphincs-shake-128f-simple	5,676,001
10	sphincs-shake-192f-simple	8,233,255
11	sphincs-sha2-256f-simple	13,540,513
12	sphincs-shake-256f-simple	21,778,482
13	sphincs-sha2-256s-simple	213,627,804
14	sphincs-sha2-128s-simple	222,113,099
15	sphincs-sha2-192s-simple	323,514,039
16	sphincs-shake-256s-simple	346,536,078
17	sphincs-shake-128s-simple	359,032,336
18	sphincs-shake-192s-simple	524,022,262

Rank	Scheme	sign(avg)
1	AIMer128f	7,020,534
2	AIMer192f	13,314,965
3	AIMer256f	34,079,571
4	AIMer128s	54,057,979
5	sphincs-sha2-128f-simple	81,191,592
6	AIMer192s	103,955,683
7	sphincs-shake-128f-simple	131,239,147
8	sphincs-sha2-192f-simple	133,555,092
9	sphincs-shake-192f-simple	212,061,614
10	AIMer256s	263,703,008
11	sphincs-sha2-256f-simple	272,214,600
12	sphincs-shake-256f-simple	440,803,080
13	sphincs-sha2-128s-simple	1,690,989,832
14	sphincs-sha2-256s-simple	2,630,577,802
15	sphincs-shake-128s-simple	2,725,259,275
16	sphincs-sha2-192s-simple	2,989,370,882
17	sphincs-shake-256s-simple	4,124,025,081
18	sphincs-shake-192s-simple	4,710,044,154

Rank	Scheme	verify(avg)
1	sphincs-sha2-128s-simple	1,769,297
2	sphincs-sha2-192s-simple	2,394,749
3	sphincs-shake-128s-simple	2,763,529
4	sphincs-sha2-256s-simple	3,554,691
5	sphincs-shake-192s-simple	3,872,933
6	sphincs-sha2-128f-simple	4,989,027
7	sphincs-shake-256s-simple	5,538,012
8	AIMer128f	6,429,942
9	sphincs-sha2-256f-simple	7,081,467
10	sphincs-sha2-192f-simple	7,123,456
11	sphincs-shake-128f-simple	7,812,153
12	sphincs-shake-192f-simple	11,374,784
13	sphincs-shake-256f-simple	11,778,032
14	AIMer192f	13,025,861
15	AIMer256f	31,939,622
16	AIMer128s	53,876,227
17	AIMer192s	102,567,956
18	AIMer256s	257,224,646

Kyber, SMAUG-T, NTRU+ 성능 비교 (ARM)

Rank	Scheme	Keypair(avg)
1	SMAUG-T1	47,999
2	SMAUG-Timer	48,155
3	kyber512	70,685
4	NTRU+PKE576	88,560
5	NTRU+KEM576	88,861
6	SMAUG-T3	94,312
7	Kyber768	109,502
8	NTRU+PKE864	126,033
9	NTRU+KEM768	126,058
10	NTRU+PKE768	135,166
11	NTRU+KEM864	136,047
12	SMAUG-T5	150,879
13	Kyber1024	165,118
14	NTRU+PKE1152	227,871
15	NTRU+KEM1152	228,330

Rank	Scheme	encap(avg)
1	SMAUG-T1	42,359
2	NTRU+PKE576	43,327
3	NTRU+KEM576	43,354
4	SMAUG-Timer	55,262
5	NTRU+PKE768	64,254
6	NTRU+KEM768	67,050
7	NTRU+KEM864	68,712
8	NTRU+PKE864	69,044
9	SMAUG-T3	86,495
10	NTRU+KEM1152	92,557
11	NTRU+PKE1152	92,727
12	Kyber512	125,375
13	SMAUG-T5	139,620
14	Kyber768	174,804
15	Kyber1024	244,419

Rank	Scheme	decap(avg)
1	NTRU+PKE576	46,638
2	NTRU+KEM576	49,144
3	SMAUG-T1	56,712
4	SMAUG-Timer	55,651
5	NTRU+PKE768	71,626
6	NTRU+KEM768	74,659
7	NTRU+PKE864	82,034
8	NTRU+KEM864	85,392
9	Kyber512	87,617
10	SMAUG-T3	105,874
11	NTRU+PKE1152	106,047
12	NTRU+KEM1152	110,782
13	Kyber768	134,352
14	SMAUG-T5	165,481
15	kyber1024	195,524

