liboqs KPQC 포팅

https://youtu.be/DEBnuNpbLrU

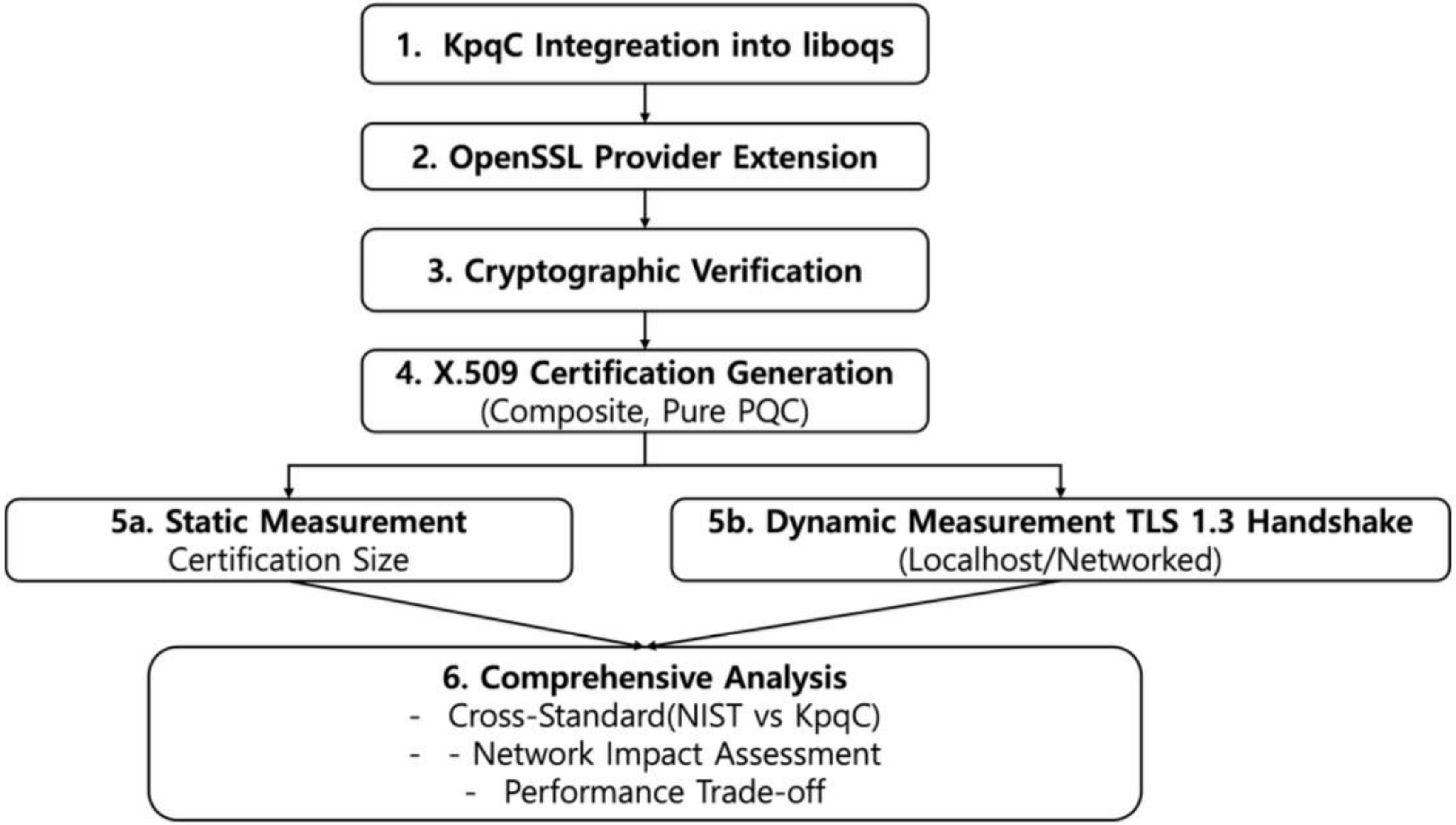
HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

Liboqs & oqs_provider

- Liboqs
  - 양자내성암호들의 표준화된 구현체를 모아놓은 라이브러리
  - 플랫폼 독립적인 API 제공
  - 벤치마크 기능 포함(알고리즘별 보안레벨, 키 사이즈, 속도 비교 가능)
  - 다른 소프트웨어에서 PQC 알고리즘을 쉽게 호출하도록 API 제공
  - 새로운 PQC 알고리즘을 연구하고 테스트하는데 사용됨
- Oqs-provider
  - Openssl 3.x의 “provider” 매커니즘을 이용해 PQC 알고리즘을 Openssl에 통합하는 모듈
  - 주요 기능
    - TLS 1.3 핸드쉐이크에서 PQC KEM 사용
    - PQC 기반 서명을 X.509 인증서에 사용
  - Liboqs에 구현된 PQC 알고리즘을 내부적으로 호출해 사용
- **Liboqs가 실제 PQC 알고리즘을 제공하는 구조**
- **Oqs-provider는 openssl 인터페이스에 맞춰 연결해주는 역할**

Liboqs KpqC 실험



1. KpqC Integreation into liboqs

2. OpenSSL Provider Extension

3. Cryptographic Verification

4. X.509 Certification Generation
(Composite, Pure PQC)

5a. Static Measurement
Certification Size

5b. Dynamic Measurement TLS 1.3 Handshake
(Localhost/Networked)

6. Comprehensive Analysis
- Cross-Standard(NIST vs KpqC)
- - Network Impact Assessment
- Performance Trade-off

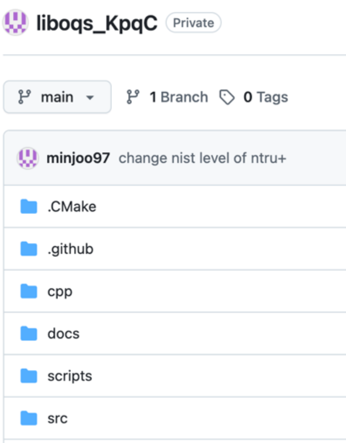
liboqs_KpqC

- 사용한 알고리즘
  - HAETAE와 AIMer의 경우 KpqCleanver2 사용
  - SMAUG-T와 NTRU+의 경우 최신 코드 사용
    - NTRU+

      github 86022d4319baf62377964f959d186af1e8d8f029 로 수정

      파라미터별 함수 명칭이 동일해서 컴파일할때 오류 발생-> 이를 위해 NAMESPACE 추가함
    - Smaug-T

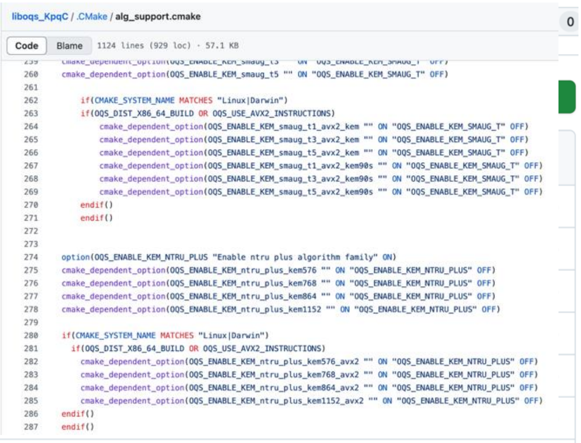
      Keccak -> SHAEK256으로 코드 수정


liboqs_KpqC Private

main · 1 Branch · 0 Tags

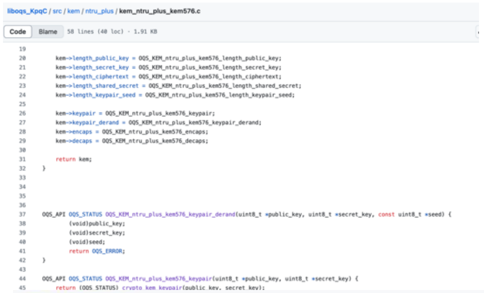
minjoo97 change nist level of ntru+

- .CMake
- .github
- cpp
- docs
- scripts
- src


liboqs_KpqC / .CMake / alg_support.cmake

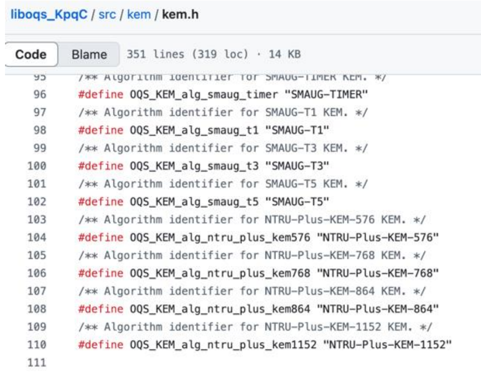
Code Blame 1124 lines (929 loc) · 57.1 KB

```
cmake_dependent_option(OQS_ENABLE_KEM_smaug_t5 "" ON "OQS_ENABLE_KEM_SMAUG_T" OFF)

    if(CMAKE_SYSTEM_NAME MATCHES "Linux|Darwin")
    if(OQS_DIST_X86_64_BUILD OR OQS_USE_AVX2_INSTRUCTIONS)
        cmake_dependent_option(OQS_ENABLE_KEM_smaug_t1_avx2_kem "" ON "OQS_ENABLE_KEM_SMAUG_T" OFF)
        cmake_dependent_option(OQS_ENABLE_KEM_smaug_t3_avx2_kem "" ON "OQS_ENABLE_KEM_SMAUG_T" OFF)
        cmake_dependent_option(OQS_ENABLE_KEM_smaug_t5_avx2_kem "" ON "OQS_ENABLE_KEM_SMAUG_T" OFF)
        cmake_dependent_option(OQS_ENABLE_KEM_smaug_t1_avx2_kem90s "" ON "OQS_ENABLE_KEM_SMAUG_T" OFF)
        cmake_dependent_option(OQS_ENABLE_KEM_smaug_t3_avx2_kem90s "" ON "OQS_ENABLE_KEM_SMAUG_T" OFF)
        cmake_dependent_option(OQS_ENABLE_KEM_smaug_t5_avx2_kem90s "" ON "OQS_ENABLE_KEM_SMAUG_T" OFF)
    endif()
    endif()


option(OQS_ENABLE_KEM_NTRU_PLUS "Enable ntru plus algorithm family" ON)
cmake_dependent_option(OQS_ENABLE_KEM_ntru_plus_kem576 "" ON "OQS_ENABLE_KEM_NTRU_PLUS" OFF)
cmake_dependent_option(OQS_ENABLE_KEM_ntru_plus_kem768 "" ON "OQS_ENABLE_KEM_NTRU_PLUS" OFF)
cmake_dependent_option(OQS_ENABLE_KEM_ntru_plus_kem864 "" ON "OQS_ENABLE_KEM_NTRU_PLUS" OFF)
cmake_dependent_option(OQS_ENABLE_KEM_ntru_plus_kem1152 "" ON "OQS_ENABLE_KEM_NTRU_PLUS" OFF)

if(CMAKE_SYSTEM_NAME MATCHES "Linux|Darwin")
  if(OQS_DIST_X86_64_BUILD OR OQS_USE_AVX2_INSTRUCTIONS)
    cmake_dependent_option(OQS_ENABLE_KEM_ntru_plus_kem576_avx2 "" ON "OQS_ENABLE_KEM_NTRU_PLUS" OFF)
    cmake_dependent_option(OQS_ENABLE_KEM_ntru_plus_kem768_avx2 "" ON "OQS_ENABLE_KEM_NTRU_PLUS" OFF)
    cmake_dependent_option(OQS_ENABLE_KEM_ntru_plus_kem864_avx2 "" ON "OQS_ENABLE_KEM_NTRU_PLUS" OFF)
    cmake_dependent_option(OQS_ENABLE_KEM_ntru_plus_kem1152_avx2 "" ON "OQS_ENABLE_KEM_NTRU_PLUS" OFF)
endif()
endif()
```


