KpqClean ver2

https://youtu.be/KsQskcm6tx0

KpqClean

- 2021년 말부터 국내 양자내성암호 표준 선정을 위한 KpqC 공모전이 진행 중
- 작년 12월 2라운드 진출 알고리즘 8종 발표
- 24.4~24.11 2라운드가 진행 중

Selected Algorithms from the KpqC Competition Round 1

(December 7, 2023)

After nearly a year of evaluation, the KpqC team is pleased to announce the algorithms that will advance to the KpqC Competition Round 2.
The following are eight Round 2 candidates.

Digital Signature	PKE/KEM
AIMer	NTRU+
HAETAE	PALOMA
MQ-Sign	REDOG
NCC-Sign	SMAUG+TiGER (merged)

KpqC Competition Round 2

(April 2024 ~ November 2024)

- KpqC-bulletin board : The kpqc-bulletin Google group for any official comments on the KpqC algorithms (To send a post, refer to here.)
- Email kpqcrypto@gmail.com for any administrative questions.

Digital Signature Algorithms

* : Principal submitter

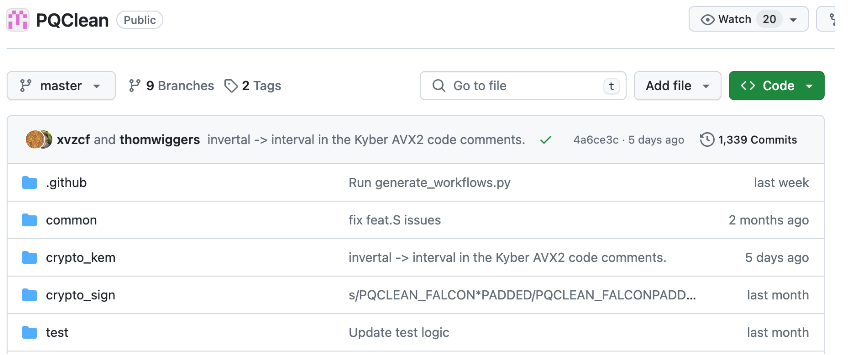
Algorithm	Algorithm Information	Submitters
AIMer	Document Implementation package Website	Seongkwang Kim Jincheol Ha Mincheol Son Byeonghak Lee Dukjae Moon Joohee Lee Sangyub Lee Jihoon Kwon Jihoon Cho Hyojin Yoon Jooyoung Lee*

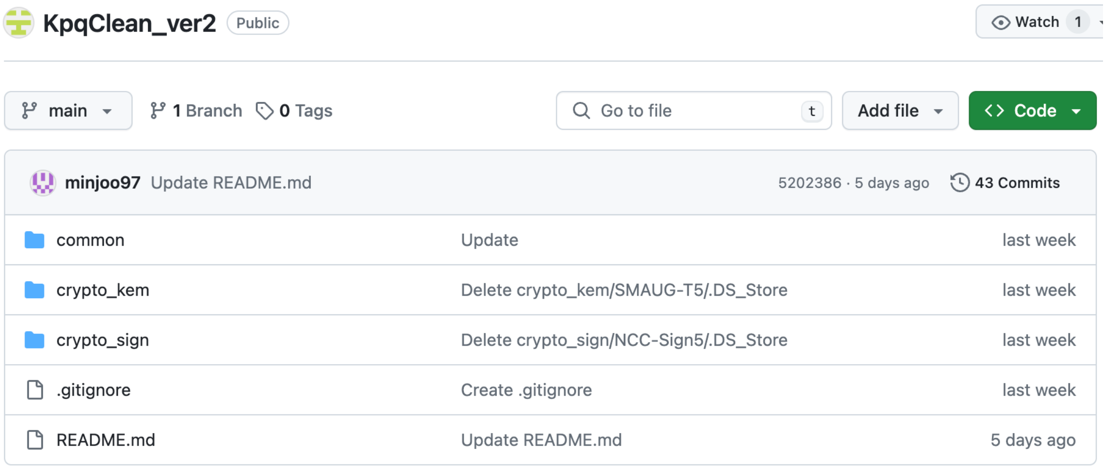
https://www.kpqc.or.kr/competition_02.html