Dililthium, FALCON, HAETAE 성능 비교 (ARM)

Rank	Scheme	Keypair (avg)
1	dilithium2	190,398
2	dilithium3	362,409
3	dilithium5	540,914
4	HAETAE2	835,765
5	HAETAE3	1,180,426
6	HAETAE5	1,378,932
7	falcon-512	35,583,909
8	falcon-1024	106,966,738

Rank	Scheme	Keypair (avg)
1	dilithium2	306,143
2	HAETAE2	828,534
3	HAETAE3	979,182
4	dilithium5	1,023,099
5	HAETAE5	1,026,078
6	dilithium3	1,741,397
7	falcon-512	10,044,417
8	falcon-1024	21,844,507

Rank	Scheme	Keypair (avg)
1	Falcon-512	91,243
2	HAETAE2	136,239
3	Falcon-1024	182,233
4	Dilithium2	195,294
5	HAETAE3	237,614
6	HAETAE5	303,049
7	Dilithium3	321,151
8	Dilithium5	542,833

SPHINCS, AIMer 성능 비교 (ARM)

Rank	Scheme	Keypair(avg)
1	AIMer128f	56,314
2	AIMer128s	56,395
3	AIMer192f	123,386
4	AIMer192s	169,004
5	AIMer256s	291,148
6	AIMer256f	292,269
7	sphincs-sha2-128f-simple	3,649,196
8	sphincs-shake-128f-simple	3,952,483
9	sphincs-sha2-192f-simple	5,312,027
10	sphincs-shake-192f-simple	5,800,574
11	sphincs-sha2-256f-simple	13,962,889
12	sphincs-shake-256f-simple	15,202,018
13	sphincs-sha2-256s-simple	222,570,209
14	sphincs-sha2-128s-simple	230,498,741
15	sphincs-shake-256s-simple	242,995,939
16	sphincs-shake-128s-simple	252,400,728
17	sphincs-sha2-192s-simple	337,468,644
18	sphincs-shake-192s-simple	367,897,731

Rank	Scheme	Sign(avg)
1	AIMer128f	2,324,875
2	AIMer192f	5,887,515
3	AIMer256f	11,554,222
4	AIMer128s	18,404,651
5	AIMer192s	45,924,420
6	sphincs-sha2-128f-simple	84,542,863
7	AIMer256s	88,425,057
8	sphincs-shake-128f-simple	92,399,930
9	sphincs-sha2-192f-simple	138,373,380
10	sphincs-shake-192f-simple	148,824,161
11	sphincs-sha2-256f-simple	284,393,734
12	sphincs-shake-256f-simple	305,656,621
13	sphincs-sha2-128s-simple	1,750,772,084
14	sphincs-shake-128s-simple	1,915,535,299
15	sphincs-sha2-256s-simple	2,746,781,740
16	sphincs-shake-256s-simple	2,894,608,526
17	sphincs-sha2-192s-simple	3,103,763,239
18	sphincs-shake-192s-simple	3,310,393,909

Rank	Scheme	verify(avg)
1	sphincs-sha2-128s-simple	1,691,069
2	sphincs-shake-128s-simple	1,826,203
3	AIMer128f	2,158,816
4	sphincs-sha2-192s-simple	2,695,583
5	sphincs-shake-192s-simple	2,711,920
6	sphincs-sha2-256s-simple	3,758,715
7	sphincs-shake-256s-simple	3,998,496
8	sphincs-sha2-128f-simple	5,133,120
9	AIMer192f	5,482,033
10	sphincs-shake-128f-simple	5,565,225
11	sphincs-sha2-192f-simple	7,429,704
12	sphincs-sha2-256f-simple	7,765,007
13	sphincs-shake-192f-simple	8,013,609
14	sphincs-shake-256f-simple	8,148,539
15	AIMer256f	10,770,937
16	AIMer128s	18,288,170
17	AIMer192s	45,489,467
18	AIMer256s	87,568,082