liboqs_KpqC / src / kem / ntru_plus / kem_ntru_plus_kem576.c

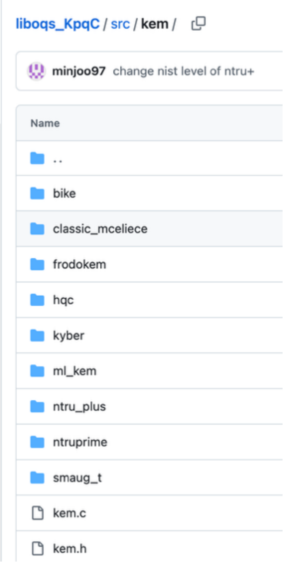
Code Blame 58 lines (40 loc) · 1.91 KB

```

    kem->length_public_key = OQS_KEM_ntru_plus_kem576_length_public_key;
    kem->length_secret_key = OQS_KEM_ntru_plus_kem576_length_secret_key;
    kem->length_ciphertext = OQS_KEM_ntru_plus_kem576_length_ciphertext;
    kem->length_shared_secret = OQS_KEM_ntru_plus_kem576_length_shared_secret;
    kem->length_keypair_seed = OQS_KEM_ntru_plus_kem576_length_keypair_seed;

    kem->keypair = OQS_KEM_ntru_plus_kem576_keypair;
    kem->keypair_derand = OQS_KEM_ntru_plus_kem576_keypair_derand;
    kem->encaps = OQS_KEM_ntru_plus_kem576_encaps;
    kem->decaps = OQS_KEM_ntru_plus_kem576_decaps;

    return kem;
}




OQS_API OQS_STATUS OQS_KEM_ntru_plus_kem576_keypair_derand(uint8_t *public_key, uint8_t *secret_key, const uint8_t *seed) {
        (void)public_key;
        (void)secret_key;
        (void)seed;
        return OQS_ERROR;
}

OQS_API OQS_STATUS OQS_KEM_ntru_plus_kem576_keypair(uint8_t *public_key, uint8_t *secret_key) {
    return (OQS_STATUS) crypto_kem_keypair(public_key, secret_key);
```


liboqs_KpqC / src / kem / kem.h

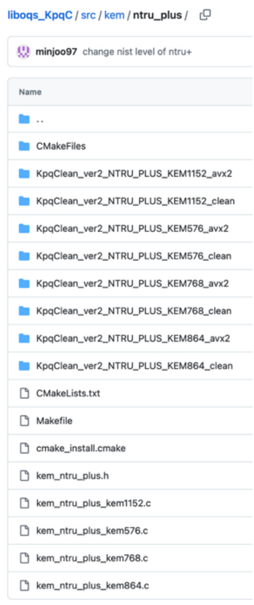
Code Blame 351 lines (319 loc) · 14 KB

```
/** Algorithm identifier for SMAUG-TIMER KEM. */
#define OQS_KEM_alg_smaug_timer "SMAUG-TIMER"
/** Algorithm identifier for SMAUG-T1 KEM. */
#define OQS_KEM_alg_smaug_t1 "SMAUG-T1"
/** Algorithm identifier for SMAUG-T3 KEM. */
#define OQS_KEM_alg_smaug_t3 "SMAUG-T3"
/** Algorithm identifier for SMAUG-T5 KEM. */
#define OQS_KEM_alg_smaug_t5 "SMAUG-T5"
/** Algorithm identifier for NTRU-Plus-KEM-576 KEM. */
#define OQS_KEM_alg_ntru_plus_kem576 "NTRU-Plus-KEM-576"
/** Algorithm identifier for NTRU-Plus-KEM-768 KEM. */
#define OQS_KEM_alg_ntru_plus_kem768 "NTRU-Plus-KEM-768"
/** Algorithm identifier for NTRU-Plus-KEM-864 KEM. */
#define OQS_KEM_alg_ntru_plus_kem864 "NTRU-Plus-KEM-864"
/** Algorithm identifier for NTRU-Plus-KEM-1152 KEM. */
#define OQS_KEM_alg_ntru_plus_kem1152 "NTRU-Plus-KEM-1152"

```


liboqs_KpqC / src / kem /

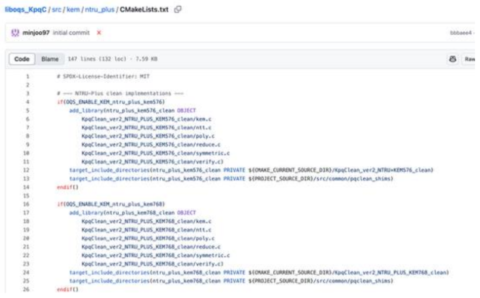
minjoo97 change nist level of ntru+

Name
- ..
- bike
- classic_mceliece
- frodokem
- hqc
- kyber
- ml_kem
- ntru_plus
- ntruprime
- smaug_t
- kem.c
- kem.h


liboqs_KpqC / src / kem / ntru_plus /

minjoo97 change nist level of ntru+

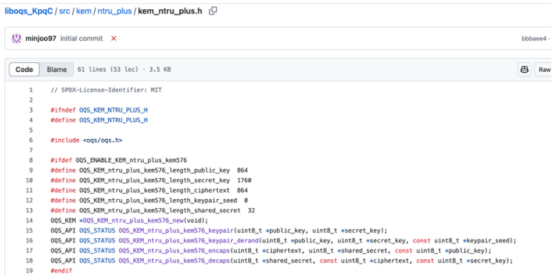
Name
- ..
- CMakeFiles
- KpqClean_ver2_NTRU_PLUS_KEM1152_avx2
- KpqClean_ver2_NTRU_PLUS_KEM1152_clean
- KpqClean_ver2_NTRU_PLUS_KEM576_avx2
- KpqClean_ver2_NTRU_PLUS_KEM576_clean
- KpqClean_ver2_NTRU_PLUS_KEM768_avx2
- KpqClean_ver2_NTRU_PLUS_KEM768_clean
- KpqClean_ver2_NTRU_PLUS_KEM864_avx2
- KpqClean_ver2_NTRU_PLUS_KEM864_clean
- CMakeLists.txt
- Makefile
- cmake_install.cmake
- kem_ntru_plus.h
- kem_ntru_plus_kem1152.c
- kem_ntru_plus_kem576.c
- kem_ntru_plus_kem768.c
- kem_ntru_plus_kem864.c


liboqs_KpqC / src / kem / ntru_plus / CMakeLists.txt

minjoo97 initial commit ✕ bbbaee4

Code Blame 147 lines (132 loc) · 7.59 KB

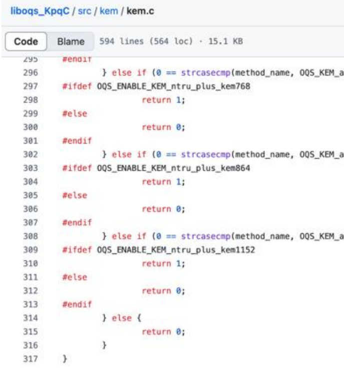
```
# SPDX-License-Identifier: MIT

# ---- NTRU-Plus clean implementations ----
if(OQS_ENABLE_KEM_ntru_plus_kem576)
    add_library(ntru_plus_kem576_clean OBJECT
        KpqClean_ver2_NTRU_PLUS_KEM576_clean/kem.c
        KpqClean_ver2_NTRU_PLUS_KEM576_clean/ntt.c
        KpqClean_ver2_NTRU_PLUS_KEM576_clean/poly.c
        KpqClean_ver2_NTRU_PLUS_KEM576_clean/reduce.c
        KpqClean_ver2_NTRU_PLUS_KEM576_clean/symmetric.c
        KpqClean_ver2_NTRU_PLUS_KEM576_clean/verify.c)
    target_include_directories(ntru_plus_kem576_clean PRIVATE ${CMAKE_CURRENT_SOURCE_DIR}/KpqClean_ver2_NTRU+KEM576_clean)
    target_include_directories(ntru_plus_kem576_clean PRIVATE ${PROJECT_SOURCE_DIR}/src/common/pqclean_shims)
endif()

if(OQS_ENABLE_KEM_ntru_plus_kem768)
    add_library(ntru_plus_kem768_clean OBJECT
        KpqClean_ver2_NTRU_PLUS_KEM768_clean/kem.c
        KpqClean_ver2_NTRU_PLUS_KEM768_clean/ntt.c
        KpqClean_ver2_NTRU_PLUS_KEM768_clean/poly.c
        KpqClean_ver2_NTRU_PLUS_KEM768_clean/reduce.c
        KpqClean_ver2_NTRU_PLUS_KEM768_clean/symmetric.c
        KpqClean_ver2_NTRU_PLUS_KEM768_clean/verify.c)
    target_include_directories(ntru_plus_kem768_clean PRIVATE ${CMAKE_CURRENT_SOURCE_DIR}/KpqClean_ver2_NTRU_PLUS_KEM768_clean)
    target_include_directories(ntru_plus_kem768_clean PRIVATE ${PROJECT_SOURCE_DIR}/src/common/pqclean_shims)
endif()
```


liboqs_KpqC / src / kem / ntru_plus / kem_ntru_plus.h

minjoo97 initial commit ✕ bbbaee4

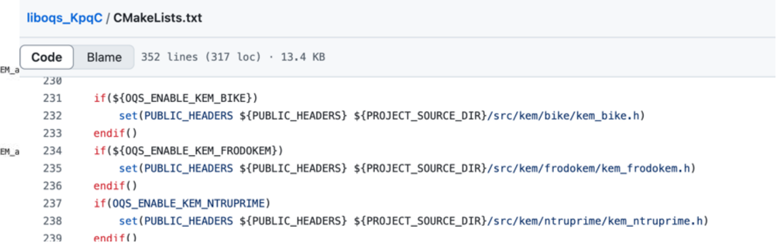
Code Blame 61 lines (53 loc) · 3.5 KB Raw

```
// SPDX-License-Identifier: MIT

#ifndef OQS_KEM_NTRU_PLUS_H
#define OQS_KEM_NTRU_PLUS_H

#include <oqs/oqs.h>

#ifdef OQS_ENABLE_KEM_ntru_plus_kem576
#define OQS_KEM_ntru_plus_kem576_length_public_key  864
#define OQS_KEM_ntru_plus_kem576_length_secret_key  1760
#define OQS_KEM_ntru_plus_kem576_length_ciphertext  864
#define OQS_KEM_ntru_plus_kem576_length_keypair_seed  0
#define OQS_KEM_ntru_plus_kem576_length_shared_secret  32
OQS_KEM *OQS_KEM_ntru_plus_kem576_new(void);
OQS_API OQS_STATUS OQS_KEM_ntru_plus_kem576_keypair(uint8_t *public_key, uint8_t *secret_key);
OQS_API OQS_STATUS OQS_KEM_ntru_plus_kem576_keypair_derand(uint8_t *public_key, uint8_t *secret_key, const uint8_t *keypair_seed);
OQS_API OQS_STATUS OQS_KEM_ntru_plus_kem576_encaps(uint8_t *ciphertext, uint8_t *shared_secret, const uint8_t *public_key);
OQS_API OQS_STATUS OQS_KEM_ntru_plus_kem576_decaps(uint8_t *shared_secret, const uint8_t *ciphertext, const uint8_t *secret_key);
#endif
```