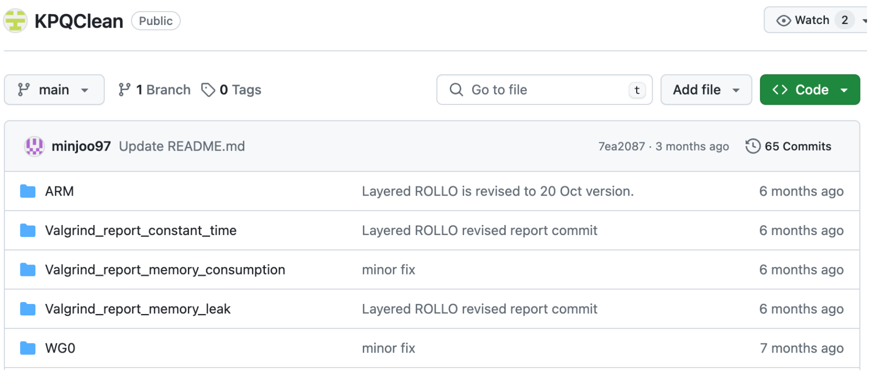
KpqClean ver2

혁동님 세미나
https://www.youtube.com/watch?v=9-AXrGfheCo&list=PLdOq9g7U6Pdt5ZlWeffEU-ViDUS6j-jFS&index=95

- KpqClean 프로젝트
  - 동일한 환경에서 2라운드 알고리즘에 대한 성능 측정을 위해 프로젝트 수행
  - 작년에 시작한 프로젝트를 기반으로 새로운 방향성으로 작업
    - PQClean과 거의 유사한 구조를 띄게 변경
    - PQClean과 차별점...?
      - 하나의 makefile로 x86, aarch64에서 성능 측정 가능
      - PQClean 프로젝트에는 m1cycles.c 가 없기때문에 x86 상에서 벤치마크 되어 있다고 판단됨 (확인 필요)







https://github.com/kpqc-cryptocraft/KpqClean_ver2

KpqClean ver2

- 동일한 환경에서 2라운드 알고리즘에 대한 성능 측정을 위해 PQClean의 코드로 대체 작업 수행
- 이외에 동일하게 사용하는 aes256ctr나 KAT파일 생성을 위한 rng.c 등은 PQClean 이외의 NIST 코드 등을 사용하여 모든 알고리즘들이 동일한 common 파일을 사용하게 수정 작업 진행

PQClean / common /

Add file ⌄ ···

vincentvbh and thomwiggers fix feat.S issues 05df469 · 2 months ago History

Name	Last commit message	Last commit date
..		
keccak2x	fix feat.S issues	2 months ago
keccak4x	chore(formatting): Update Astyle to 3.4	7 months ago
aes.c	chore(formatting): Update Astyle to 3.4	7 months ago
aes.h	chore(formatting): Update Astyle to 3.4	7 months ago
compat.h	chore(formatting): Update Astyle to 3.4	7 months ago
crypto_declassify.h	Add crypto_declassify.h to easier support TIMECOP-supporting schemes	2 years ago
fips202.c	chore(formatting): Update Astyle to 3.4	7 months ago
fips202.h	chore(formatting): Update Astyle to 3.4	7 months ago
nistseedexpander.c	chore(formatting): Update Astyle to 3.4	7 months ago
nistseedexpander.h	HQC submission (#202)	4 years ago
randombytes.c	chore(formatting): Update Astyle to 3.4	7 months ago
randombytes.h	chore(formatting): Update Astyle to 3.4	7 months ago
sha2.c	chore(formatting): Update Astyle to 3.4	7 months ago
sha2.h	chore(formatting): Update Astyle to 3.4	7 months ago
sp800-185.c	chore(formatting): Update Astyle to 3.4	7 months ago
sp800-185.h	chore(formatting): Update Astyle to 3.4	7 months ago