liboqs_KpqC / src / kem / kem.c

Code Blame 594 lines (564 loc) · 15.1 KB

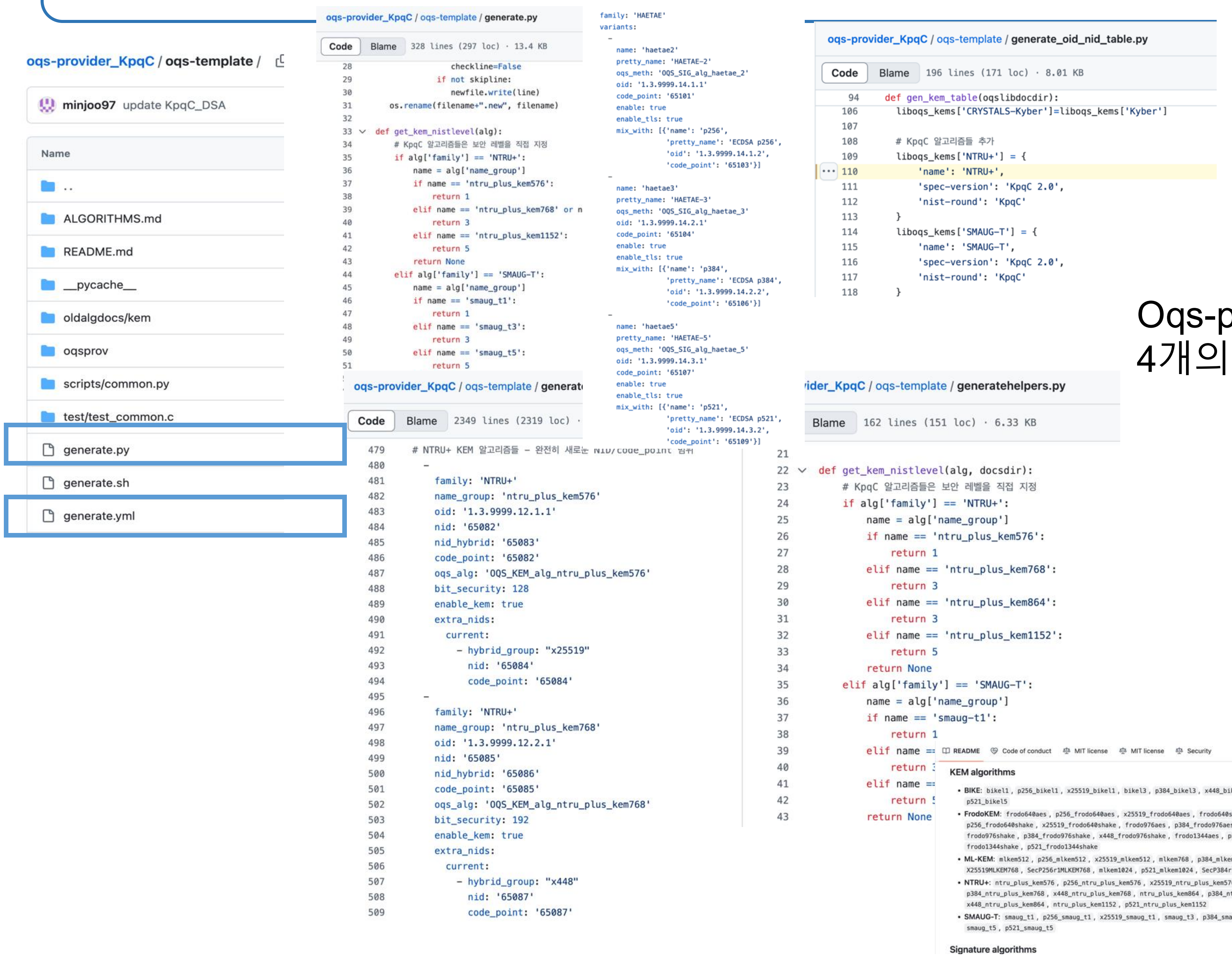
```
#endif
            } else if (0 == strcasecmp(method_name, OQS_KEM_alg_ntru_plus_kem768)) {
#ifdef OQS_ENABLE_KEM_ntru_plus_kem768
                    return 1;
#else
                    return 0;
#endif
            } else if (0 == strcasecmp(method_name, OQS_KEM_a
#ifdef OQS_ENABLE_KEM_ntru_plus_kem864
                    return 1;
#else
                    return 0;
#endif
            } else if (0 == strcasecmp(method_name, OQS_KEM_a
#ifdef OQS_ENABLE_KEM_ntru_plus_kem1152
                    return 1;
#else
                    return 0;
#endif
            } else {
                    return 0;
            }
    }
```


liboqs_KpqC / CMakeLists.txt

Code Blame 352 lines (317 loc) · 13.4 KB

```

if(${OQS_ENABLE_KEM_BIKE})
    set(PUBLIC_HEADERS ${PUBLIC_HEADERS} ${PROJECT_SOURCE_DIR}/src/kem/bike/kem_bike.h)
endif()
if(${OQS_ENABLE_KEM_FRODOKEM})
    set(PUBLIC_HEADERS ${PUBLIC_HEADERS} ${PROJECT_SOURCE_DIR}/src/kem/frodokem/kem_frodokem.h)
endif()
if(OQS_ENABLE_KEM_NTRUPRIME)
    set(PUBLIC_HEADERS ${PUBLIC_HEADERS} ${PROJECT_SOURCE_DIR}/src/kem/ntruprime/kem_ntruprime.h)
endif()
```

oqs-provider_KpqC



oqs-provider_KpqC / oqs-template /

minjoo97 update KpqC_DSA

Name
..
ALGORITHMS.md
README.md
__pycache__
oldalgdocs/kem
oqsprov
scripts/common.py
test/test_common.c
generate.py
generate.sh
generate.yml

oqs-provider_KpqC / oqs-template / generate.py

Code Blame 328 lines (297 loc) · 13.4 KB

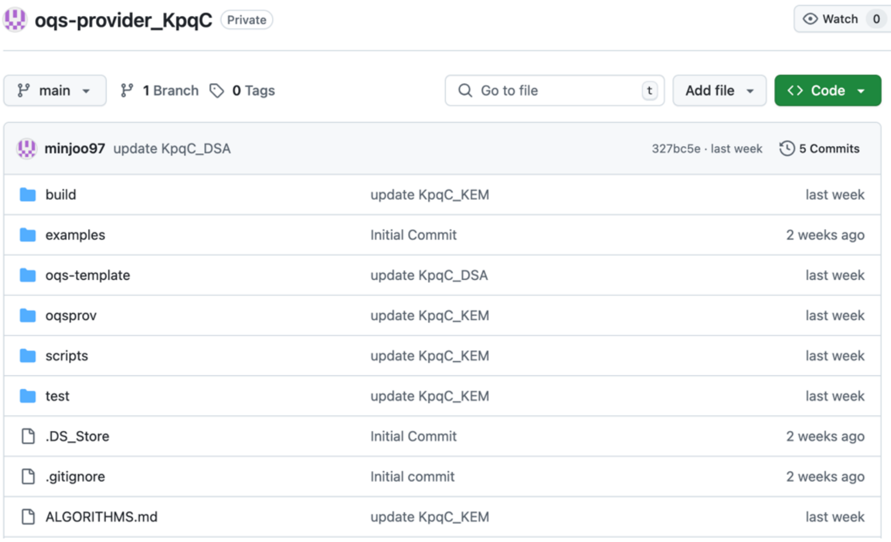
```python
                    checkline=False
                if not skipline:
                    newfile.write(line)
        os.rename(filename+".new", filename)

def get_kem_nistlevel(alg):
    # KpqC 알고리즘들은 보안 레벨을 직접 지정
    if alg['family'] == 'NTRU+':
        name = alg['name_group']
        if name == 'ntru_plus_kem576':
            return 1
        elif name == 'ntru_plus_kem768' or n
            return 3
        elif name == 'ntru_plus_kem1152':
            return 5
        return None
    elif alg['family'] == 'SMAUG-T':
        name = alg['name_group']
        if name == 'smaug_t1':
            return 1
        elif name == 'smaug_t3':
            return 3
        elif name == 'smaug_t5':
            return 5
```

```yaml
family: 'HAETAE'
variants:
  -
    name: 'haetae2'
    pretty_name: 'HAETAE-2'
    oqs_meth: 'OQS_SIG_alg_haetae_2'
    oid: '1.3.9999.14.1.1'
    code_point: '65101'
    enable: true
    enable_tls: true
    mix_with: [{'name': 'p256',
                'pretty_name': 'ECDSA p256',
                'oid': '1.3.9999.14.1.2',
                'code_point': '65103'}]
  -
    name: 'haetae3'
    pretty_name: 'HAETAE-3'
    oqs_meth: 'OQS_SIG_alg_haetae_3'
    oid: '1.3.9999.14.2.1'
    code_point: '65104'
    enable: true
    enable_tls: true
    mix_with: [{'name': 'p384',
                'pretty_name': 'ECDSA p384',
                'oid': '1.3.9999.14.2.2',
                'code_point': '65106'}]
  -
    name: 'haetae5'
    pretty_name: 'HAETAE-5'
    oqs_meth: 'OQS_SIG_alg_haetae_5'
    oid: '1.3.9999.14.3.1'
    code_point: '65107'
    enable: true
    enable_tls: true
    mix_with: [{'name': 'p521',
                'pretty_name': 'ECDSA p521',
                'oid': '1.3.9999.14.3.2',
                'code_point': '65109'}]
```

oqs-provider_KpqC / oqs-template / generate_oid_nid_table.py

Code Blame 196 lines (171 loc) · 8.01 KB

```python
def gen_kem_table(oqslibdocdir):
    liboqs_kems['CRYSTALS-Kyber']=liboqs_kems['Kyber']

    # KpqC 알고리즘들 추가
    liboqs_kems['NTRU+'] = {
        'name': 'NTRU+',
        'spec-version': 'KpqC 2.0',
        'nist-round': 'KpqC'
    }
    liboqs_kems['SMAUG-T'] = {
        'name': 'SMAUG-T',
        'spec-version': 'KpqC 2.0',
        'nist-round': 'KpqC'
    }
```



oqs-provider_KpqC Private

minjoo97 update KpqC_DSA 327bc5e · last week 5 Commits

build	update KpqC_KEM	last week
examples	Initial Commit	2 weeks ago
oqs-template	update KpqC_DSA	last week
oqsprov	update KpqC_KEM	last week
scripts	update KpqC_KEM	last week
test	update KpqC_KEM	last week
.DS_Store	Initial Commit	2 weeks ago
.gitignore	Initial commit	2 weeks ago
ALGORITHMS.md	update KpqC_KEM	last week

Oqs-provider_KpqC의 경우
4개의 파일만 수정하고 cmake 실행하면 됨

oqs-provider_KpqC / oqs-template / generat

Code Blame 2349 lines (2319 loc) ·

```yaml
# NTRU+ KEM 알고리즘들 - 완전히 새로운 NID/code_point 범위
  -
    family: 'NTRU+'
    name_group: 'ntru_plus_kem576'
    oid: '1.3.9999.12.1.1'
    nid: '65082'
    nid_hybrid: '65083'
    code_point: '65082'
    oqs_alg: 'OQS_KEM_alg_ntru_plus_kem576'
    bit_security: 128
    enable_kem: true
    extra_nids:
      current:
        - hybrid_group: "x25519"
          nid: '65084'
          code_point: '65084'
  -
    family: 'NTRU+'
    name_group: 'ntru_plus_kem768'
    oid: '1.3.9999.12.2.1'
    nid: '65085'
    nid_hybrid: '65086'
    code_point: '65085'
    oqs_alg: 'OQS_KEM_alg_ntru_plus_kem768'
    bit_security: 192
    enable_kem: true
    extra_nids:
      current:
        - hybrid_group: "x448"
          nid: '65087'
          code_point: '65087'
```

ider_KpqC / oqs-template / generatehelpers.py

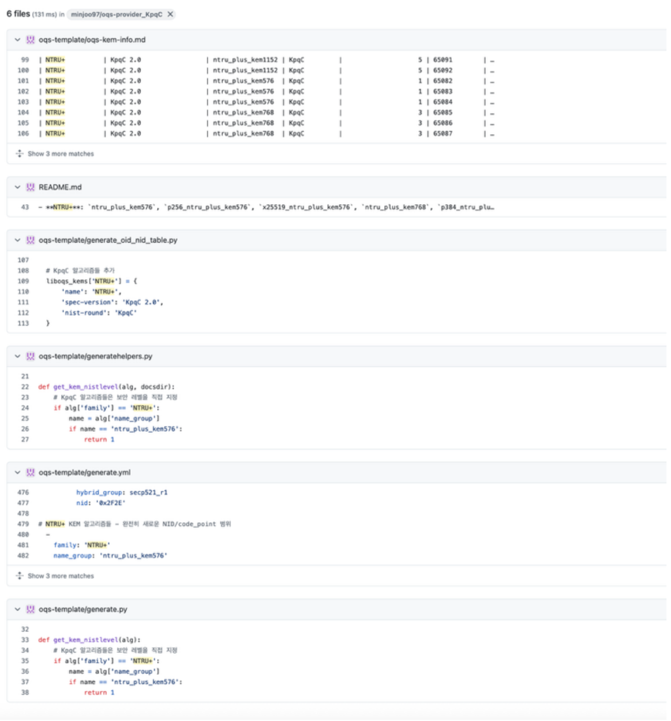
Blame 162 lines (151 loc) · 6.33 KB

```python

def get_kem_nistlevel(alg, docsdir):
    # KpqC 알고리즘들은 보안 레벨을 직접 지정
    if alg['family'] == 'NTRU+':
        name = alg['name_group']
        if name == 'ntru_plus_kem576':
            return 1
        elif name == 'ntru_plus_kem768':
            return 3
        elif name == 'ntru_plus_kem864':
            return 3
        elif name == 'ntru_plus_kem1152':
            return 5
        return None
    elif alg['family'] == 'SMAUG-T':
        name = alg['name_group']
        if name == 'smaug-t1':
            return 1
        elif name ==
            return 3
        elif name ==
            return 5
        return None
```

KEM algorithms

- BIKE: bikel1, p256_bikel1, x25519_bikel1, bikel3, p384_bikel3, x448_bikel3, bikel5, p521_bikel5
- FrodoKEM: frodo640aes, p256_frodo640aes, x25519_frodo640aes, frodo640shake, p256_frodo640shake, x25519_frodo640shake, frodo976aes, p384_frodo976aes, x448_frodo976aes, frodo976shake, p384_frodo976shake, x448_frodo976shake, frodo1344aes, p521_frodo1344aes, frodo1344shake, p521_frodo1344shake
- ML-KEM: mlkem512, p256_mlkem512, x25519_mlkem512, mlkem768, p384_mlkem768, x448_mlkem768, X25519MLKEM768, SecP256r1MLKEM768, mlkem1024, p521_mlkem1024, SecP384r1MLKEM1024
- NTRU+: ntru_plus_kem576, p256_ntru_plus_kem576, x25519_ntru_plus_kem576, ntru_plus_kem768, p384_ntru_plus_kem768, x448_ntru_plus_kem768, ntru_plus_kem864, p384_ntru_plus_kem864, x448_ntru_plus_kem864, ntru_plus_kem1152, p521_ntru_plus_kem1152
- SMAUG-T: smaug_t1, p256_smaug_t1, x25519_smaug_t1, smaug_t3, p384_smaug_t3, x448_smaug_t3, smaug_t5, p521_smaug_t5

Signature algorithms



6 files (131 ms) in minjoo97/oqs-provider_KpqC

oqs-template/oqs-kem-info.md

```
| NTRU+          | KpqC 2.0           | ntru_plus_kem1152 | KpqC        |                    5 | 65091       | –
| NTRU+          | KpqC 2.0           | ntru_plus_kem1152 | KpqC        |                    5 | 65092       | –
| NTRU+          | KpqC 2.0           | ntru_plus_kem576  | KpqC        |                    1 | 65082       | –
| NTRU+          | KpqC 2.0           | ntru_plus_kem576  | KpqC        |                    1 | 65083       | –
| NTRU+          | KpqC 2.0           | ntru_plus_kem576  | KpqC        |                    1 | 65084       | –
| NTRU+          | KpqC 2.0           | ntru_plus_kem768  | KpqC        |                    3 | 65085       | –
| NTRU+          | KpqC 2.0           | ntru_plus_kem768  | KpqC        |                    3 | 65086       | –
| NTRU+          | KpqC 2.0           | ntru_plus_kem768  | KpqC        |                    3 | 65087       | –
```

Show 3 more matches

README.md

```
- **NTRU+**: `ntru_plus_kem576`, `p256_ntru_plus_kem576`, `x25519_ntru_plus_kem576`, `ntru_plus_kem768`, `p384_ntru_plu…
```

oqs-template/generate_oid_nid_table.py

```python

    # KpqC 알고리즘들 추가
    liboqs_kems['NTRU+'] = {
        'name': 'NTRU+',
        'spec-version': 'KpqC 2.0',
        'nist-round': 'KpqC'
    }
```

oqs-template/generatehelpers.py

```python

def get_kem_nistlevel(alg, docsdir):
    # KpqC 알고리즘들은 보안 레벨을 직접 지정
    if alg['family'] == 'NTRU+':
        name = alg['name_group']
        if name == 'ntru_plus_kem576':
            return 1
```

oqs-template/generate.yml

```yaml
          hybrid_group: secp521_r1
          nid: '0x2F2E'

# NTRU+ KEM 알고리즘들 - 완전히 새로운 NID/code_point 범위
  -
    family: 'NTRU+'
    name_group: 'ntru_plus_kem576'
```

Show 3 more matches

oqs-template/generate.py

```python

def get_kem_nistlevel(alg):
    # KpqC 알고리즘들은 보안 레벨을 직접 지정
    if alg['family'] == 'NTRU+':
        name = alg['name_group']
        if name == 'ntru_plus_kem576':
            return 1
```

사전 작업

-GNinja : Ninja 빌드 시스템 생성
-DCMAKE_INSTALL_PREFIX : make install 시 설치 경로 지정
-DCMAKE_OSX_ARCHITECTURES=arm64 : Apple Silicon 전용 컴파일
-S . -B build : 현재 소스 → build/ 폴더로 빌드 파일 생성

1. 필요 패키지 설치

- brew install cmake ninja openssl@3 git bc

2. Liboqs_KpqC 설치

```
cd ~
git clone --recursive https://github.com/minjoo97/liboqs_KpqC.git
cd liboqs_KpqC
cmake -GNinja \
    -DCMAKE_INSTALL_PREFIX=/usr/local \
    -DCMAKE_OSX_ARCHITECTURES=arm64 \
    -S . -B build
cmake --build build
sudo cmake --install build
```

사전 작업

-DCMAKE_BUILD_TYPE=Release : 최적화 빌드
-DOPENSSL_ROOT_DIR=… : Homebrew OpenSSL 위치 지정
-Dliboqs_DIR=… : liboqs CMake 설정 파일 위치 지정
-DBUILD_SHARED_LIBS=ON : .dylib 형태로 빌드

3. oqs-provider_KpqC 설치

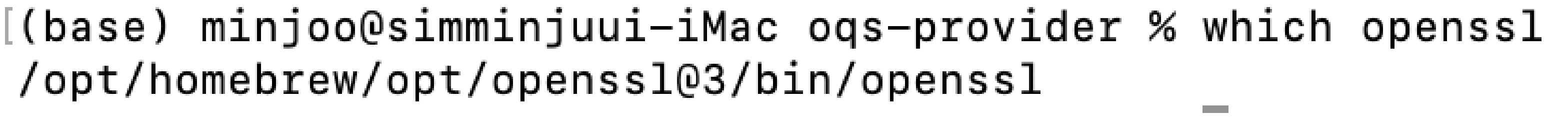
```
cd ~
rm -rf oqs-provider
git clone https://github.com/minjoo97/oqs-provider_KpqC.git
cd oqs-provider_KpqC
cmake -GNinja \
    -DCMAKE_BUILD_TYPE=Release \
    -DOPENSSL_ROOT_DIR=$(brew --prefix openssl@3) \
    -Dliboqs_DIR=/usr/local/lib/cmake/liboqs \
    -DCMAKE_OSX_ARCHITECTURES=arm64 \
    -DBUILD_SHARED_LIBS=ON \
    -S . -B build
cmake --build build
```

Case 1) 기존 인증서 사용+ 하이브리드 TLS

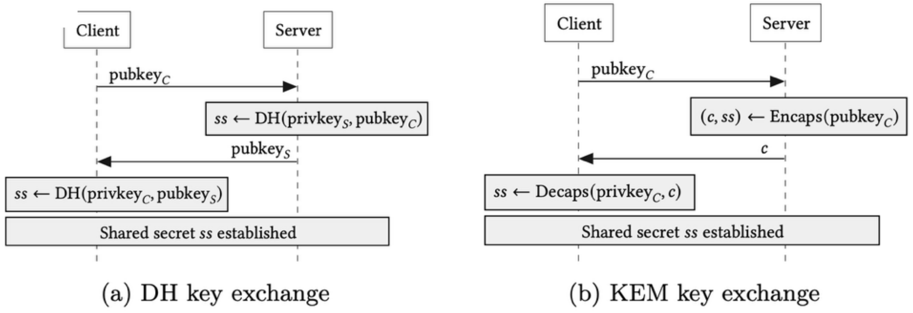
- P-256+KpqC(SMAUG-T, NTRU+) 키 교환(TLS 1.3)

1) 쉘 환경 설정

```
# (1) Homebrew OpenSSL 3.x 우선 호출
export PATH="$(brew --prefix openssl@3)/bin:$PATH"

# (2) oqs-provider 모듈 위치 지정(OpenSSL이 외부 provider 모듈을 찾는 경로 지정)
export OPENSSL_MODULES="$HOME/oqs-provider_KpqC/build/lib"
```



```
[(base) minjoo@simminjuui-iMac oqs-provider % which openssl
/opt/homebrew/opt/openssl@3/bin/openssl
```



Client
Server
pubkey_C
$ss \leftarrow \mathrm{DH}(\mathrm{privkey}_S, \mathrm{pubkey}_C)$
pubkey_S
$ss \leftarrow \mathrm{DH}(\mathrm{privkey}_C, \mathrm{pubkey}_S)$
Shared secret ss established
Client
Server
pubkey_C
$(c, ss) \leftarrow \mathrm{Encaps}(\mathrm{pubkey}_C)$
c
$ss \leftarrow \mathrm{Decaps}(\mathrm{privkey}_C, c)$
Shared secret ss established

(a) DH key exchange

(b) KEM key exchange

Figure 2: Key exchange diagrams

ALNAHAWI, Nouri, et al. A comprehensive survey on post-quantum tls. *IACR Communications in Cryptology*, 2024

Case 1) 기존 인증서 사용+ 하이브리드 TLS

2) 서버용 인증서 및 키 준비

```
cd ~
# ECDSA P-256 키·인증서 생성
openssl ecparam -name prime256v1 -genkey -noout -out server.key
openssl req -x509 -key server.key -out server.crt -nodes \
  -subj "/CN=localhost"
```

ecparam -name prime256v1 :
P-256(ECDSA) 파라미터 사용
-genkey -noout : 키만 생성(파라미터 출력 생략)
req -x509 : self-signed X.509 인증서 생성
-nodes : 키 암호화 없이 저장
-subj "/CN=localhost" : 인증서 Subject 설정