KpqClean_ver2 / common /

minjoo97 Update

Name
..
keccak4x
aes.c
aes.h
aes256ctr.c
aes256ctr.h
compat.h
cpucycles.c
cpucycles.h
fips202.c
fips202.h
randombytes.c
randombytes.h
rng.c
rng.h
sha2.c
sha2.h

<Detail>

PKE/KEM
- AIMer : randombytes, fips202
- HAETAE : randombytes, fips202
- MQSign : fips202, aes
- NCCSign : randombytes, fips202

DSA
- NTRU+ : randombytes, aes, sha2
- PALOMA : none
- SMAUG : randombytes, aes, sha2

KpqClean ver2

```
CC =gcc
CFLAGS += -Wall -Wextra -Wpedantic -Wmissing-prototypes -Wredundant-decls \
  -Wshadow -Wpointer-arith -O3 -fomit-frame-pointer -I../../../common -I./
NISTFLAGS += -Wno-unused-result -O3 -fomit-frame-pointer -I../../../common -I./
RM = /bin/rm

COMMON_DIR=../../../common
COMMON_FILES= $(COMMON_DIR)/randombytes.c  $(COMMON_DIR)/cpucycles.c $(COMMON_DIR)/fips202.c
COMMON_HEADERS= $(COMMON_DIR)/randombytes.h  $(COMMON_DIR)/cpucycles.h $(COMMON_DIR)/fips202.h

BENCH_DIR=../../sign_bench
BENCH_FILES=$(BENCH_DIR)/Sign_KPQC_bench.c
GENKAT_FILES= $(BENCH_DIR)/PQCgenKAT_sign.c

SOURCES= decompose.c encoding.c fft.c fixpoint.c ntt.c packing.c poly.c polyfix.c polymat.c polyvec.c reduce.c sampler.c sign.c symmetric-shake.c
HEADERS=  decompose.h encoding.h fft.h fixpoint.h ntt.h packing.h poly.h polyfix.h polymat.h polyvec.h reduce.h sampler.h sign.h symmetric.h rans_byte.h

GENKAT_SOURCES= $(COMMON_DIR)/aes.c $(COMMON_DIR)/sha2.c $(COMMON_DIR)/rng.c $(COMMON_DIR)/fips202.c
GENKAT_HEADERS= $(COMMON_DIR)/aes.h $(COMMON_DIR)/sha2.h $(COMMON_DIR)/rng.h $(COMMON_DIR)/fips202.h

.PHONY: all KpqC_bench PQCgenKAT clean

all: \
	KpqC_bench \
	PQCgenKAT

KpqC_bench: $(COMMON_HEADERS) $(HEADERS) $(COMMON_FILES) $(SOURCES) $(BENCH_FILES)
	$(CC) $(CFLAGS) -o $@ $(BENCH_FILES) $(SOURCES) $(COMMON_FILES)

PQCgenKAT: $(GENKAT_HEADERS) $(HEADERS) $(GENKAT_SOURCES) $(SOURCES) $(GENKAT_FILES)
	$(CC) $(NISTFLAGS) -o $@ $(GENKAT_FILES) $(SOURCES) $(GENKAT_SOURCES)

clean:
	-$(RM) -f KpqC_bench
	-$(RM) -f PQCgenKAT
	rm -f *.rsp
	rm -f *.req
```

- common
  - keccak4x
  - aes.c
  - aes.h
  - aes256ctr.c
  - aes256ctr.h
  - compat.h
  - cpucycles.c
  - cpucycles.h
  - fips202.c
  - fips202.h
  - randombytes.c
  - randombytes.h
  - rng.c
  - rng.h
  - sha2.c
  - sha2.h

- crypto_kem
  - kem_pke_bench
  - NTRU+KEM576
  - NTRU+KEM768
  - NTRU+KEM864
  - NTRU+KEM1152
  - NTRU+PKE576
  - NTRU+PKE768
  - NTRU+PKE864
  - NTRU+PKE1152
  - PALOMA-128
  - PALOMA-192
  - PALOMA-256
  - SMAUG-T1
  - SMAUG-T3
  - SMAUG-T5
  - SMAUG-Timer