3) 벤치마크용 스크립트 생성

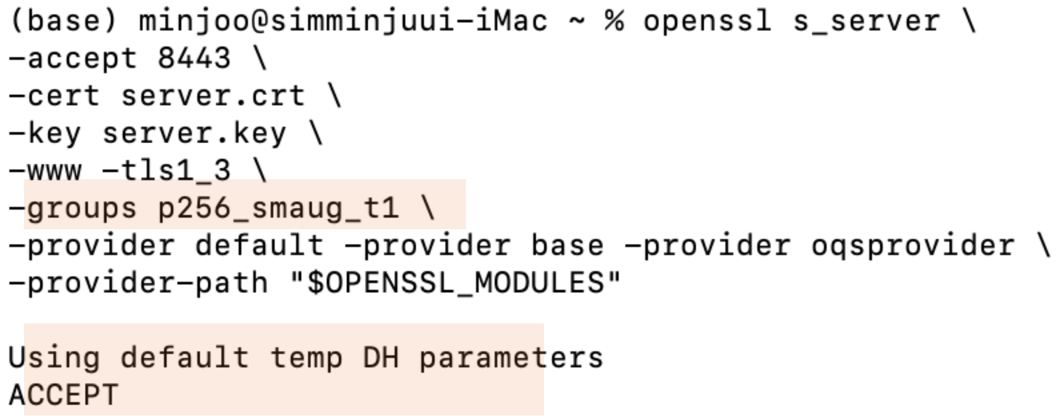
```
cd ~
cat << 'EOF' > bench.sh
#!/usr/bin/env bash
HOST=localhost
PORT=8443
N=200          # 반복 횟수 (원하는 만큼 조절 가능)
PROV_PATH="$HOME/oqs-provider/build/lib"

# 시작 시간
START=$(date +%s.%N)
for i in $(seq 1 $N); do
  printf '' | openssl s_client \
    -connect ${HOST}:${PORT} \
    -tls1_3 \
    -groups p256_smaug_t1 \
    -provider default -provider base -provider oqsprovider \
    -provider-path "${PROV_PATH}" \
    > /dev/null 2>&1
done
END=$(date +%s.%N)

# 결과 계산
ELAPSED=$(echo "$END - $START" | bc)
echo "→ $N handshakes in ${ELAPSED}s"
echo "→ Avg handshake time: $(echo "$ELAPSED / $N" | bc -l)s"
echo "→ Throughput: $(echo "$N / $ELAPSED" | bc -l) handshakes/sec"
EOF

chmod +x bench.sh
```

Case 1) 기존 인증서 사용+ 하이브리드 TLS

4) TLS 서버 띄우기(서버용 새로운 터미널)

쉘 환경 설정(앞과 동일, 새로운 터미널 켜면 무조건 수행)

```
    # (1) Homebrew OpenSSL 3.x 우선 호출
    export PATH="$(brew --prefix openssl@3)/bin:$PATH"

    # (2) oqs-provider 모듈 위치 지정
    export OPENSSL_MODULES="$HOME/oqs-provider_KpqC/build/lib"

    # (3) TLS 서버 터미널 동작
openssl s_server \
-accept 8443 \
-cert server.crt \
-key server.key \
-www -tls1_3 \
-groups p256_smaug_t1 \
-provider default -provider base -provider oqsprovider \
-provider-path "$OPENSSL_MODULES"
```



```
(base) minjoo@simminjuui-iMac ~ % openssl s_server \
-accept 8443 \
-cert server.crt \
-key server.key \
-www -tls1_3 \
-groups p256_smaug_t1 \
-provider default -provider base -provider oqsprovider \
-provider-path "$OPENSSL_MODULES"

Using default temp DH parameters
ACCEPT
```

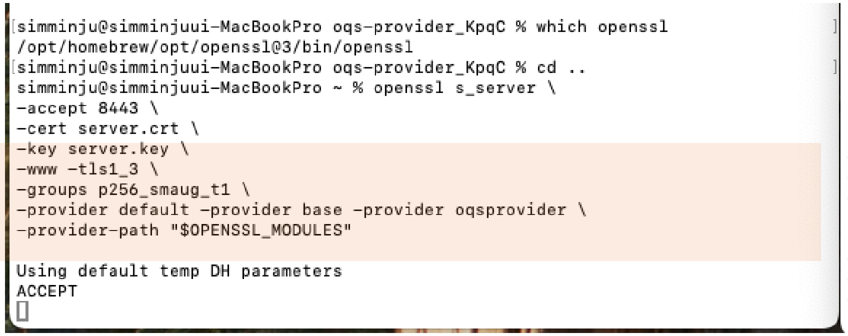
Case 1) 기존 인증서 사용+ 하이브리드 TLS

5) Handshake 벤치마크 실행(클라이언트 터미널_기존 터미널)

./bench.sh

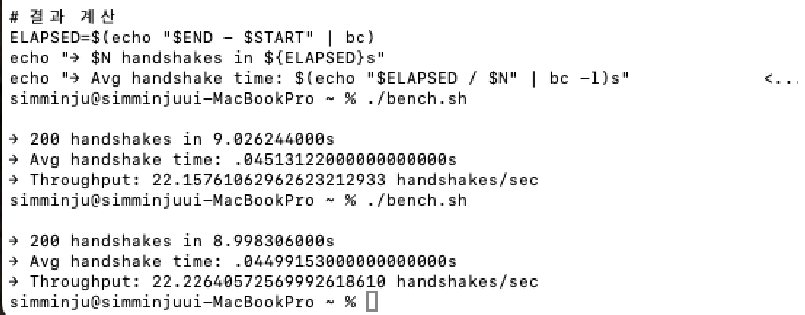
ECDSA 키·인증서 생성

P-256+SMAUG_T1 키 교환(TLS 1.3)



```
simminju@simminjuui-MacBookPro oqs-provider_KpqC % which openssl
/opt/homebrew/opt/openssl@3/bin/openssl
simminju@simminjuui-MacBookPro oqs-provider_KpqC % cd ..
simminju@simminjuui-MacBookPro ~ % openssl s_server \
-accept 8443 \
-cert server.crt \
-key server.key \
-www -tls1_3 \
-groups p256_smaug_t1 \
-provider default -provider base -provider oqsprovider \
-provider-path "$OPENSSL_MODULES"

Using default temp DH parameters
ACCEPT
```

서버



```
# 결과 계산
ELAPSED=$(echo "$END - $START" | bc)
echo "→ $N handshakes in ${ELAPSED}s"
echo "→ Avg handshake time: $(echo "$ELAPSED / $N" | bc -l)s"            <...
simminju@simminjuui-MacBookPro ~ % ./bench.sh

→ 200 handshakes in 9.026244000s
→ Avg handshake time: .04513122000000000000s
→ Throughput: 22.15761062962623212933 handshakes/sec
simminju@simminjuui-MacBookPro ~ % ./bench.sh

→ 200 handshakes in 8.998306000s
→ Avg handshake time: .04499153000000000000s
→ Throughput: 22.22640572569992618610 handshakes/sec
simminju@simminjuui-MacBookPro ~ %
```

클라이언트

Case 2) PQC 인증서 + 하이브리드 TLS

Case1의 쉘환경 설정 동일하게 수행

0) 실험용 디렉터리 생성 및 이동

```
mkdir -p ~/tls-test_KpqC
cd ~/tls-test_KpqC
```

1) PQC 인증서와 키 생성(haetae2)

2)TLS 서버 실행

```
openssl s_server \
-accept 8443 \
-cert server.crt \
-key server.key \
-www -tls1_3 \
-groups p256_smaug_t1 \
-provider default -provider base -provider oqsprovider \
-provider-path "$OPENSSL_MODULES"
```

-accept 8443 : 8443 포트 대기

-groups p256_smaug_t1 : P-256 + smaug_t1하이브리드 KEM

oqsprovider 로딩으로 smaug_t1·haetae2 동작

```
# ① 개인키(server.key) 생성
openssl genpkey \
  -provider default \
  -provider base \
  -provider oqsprovider \
  -algorithm haetae2\
  -out server.key

# ② Self-signed 인증서(server.crt) 발급 (유효기간 1년)
openssl req -new -x509 \
  -provider default \
  -provider base \
  -provider oqsprovider \
  -key server.key \
  -out server.crt \
  -days 365 -nodes \
  -subj "/C=KR/ST=Seoul/L=Seoul/O=MyOrg/OU=IT/CN=localhost"
```

Case 2) PQC 인증서 + 하이브리드 TLS

[클라이언트 터미널]

3) 핸드쉐이크 기능 검증

```
cd ~/ tls-test_KpqC

openssl s_client \
  -connect localhost:8443 \
  -tls1_3 \
  -groups p256_smaug_t1 \
  -provider default \
  -provider base \
  -provider oqsprovider \
  -CAfile server.crt \
  -msg
```

-groups p256_smuag_t1 : ClientHello 에 P-256+smaug_t1 제안 (키교환만 하이브리드)
-CAfile server.crt : Self-signed 인증서 신뢰
-msg : 메시지 레벨 로그 출력

Case 2) PQC 인증서 + 하이브리드 TLS

- 10초 동안 몇번 핸드쉐이크가 돌아가는지 확인

```
cat > ~/tls-test_KpqC/time_bench.sh << 'EOF'
#!/usr/bin/env bash
HOST=localhost
PORT=8443
DURATION=10                    # 측정 시간(초)
PROV_PATH="$HOME/ oqs-provider_KpqC/build/lib"

# 환경 체크
export PATH="$(brew --prefix openssl@3)/bin:$PATH"
export OPENSSL_MODULES="$PROV_PATH"

START_TS=$(date +%s)
END_TS=$((START_TS + DURATION))
COUNT=0

while [ "$(date +%s)" -lt "$END_TS" ]; do
  # 빈 줄 입력 → 핸드셰이크. 출력 모두 버림
  printf '' | openssl s_client \
     -connect ${HOST}:${PORT} \
     -tls1_3 \
     -groups p256_smaug_t1 \
     -provider default -provider base -provider oqsprovider \
     -provider-path "${PROV_PATH}" \
     -CAfile server.crt \
     > /dev/null 2>&1
  COUNT=$((COUNT + 1))
done

echo "→ $COUNT handshakes in ${DURATION}s"
printf "→ %.2f handshakes/sec\n" "$(bc -l <<< "$COUNT / $DURATION")"
printf "→ Avg handshake time: %.4fs\n" "$(bc -l <<< "$DURATION /
$COUNT")"
EOF

chmod +x ~/tls-test_KpqC /time-bench.sh
```

Case 3) PQC 인증서 + PQC TLS

0. Case1의 쉘환경 설정 동일하게 수행

Cas2의 실험용 디렉터리로이동

1. PQC 서명용 개인키(haetae2 생성) 및 인증서 생성

```
openssl genpkey \
  -provider default \
  -provider base \
  -provider oqsprovider \
  -algorithm haetae2 \
  -out server.key
```

```
openssl req -new -x509 \
  -provider default \
  -provider base \
  -provider oqsprovider \
  -key server.key \
  -out server.crt \
  -days 365 \
  -subj "/CN=localhost"
```

Case 3) PQC 인증서 + PQC TLS

2. 서버 실행(순수 ML-smaug_t1-only)

```
 openssl s_server \
-accept 8443 \
-cert server.crt \
  -key server.key \
  -tls1_3 \
  -groups smaug_t1 \
  -provider default \
  -provider base \
  -provider oqsprovider \
  -provider-path $PROV_PATH \
  -www
```