- crypto_sign
  - AIMer128f
  - AIMer128s
  - AIMer192f
  - AIMer192s
  - AIMer256f
  - AIMer256s
  - HAETAE2
  - HAETAE3
  - HAETAE5
  - MQ-Sign_MQLR_256_72_46
  - MQ-Sign_MQLR_256_112_72
  - MQ-Sign_MQLR_256_148_96
  - MQ-Sign_MQRR_256_72_46
  - MQ-Sign_MQRR_256_112_72
  - MQ-Sign_MQRR_256_148_96
  - NCC-Sign1
  - NCC-Sign3
  - NCC-Sign5
  - sign_bench

성능 측정

```c
#include <dlfcn.h>
#include <pthread.h>
#include "cpucycles.h"

#if defined(__x86_64__) || defined(_M_X64)
uint64_t cpucycles(void) {
    unsigned int hi, lo;

    __asm__ __volatile__ ("rdtsc\n\t" : "=a" (lo), "=d"(hi));

    return ((int64_t)lo) | (((int64_t)hi) << 32);
}

#elif defined(__aarch64__)


#define KPERF_LIST                                   \
    /*  ret, name, params */                         \
    F(int, kpc_get_counting, void)                   \
    F(int, kpc_force_all_ctrs_set, int)              \
    F(int, kpc_set_counting, uint32_t)               \
    F(int, kpc_set_thread_counting, uint32_t)        \
    F(int, kpc_set_config, uint32_t, void *)         \
    F(int, kpc_get_config, uint32_t, void *)         \
    F(int, kpc_set_period, uint32_t, void *)         \
    F(int, kpc_get_period, uint32_t, void *)         \
    F(uint32_t, kpc_get_counter_count, uint32_t)     \
    F(uint32_t, kpc_get_config_count, uint32_t)      \
    F(int, kperf_sample_get, int *)                  \
    F(int, kpc_get_thread_counters, int, unsigned int, void *)

#define F(ret, name, ...)                            \
    typedef ret name##proc(__VA_ARGS__);             \
    static name##proc *name;
KPERF_LIST
#undef F
```

```c
#ifndef m1cycles_h
#define m1cycles_h

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

#if defined(__x86_64__) || defined(_M_X64)
    uint64_t cpucycles(void);
#elif defined(__aarch64__)
    void setup_rdtsc(void);
    extern unsigned long long int cpucycles(void);
#endif

#endif /* m1cycles_h */
```

Home / Archives / Volume 2022, Issue 1 / Articles

Neon NTT: Faster Dilithium, Kyber, and Saber on Cortex-A72 and Apple M1

Hanno Becker
Arm Research, Cambridge, UK

Vincent Hwang
National Taiwan University, Taipei, Taiwan; Academia Sinica, Taipei, Taiwan

Matthias J. Kannwischer
Academia Sinica, Taipei, Taiwan

Bo-Yin Yang
Academia Sinica, Taipei, Taiwan

Shang-Yi Yang
Chelpis Co. Ltd., Taipei, Taiwan

PDF

Published
2021-11-19

How to Cite

Becker, H., Hwang, V., Kannwischer, M. J., Yang, B.-Y., & Yang, S.-Y. (2021). Neon NTT: Faster Dilithium, Kyber, and Saber on Cortex-A72 and Apple M1. *IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022*(1), 221–244.

DOI: https://doi.org/10.46586/tches.

Keywords: NIST PQC, Armv8-A, Neon

neon-ntt / common / **m1cycles.c**

vincentvbh more namespacing

Code | Blame | 187 lines (156 loc) · 4.91 KB

```c
/*
 * Duc Tri Nguyen (CERG GMU)
 * Modified from M1:
 * https://gist.github.com/dougallj/5bafb113492047c865c0c8cfbc930155#file-m1_robsize-c-L390
 */

#ifdef __APPLE__

#include <dlfcn.h>
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include "m1cycles.h"

#define KPERF_LIST                                   \
    /*  ret, name, params */                         \
    F(int, kpc_get_counting, void)                   \
    F(int, kpc_force_all_ctrs_set, int)              \
    F(int, kpc_set_counting, uint32_t)               \
    F(int, kpc_set_thread_counting, uint32_t)        \
    F(int, kpc_set_config, uint32_t, void *)         \
    F(int, kpc_get_config, uint32_t, void *)         \
    F(int, kpc_set_period, uint32_t, void *)         \
    F(int, kpc_get_period, uint32_t, void *)         \
    F(uint32_t, kpc_get_counter_count, uint32_t)     \
    F(uint32_t, kpc_get_config_count, uint32_t)      \
    F(int, kperf_sample_get, int *)                  \
    F(int, kpc_get_thread_counters, int, unsigned int, void *)