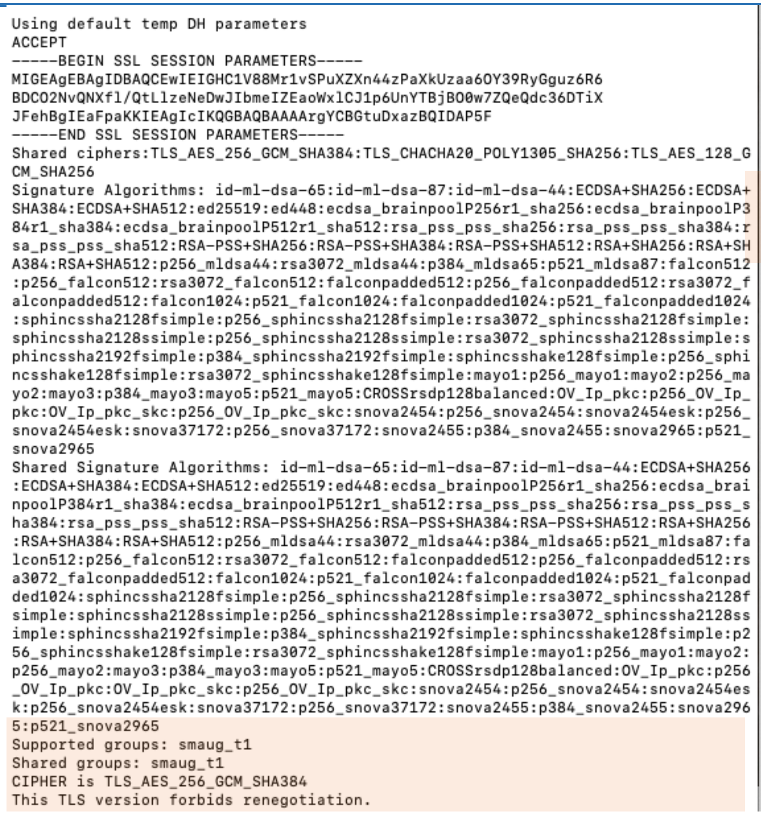
키 교환(KEM): smaug_t1
인증서 서명: haetae2

Case 3) PQC 인증서 + PQC TLS

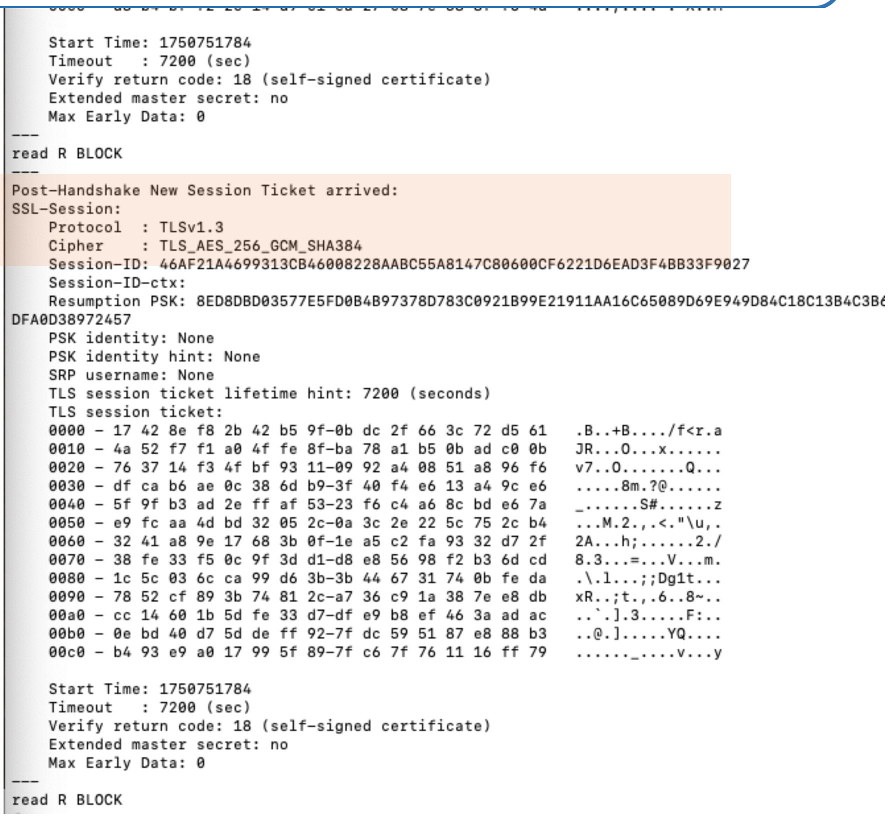
[서버]

3. 단일 핸드쉐이크

```
openssl s_client \
  -connect localhost:8443 \
  -tls1_3 \
  -groups smaug_t1 \
  -provider default \
  -provider base \
  -provider oqsprovider \
  -provider-path "$OPENSSL_MODULES"
```



```
Using default temp DH parameters
ACCEPT
-----BEGIN SSL SESSION PARAMETERS-----
MIGEAgEBAgIDBAQCEwIEIGHC1V88Mr1vSPuXZXn44zPaXkUzaa6OY39RyGguz6R6
BDCO2NvQNXfl/QtLlzeNeDwJIbmeIZEaoWxlCJ1p6UnYTBjBO0w7ZQeQdc36DTiX
JFehBgIEaFpaKKIEAgIcIKQGBAQBAAAArgYCBGtuDxazBQIDAP5F
-----END SSL SESSION PARAMETERS-----
Shared ciphers:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_G
CM_SHA256
Signature Algorithms: id-ml-dsa-65:id-ml-dsa-87:id-ml-dsa-44:ECDSA+SHA256:ECDSA+
SHA384:ECDSA+SHA512:ed25519:ed448:ecdsa_brainpoolP256r1_sha256:ecdsa_brainpoolP3
84r1_sha384:ecdsa_brainpoolP512r1_sha512:rsa_pss_pss_sha256:rsa_pss_pss_sha384:r
sa_pss_pss_sha512:RSA-PSS+SHA256:RSA-PSS+SHA384:RSA-PSS+SHA512:RSA+SHA256:RSA+SH
A384:RSA+SHA512:p256_mldsa44:rsa3072_mldsa44:p384_mldsa65:p521_mldsa87:falcon512
:p256_falcon512:rsa3072_falcon512:falconpadded512:p256_falconpadded512:rsa3072_f
alconpadded512:falcon1024:p521_falcon1024:falconpadded1024:p521_falconpadded1024
:sphincssha2128fsimple:p256_sphincssha2128fsimple:rsa3072_sphincssha2128fsimple:
sphincssha2128ssimple:p256_sphincssha2128ssimple:rsa3072_sphincssha2128ssimple:s
phincssha2192fsimple:p384_sphincssha2192fsimple:sphincsshake128fsimple:p256_sphi
ncsshake128fsimple:rsa3072_sphincsshake128fsimple:mayo1:p256_mayo1:mayo2:p256_ma
yo2:mayo3:p384_mayo3:mayo5:p521_mayo5:CROSSrsdp128balanced:OV_Ip_pkc:p256_OV_Ip_
pkc:OV_Ip_pkc_skc:p256_OV_Ip_pkc_skc:snova2454:p256_snova2454:snova2454esk:p256_
snova2454esk:snova37172:p256_snova37172:snova2455:p384_snova2455:snova2965:p521_
snova2965
Shared Signature Algorithms: id-ml-dsa-65:id-ml-dsa-87:id-ml-dsa-44:ECDSA+SHA256
:ECDSA+SHA384:ECDSA+SHA512:ed25519:ed448:ecdsa_brainpoolP256r1_sha256:ecdsa_brai
npoolP384r1_sha384:ecdsa_brainpoolP512r1_sha512:rsa_pss_pss_sha256:rsa_pss_pss_s
ha384:rsa_pss_pss_sha512:RSA-PSS+SHA256:RSA-PSS+SHA384:RSA-PSS+SHA512:RSA+SHA256
:RSA+SHA384:RSA+SHA512:p256_mldsa44:rsa3072_mldsa44:p384_mldsa65:p521_mldsa87:fa
lcon512:p256_falcon512:rsa3072_falcon512:falconpadded512:p256_falconpadded512:rs
a3072_falconpadded512:falcon1024:p521_falcon1024:falconpadded1024:p521_falconpad
ded1024:sphincssha2128fsimple:p256_sphincssha2128fsimple:rsa3072_sphincssha2128f
simple:sphincssha2128ssimple:p256_sphincssha2128ssimple:rsa3072_sphincssha2128ss
imple:sphincssha2192fsimple:p384_sphincssha2192fsimple:sphincsshake128fsimple:p2
56_sphincsshake128fsimple:rsa3072_sphincsshake128fsimple:mayo1:p256_mayo1:mayo2:
p256_mayo2:mayo3:p384_mayo3:mayo5:p521_mayo5:CROSSrsdp128balanced:OV_Ip_pkc:p256
_OV_Ip_pkc:OV_Ip_pkc_skc:p256_OV_Ip_pkc_skc:snova2454:p256_snova2454:snova2454es
k:p256_snova2454esk:snova37172:p256_snova37172:snova2455:p384_snova2455:snova296
5:p521_snova2965
Supported groups: smaug_t1
Shared groups: smaug_t1
CIPHER is TLS_AES_256_GCM_SHA384
This TLS version forbids renegotiation.
```

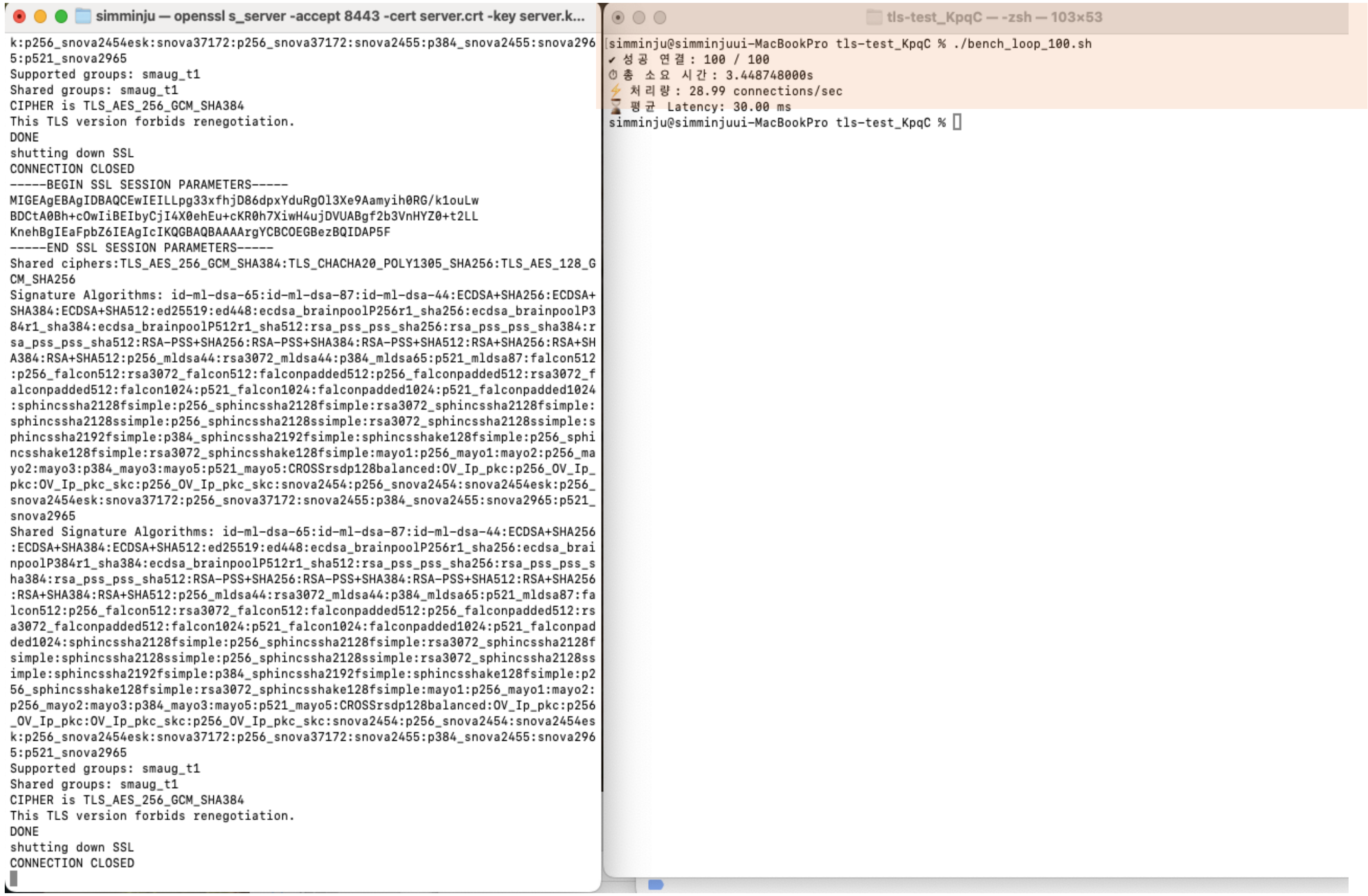


```
    Start Time: 1750751784
    Timeout   : 7200 (sec)
    Verify return code: 18 (self-signed certificate)
    Extended master secret: no
    Max Early Data: 0
---
read R BLOCK
---
Post-Handshake New Session Ticket arrived:
SSL-Session:
    Protocol  : TLSv1.3
    Cipher    : TLS_AES_256_GCM_SHA384
    Session-ID: 46AF21A4699313CB46008228AABC55A8147C80600CF6221D6EAD3F4BB33F9027
    Session-ID-ctx:
    Resumption PSK: 8ED8DBD03577E5FD0B4B97378D783C0921B99E21911AA16C65089D69E949D84C18C13B4C3B6
DFA0D38972457
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 7200 (seconds)
    TLS session ticket:
    0000 - 17 42 8e f8 2b 42 b5 9f-0b dc 2f 66 3c 72 d5 61   .B..+B..../f<r.a
    0010 - 4a 52 f7 f1 a0 4f fe 8f-ba 78 a1 b5 0b ad c0 0b   JR...O...x......
    0020 - 76 37 14 f3 4f bf 93 11-09 92 a4 08 51 a8 96 f6   v7..O.......Q...
    0030 - df ca b6 ae 0c 38 6d b9-3f 40 f4 e6 13 a4 9c e6   .....8m.?@......
    0040 - 5f 9f b3 ad 2e ff af 53-23 f6 c4 a6 8c bd e6 7a   _......S#......z
    0050 - e9 fc aa 4d bd 32 05 2c-0a 3c 2e 22 5c 75 2c b4   ...M.2.,.<."\u,.
    0060 - 32 41 a8 9e 17 68 3b 0f-1e a5 c2 fa 93 32 d7 2f   2A...h;......2./
    0070 - 38 fe 33 f5 0c 9f 3d d1-d8 e8 56 98 f2 b3 6d cd   8.3...=...V...m.
    0080 - 1c 5c 03 6c ca 99 d6 3b-3b 44 67 31 74 0b fe da   .\.l...;;Dg1t...
    0090 - 78 52 cf 89 3b 74 81 2c-a7 36 c9 1a 38 7e e8 db   xR..;t.,.6..8~..
    00a0 - cc 14 60 1b 5d fe 33 d7-df e9 b8 ef 46 3a ad ac   ..`.].3.....F:..
    00b0 - 0e bd 40 d7 5d de ff 92-7f dc 59 51 87 e8 88 b3   ..@.].....YQ....
    00c0 - b4 93 e9 a0 17 99 5f 89-7f c6 7f 76 11 16 ff 79   ......_....v...y

    Start Time: 1750751784
    Timeout   : 7200 (sec)
    Verify return code: 18 (self-signed certificate)
    Extended master secret: no
    Max Early Data: 0
---
read R BLOCK
```