```

https://gist.github.com/dougallj/5bafb113492047c865c0c8cfbc930155#file-m1_robsize-c-L390

성능 측정 시 주의 사항

- aarch64 환경에서 sudo ./KpqC_bench 를 입력 해야함

```
(base) minjoo@simminjuui-iMac KpqClean ver2 % cd crypto_sign
(base) minjoo@simminjuui-iMac crypto_sign % cd HAETAE2
(base) minjoo@simminjuui-iMac HAETAE2 % ls
AVX2    clean
(base) minjoo@simminjuui-iMac HAETAE2 % cd clean
```

```
(base) minjoo@simminjuui-iMac clean % make clean
/bin/rm test
(base) minjoo@simminjuui-iMac clean % make
gcc -Wall -Wextra -Wpedantic -Wmissing-prototypes -Wredundant-decls -Wshadow -Wpointer-arith -
yvec.c reduce.c sampler.c sign.c symmetric-shake.c ../../../common/randombytes.c  ../../../co
../../sign_bench/Sign_KPQC_bench.c:59:35: warning: incompatible pointer types passing 'unsigne
        crypto_sign_signature(sm, &smlen, m, mlen, sk);
                                  ^~~~~~
./api.h:15:49: note: passing argument to parameter 'siglen' here
int crypto_sign_signature(uint8_t *sig, size_t *siglen, const uint8_t *m,
                                                ^
../../sign_bench/Sign_KPQC_bench.c:71:39: warning: incompatible pointer types passing 'unsigne
        result = crypto_sign_open(m2, &mlen, sm, smlen, pk);
                                      ^~~~~
```

```
kpc_get_thread_counters failed
kpc_get_thread_counters failed
kpc_get_thread_counters failed
kpc_get_thread_counters failed
kpc_get_thread_counters failed
kpc_get_thread_counters failed
kpc_get_thread_counters failed
kpc_get_thread_counters failed
kpc_get_thread_counters failed
kpc_get_thread_counters failed
kpc_get_thread_counters failed
  Verify runs in .................        0 cycles
============================================================
(base) minjoo@simminjuui-iMac clean % ./test
```

```
.PHONY: all KpqC_bench PQCgenKAT clean

all: \
        KpqC_bench \
        PQCgenKAT

KpqC_bench: $(COMMON_HEADERS) $(HEADERS) $(COMMON_FILES) $(SOURCES) $(BENCH_FILES)
        $(CC) $(CFLAGS) -o $@ $(BENCH_FILES) $(SOURCES) $(COMMON_FILES)

PQCgenKAT: $(GENKAT_HEADERS) $(HEADERS) $(GENKAT_SOURCES) $(SOURCES) $(GENKAT_FILES)
        $(CC) $(NISTFLAGS) -o $@ $(GENKAT_FILES) $(SOURCES) $(GENKAT_SOURCES)

clean:
        -$(RM) -f KpqC_bench
        -$(RM) -f PQCgenKAT
        rm -f *.rsp
        rm -f *.req
```

```
(base) minjoo@simminjuui-iMac clean % sudo ./test
Password:
BENCHMARK ENVIRONMENTS  ==============================
CRYPTO_ALGNAME: HAETAE2
CRYPTO_PUBLICKEYBYTES: 992
CRYPTO_SECRETKEYBYTES: 1408
CRYPTO_BYTES: 1474
Number of loop: 1000
KeyGen ///////////////////////////////////////////////
  KeyGen runs in .................   927277 cycles
Sign /////////////////////////////////////////////////
  Sign runs in .................   533404 cycles
Verify ///////////////////////////////////////////////
  Verify runs in .................   136568 cycles
======================================================
(base) minjoo@simminjuui-iMac clean %
```