Case 3) PQC 인증서 + PQC TLS

[서버]

4. 100회 핸드쉐이크

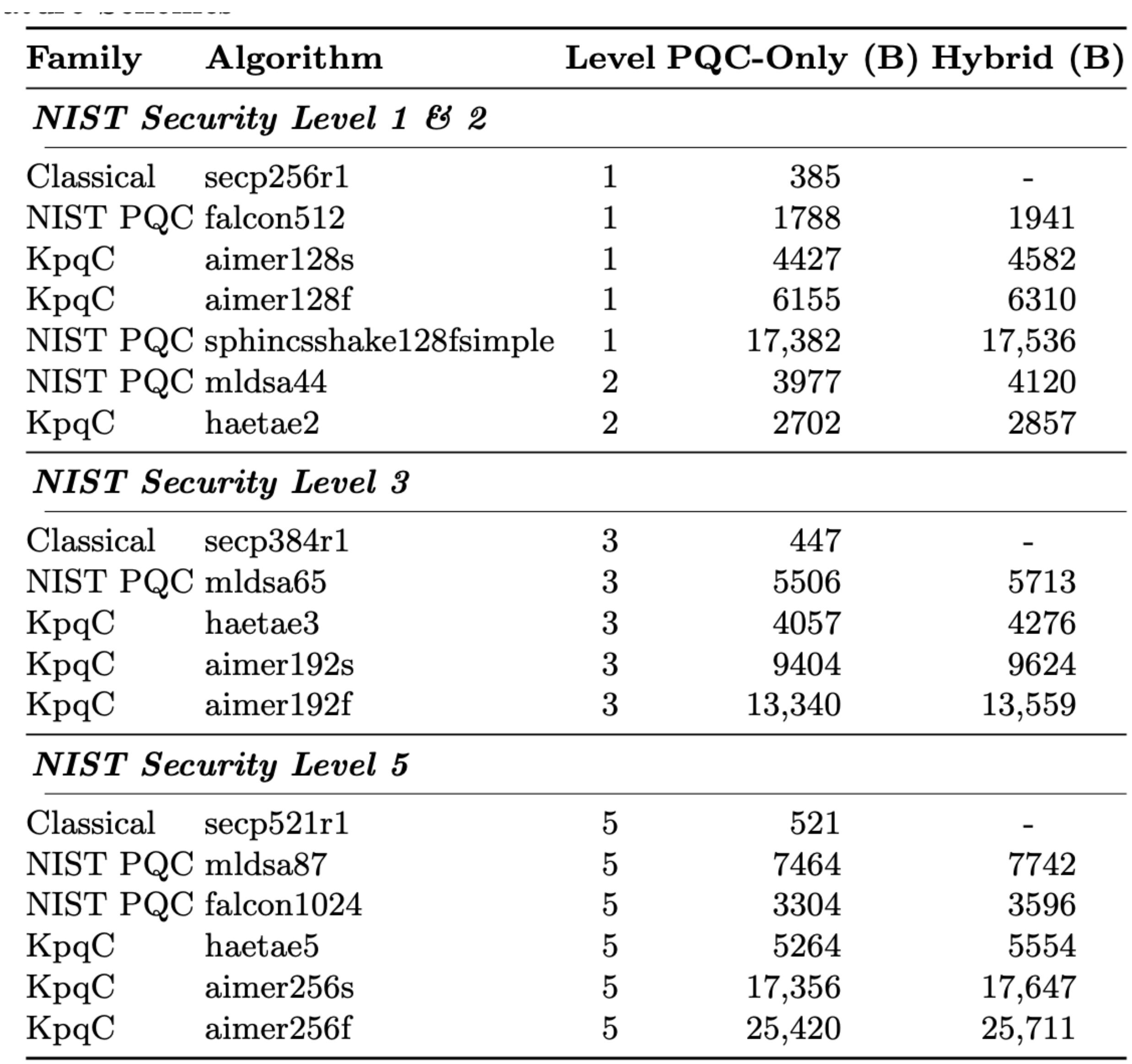

simminju — openssl s_server -accept 8443 -cert server.crt -key server.k...
tls-test_KpqC — -zsh — 103×53

```
k:p256_snova2454esk:snova37172:p256_snova37172:snova2455:p384_snova2455:snova296
5:p521_snova2965
Supported groups: smaug_t1
Shared groups: smaug_t1
CIPHER is TLS_AES_256_GCM_SHA384
This TLS version forbids renegotiation.
DONE
shutting down SSL
CONNECTION CLOSED
-----BEGIN SSL SESSION PARAMETERS-----
MIGEAgEBAgIDBAQCEwIEILLpg33xfhjD86dpxYduRgOl3Xe9Aamyih0RG/k1ouLw
BDCtA0Bh+cOwIiBEIbyCjI4X0ehEu+cKR0h7XiwH4ujDVUABgf2b3VnHYZ0+t2LL
KnehBgIEaFpbZ6IEAgIcIKQGBAQBAAAArgYCBCOEGBezBQIDAP5F
-----END SSL SESSION PARAMETERS-----
Shared ciphers:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_G
CM_SHA256
Signature Algorithms: id-ml-dsa-65:id-ml-dsa-87:id-ml-dsa-44:ECDSA+SHA256:ECDSA+
SHA384:ECDSA+SHA512:ed25519:ed448:ecdsa_brainpoolP256r1_sha256:ecdsa_brainpoolP3
84r1_sha384:ecdsa_brainpoolP512r1_sha512:rsa_pss_pss_sha256:rsa_pss_pss_sha384:r
sa_pss_pss_sha512:RSA-PSS+SHA256:RSA-PSS+SHA384:RSA-PSS+SHA512:RSA+SHA256:RSA+SH
A384:RSA+SHA512:p256_mldsa44:rsa3072_mldsa44:p384_mldsa65:p521_mldsa87:falcon512
:p256_falcon512:rsa3072_falcon512:falconpadded512:p256_falconpadded512:rsa3072_f
alconpadded512:falcon1024:p521_falcon1024:falconpadded1024:p521_falconpadded1024
:sphincssha2128fsimple:p256_sphincssha2128fsimple:rsa3072_sphincssha2128fsimple:
sphincssha2128ssimple:p256_sphincssha2128ssimple:rsa3072_sphincssha2128ssimple:s
phincssha2192fsimple:p384_sphincssha2192fsimple:sphincsshake128fsimple:p256_sphi
ncsshake128fsimple:rsa3072_sphincsshake128fsimple:mayo1:p256_mayo1:mayo2:p256_ma
yo2:mayo3:p384_mayo3:mayo5:p521_mayo5:CROSSrsdp128balanced:OV_Ip_pkc:p256_OV_Ip_
pkc:OV_Ip_pkc_skc:p256_OV_Ip_pkc_skc:snova2454:p256_snova2454:snova2454esk:p256_
snova2454esk:snova37172:p256_snova37172:snova2455:p384_snova2455:snova2965:p521_
snova2965
Shared Signature Algorithms: id-ml-dsa-65:id-ml-dsa-87:id-ml-dsa-44:ECDSA+SHA256
:ECDSA+SHA384:ECDSA+SHA512:ed25519:ed448:ecdsa_brainpoolP256r1_sha256:ecdsa_brai
npoolP384r1_sha384:ecdsa_brainpoolP512r1_sha512:rsa_pss_pss_sha256:rsa_pss_pss_s
ha384:rsa_pss_pss_sha512:RSA-PSS+SHA256:RSA-PSS+SHA384:RSA-PSS+SHA512:RSA+SHA256
:RSA+SHA384:RSA+SHA512:p256_mldsa44:rsa3072_mldsa44:p384_mldsa65:p521_mldsa87:fa
lcon512:p256_falcon512:rsa3072_falcon512:falconpadded512:p256_falconpadded512:rs
a3072_falconpadded512:falcon1024:p521_falcon1024:falconpadded1024:p521_falconpad
ded1024:sphincssha2128fsimple:p256_sphincssha2128fsimple:rsa3072_sphincssha2128f
simple:sphincssha2128ssimple:p256_sphincssha2128ssimple:rsa3072_sphincssha2128ss
imple:sphincssha2192fsimple:p384_sphincssha2192fsimple:sphincsshake128fsimple:p2
56_sphincsshake128fsimple:rsa3072_sphincsshake128fsimple:mayo1:p256_mayo1:mayo2:
p256_mayo2:mayo3:p384_mayo3:mayo5:p521_mayo5:CROSSrsdp128balanced:OV_Ip_pkc:p256
_OV_Ip_pkc:OV_Ip_pkc_skc:p256_OV_Ip_pkc_skc:snova2454:p256_snova2454:snova2454es
k:p256_snova2454esk:snova37172:p256_snova37172:snova2455:p384_snova2455:snova296
5:p521_snova2965
Supported groups: smaug_t1
Shared groups: smaug_t1
CIPHER is TLS_AES_256_GCM_SHA384
This TLS version forbids renegotiation.
DONE
shutting down SSL
CONNECTION CLOSED
```

```
simminju@simminjuui-MacBookPro tls-test_KpqC % ./bench_loop_100.sh
✔ 성공 연결: 100 / 100
⏱ 총 소요 시간: 3.448748000s
⚡ 처리량: 28.99 connections/sec
⏳ 평균 Latency: 30.00 ms
simminju@simminjuui-MacBookPro tls-test_KpqC %
```

```
cat > bench_loop_100.sh << 'EOF'
#!/usr/bin/env bash

export PATH="$(brew --prefix openssl@3)/bin:$PATH"
export OPENSSL_MODULES=~/oqs-provider_KpqC/build/lib
export PROV_PATH=~/oqs-provider_KpqC/build/lib

N=100
START=$(date +%s.%N)
SUCCESS=0

# POSIX 방식 for 루프
for i in $(seq 1 $N); do
  if openssl s_client \
      -connect localhost:8443 \
      -tls1_3 \
      -groups smuag_t1 \
      -provider default -provider base -provider oqsprovider \
      -provider-path $PROV_PATH \
      < /dev/null \
      > /dev/null 2>&1; then
    SUCCESS=$((SUCCESS+1))
  fi
done

END=$(date +%s.%N)
ELAPSED=$(echo "$END - $START" | bc)
TPS=$(echo "scale=2; $SUCCESS / $ELAPSED" | bc)
AVG_MS=$(echo "scale=2; ($ELAPSED / $SUCCESS) * 1000" | bc)

echo "✔ 성공 연결: $SUCCESS / $N"
echo "⏱ 총 소요 시간: ${ELAPSED}s"
echo "⚡ 처리량: ${TPS} connections/sec"
echo "⏳ 평균 Latency: ${AVG_MS} ms"
EOF

chmod +x bench_loop_100.sh
bash bench_loop_100.sh
```

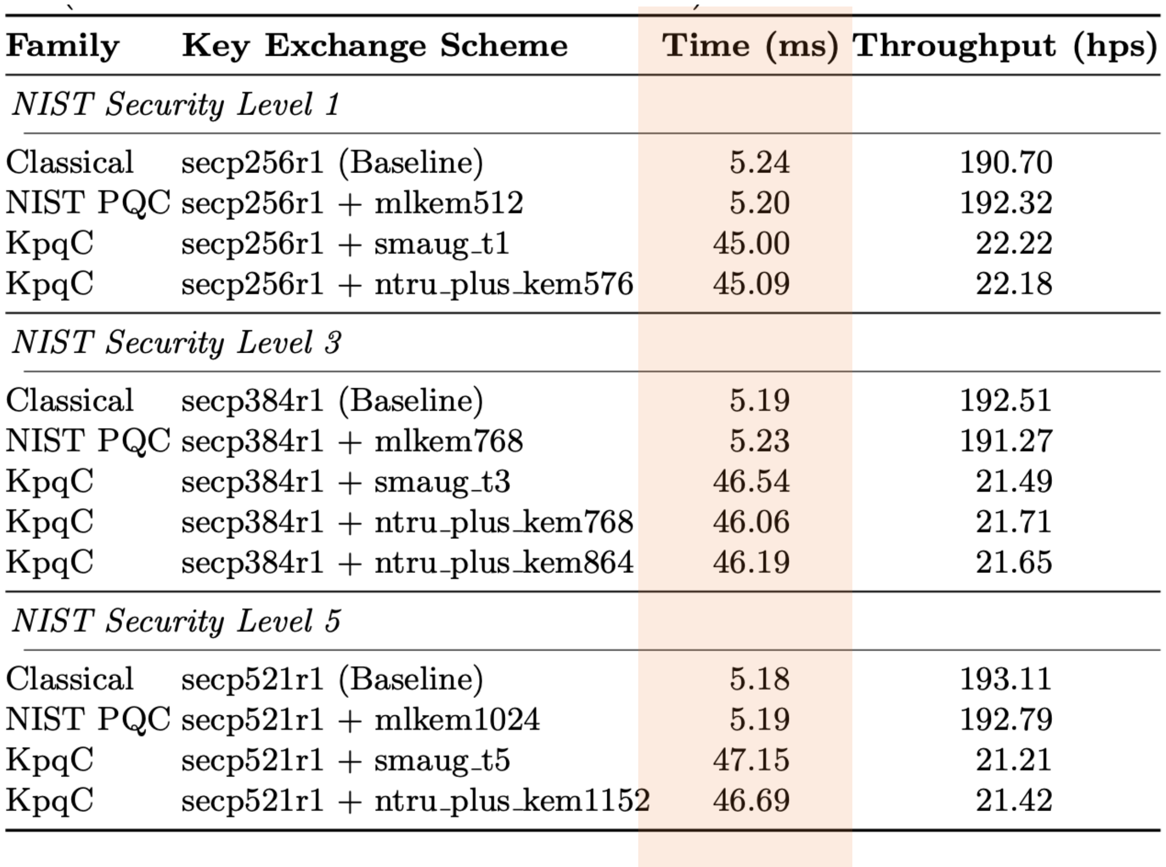
Composite 인증서 측정



Family	Algorithm	Level	PQC-Only (B)	Hybrid (B)
NIST Security Level 1 & 2				
Classical	secp256r1	1	385	-
NIST PQC	falcon512	1	1788	1941
KpqC	aimer128s	1	4427	4582
KpqC	aimer128f	1	6155	6310
NIST PQC	sphincsshake128fsimple	1	17,382	17,536
NIST PQC	mldsa44	2	3977	4120
KpqC	haetae2	2	2702	2857
NIST Security Level 3				
Classical	secp384r1	3	447	-
NIST PQC	mldsa65	3	5506	5713
KpqC	haetae3	3	4057	4276
KpqC	aimer192s	3	9404	9624
KpqC	aimer192f	3	13,340	13,559
NIST Security Level 5				
Classical	secp521r1	5	521	-
NIST PQC	mldsa87	5	7464	7742
NIST PQC	falcon1024	5	3304	3596
KpqC	haetae5	5	5264	5554
KpqC	aimer256s	5	17,356	17,647
KpqC	aimer256f	5	25,420	25,711

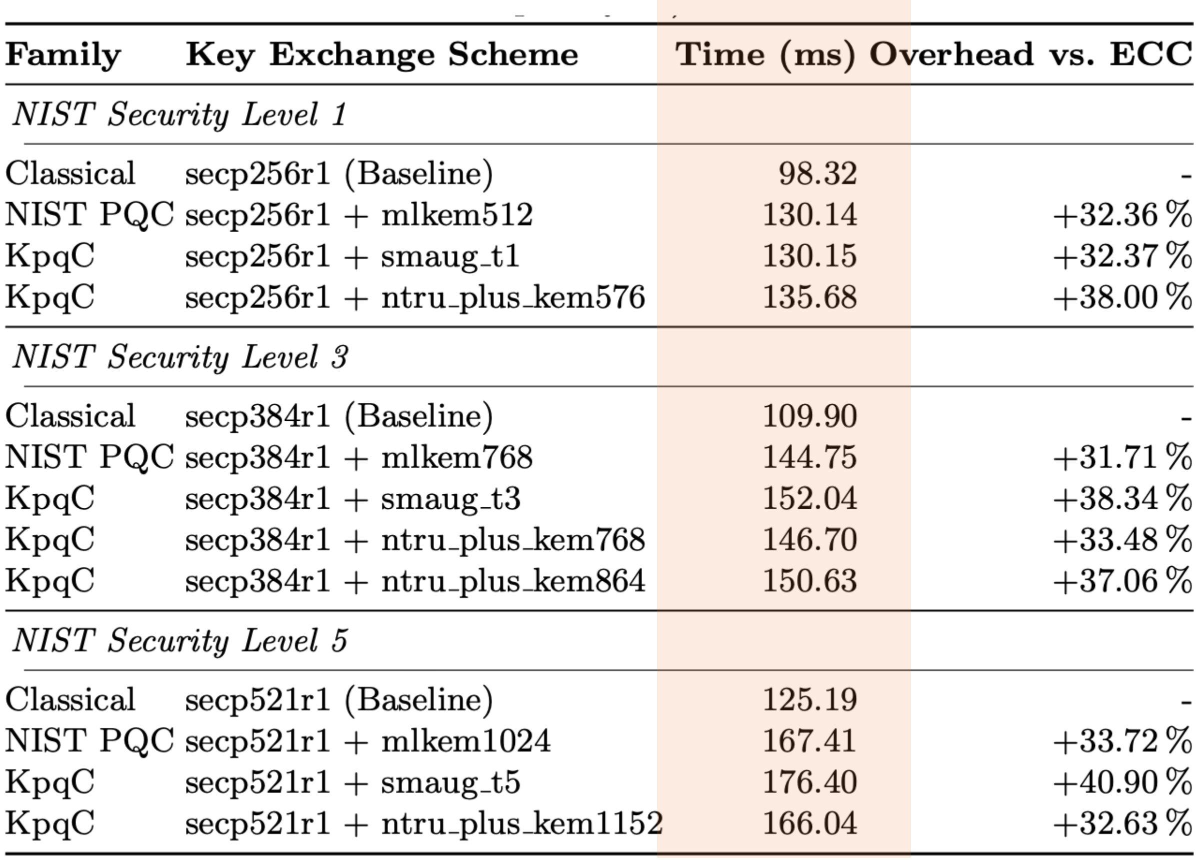
P-256인증서+ 하이브리드 TLS(200회 반복)

- 200회 핸드쉐이크 수행



Family	Key Exchange Scheme	Time (ms)	Throughput (hps)
NIST Security Level 1			
Classical	secp256r1 (Baseline)	5.24	190.70
NIST PQC	secp256r1 + mlkem512	5.20	192.32
KpqC	secp256r1 + smaug_t1	45.00	22.22
KpqC	secp256r1 + ntru_plus_kem576	45.09	22.18
NIST Security Level 3			
Classical	secp384r1 (Baseline)	5.19	192.51
NIST PQC	secp384r1 + mlkem768	5.23	191.27
KpqC	secp384r1 + smaug_t3	46.54	21.49
KpqC	secp384r1 + ntru_plus_kem768	46.06	21.71
KpqC	secp384r1 + ntru_plus_kem864	46.19	21.65
NIST Security Level 5			
Classical	secp521r1 (Baseline)	5.18	193.11
NIST PQC	secp521r1 + mlkem1024	5.19	192.79
KpqC	secp521r1 + smaug_t5	47.15	21.21
KpqC	secp521r1 + ntru_plus_kem1152	46.69	21.42

맥북 로컬테스트



Family	Key Exchange Scheme	Time (ms)	Overhead vs. ECC
NIST Security Level 1			
Classical	secp256r1 (Baseline)	98.32	-
NIST PQC	secp256r1 + mlkem512	130.14	+32.36 %
KpqC	secp256r1 + smaug_t1	130.15	+32.37 %
KpqC	secp256r1 + ntru_plus_kem576	135.68	+38.00 %
NIST Security Level 3			
Classical	secp384r1 (Baseline)	109.90	-
NIST PQC	secp384r1 + mlkem768	144.75	+31.71 %
KpqC	secp384r1 + smaug_t3	152.04	+38.34 %
KpqC	secp384r1 + ntru_plus_kem768	146.70	+33.48 %
KpqC	secp384r1 + ntru_plus_kem864	150.63	+37.06 %
NIST Security Level 5			
Classical	secp521r1 (Baseline)	125.19	-
NIST PQC	secp521r1 + mlkem1024	167.41	+33.72 %
KpqC	secp521r1 + smaug_t5	176.40	+40.90 %
KpqC	secp521r1 + ntru_plus_kem1152	166.04	+32.63 %

맥북(클라이언트)-라즈베리(서버)

Q & A