성능 측정 방향성

PKE/KEM (Environment1, -O3)

▼ PKE/KEM-Env1-O3 Table (Unit: clock cycles)

Algorithm	Keygen(Avr.)	Encapsulation(Avr.)	Decapsulation(Avr.)
NTRUplus-KEM576	332,182	81,645	104,401
NTRUplus-KEM768	340,466	106,755	135,825
NTRUplus-KEM864	348,919	111,143	149,990
NTRUplus-KEM1152	712,237	162,622	222,245
NTRUplus-PKE576	318,433	83,165	106,562
NTRUplus-PKE768	353,751	111,652	143,420
NTRUplus-PKE864	349,104	110,053	152,146
NTRUplus-PKE1152	731,388	162,449	226,607
PALOMA-128	128,929,992	131,703	8,337,680
PALOMA-192	615,092,474	177,833	43,985,795
PALOMA-256	725,282,687	206,778	45,836,617
SMAUG-T1	145,860	47,136	62,174
SMAUG-T3	161,114	85,677	117,123
SMAUG-T5	234,227	136,419	165,587
SMAUG-Timer	144,698	46,523	61,796

Table 2: Average execution speed for key generation, signature generation, and signature verification for each scheme implementation, as measured on the Nucleo-L4R5ZI evaluation board. Execution speed is shown in thousands of cycles, with the difference to the minimum and maximum shown in the super- and subscript respectively. Cycles spent on symmetric cryptography shown in parentheses.

Scheme	impl.	keygen		sign		verify	
`dilithium2`	`clean`	1874^{+41}_{-35}	(62%)	7283^{+13672}_{-3962}	(37%)	2062^{+0}_{-0}	(53%)
	`m4f`	1426^{+40}_{-47}	(80%)	3815^{+7908}_{-2001}	(67%)	1417^{+0}_{-0}	(77%)
`dilithium3`	`clean`	3205^{+2}_{-1}	(65%)	12893^{+52247}_{-7796}	(40%)	3376^{+0}_{-0}	(57%)
	`m4f`	2516^{+1}_{-1}	(82%)	6374^{+11353}_{-3439}	(69%)	2411^{+0}_{-0}	(79%)
`dilithium5`	`clean`	5340^{+66}_{-53}	(67%)	15533^{+35954}_{-7581}	(45%)	5610^{+0}_{-0}	(61%)
	`m4f`	4277^{+41}_{-46}	(84%)	8473^{+16493}_{-3591}	(74%)	4185^{+0}_{-0}	(82%)
`haetae2`	`ref`	9265^{+49825}_{-7549}	(25%)	$32068^{+153018}_{-25792}$	(43%)	1154^{+450}_{-50}	(45%)
	`m4f`	9184^{+34372}_{-7629}	(27%)	26104^{+95950}_{-21385}	(57%)	918^{+0}_{-0}	(54%)
`haetae3`	`ref`	17553^{+59078}_{-14530}	(30%)	$44320^{+116183}_{-34537}$	(43%)	2097^{+890}_{-99}	(50%)
	`m4f`	14630^{+63266}_{-11877}	(33%)	$30588^{+159334}_{-23135}$	(57%)	1761^{+0}_{-0}	(57%)

향후 계획

- 현재 AVX2 작업 중
  - 일부 알고리즘의 성능 측정 결과 KeyGen 값이 튀는 현상도 추가로 확인되어 해당 부분도 다시 검증 예정
  - 5월 초에 AVX2 작업을 마무리하고 깃헙 업로드 예정
- 작년과 동일하게 Valgrind를 활용하여 메모리 사용량과 constant timing 검증 등을 수행할 계획
  - 이 과정에서 SUPERCOP도 같이 활용할 예정
  - 하지만 SUPERCOP은 아직 정확한 사용 방법을 습득하지 못하여 익숙한 Valgrind로 먼저 수행하려고 합니다...

Q & A