KpqC 공모전 알고리즘 기본 구현에 대한 성능 비교 분석

(KpqC 연구단 6차 워크숍@아일랜드 리솜리조트)
2023. 7. 13.

HANSUNG UNIVERSITY

권혁동, 송경주, 심민주, 이민우, 김상원, **서화정**

CryptoCraft LAB

벤치마크 관련 진행 사항

1차 성능 평가 결과

성능 평가 시 특이사항

향후 고도화 계획

본 발표에서 다루는 내용

- **암호화 알고리즘의 효율성 비교 분석**
- **효율성이란?**
  - 동일한 일을 좋은 방법으로 빠르게 수행하는 것
  - 예시) 안면도를 오는 다양한 방법론 (루트)

1 실시간 추천
2시간 **32**분 | 196km
택시비 209,640원 | 통행료 10,300원 | 연료비 27,016원

서행 내부순환로 12km → 원활 서해안고속도로 148km →
미확인 원산대로 13km

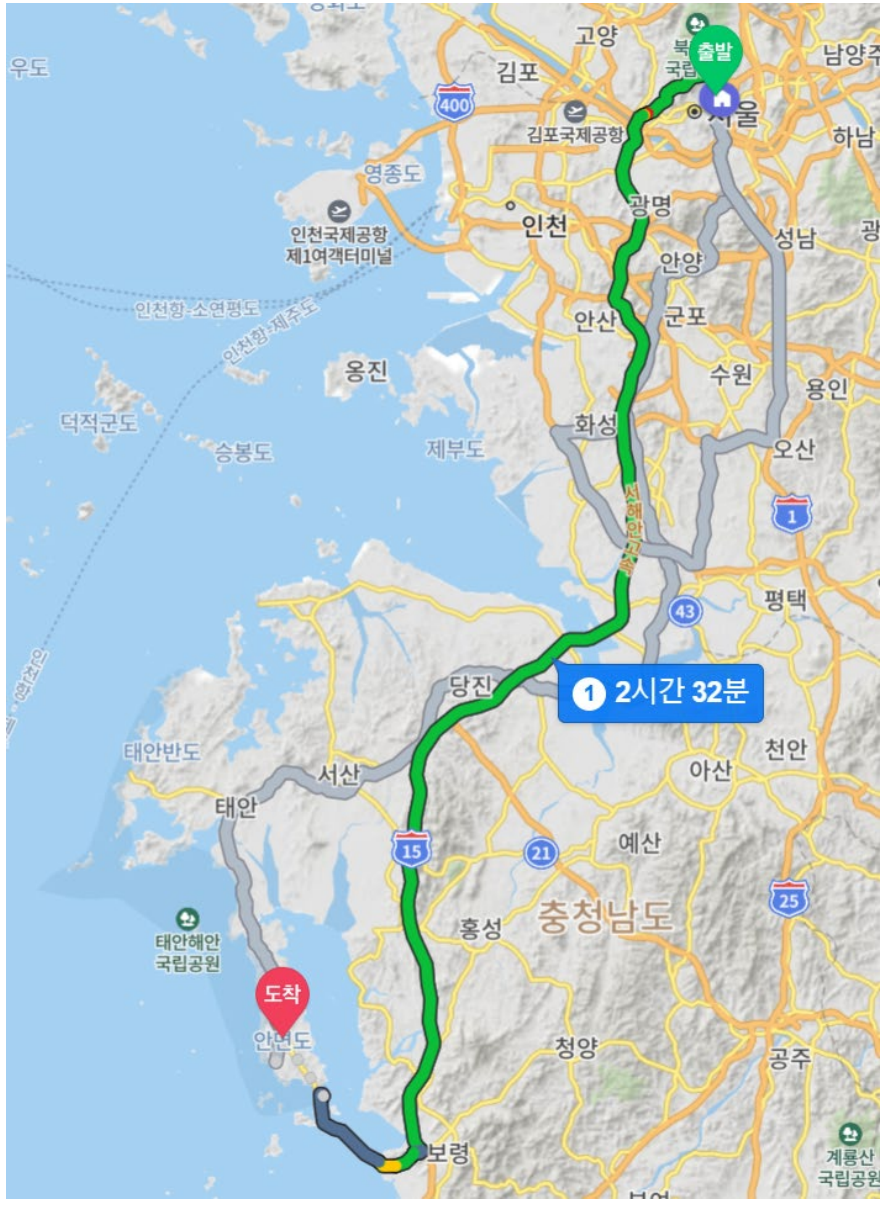
상세보기 >

2 서해안고속도로 이용
2시간 **43**분 | 204km
택시비 221,620원 | 통행료 11,100원 | 연료비 28,078원

원활 경부고속도로 40km →
원활 평택파주고속도로(평택-화성) 18km →
원활 서해안고속도로 100km

상세보기 >

3 서해안고속도로 이용
2시간 **41**분 | 208km
택시비 224,040원 | 통행료 12,300원 | 연료비 28,682원



본 발표에서 다루는 내용

- **암호화 연산이 효율적이면 좋은 점?**
  - 서비스 가용성 향상
  - 전력 소모 감소

- **NIST에서는 2라운드부터 효율성을 강조**
  - 최종 표준안인
    Dilithium의 경우 매우 높은 효율성을 가짐

Signature Algorithm	Local Machine (ms)				Cloud Instance (ms)			
	Sign		Verify		Sign		Verify	
	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.
RSA 3072	3.19	0.023	0.06	0.001	2.39	0.010	0.04	0.002
ECDSA 384	1.32	0.012	1.05	0.020	1.28	0.015	0.93	0.004
Dilithium II	0.82	0.021	0.16	0.005	0.41	0.018	0.12	0.008
Falcon 512	5.22	0.054	0.05	0.004	6.50	0.091	0.07	0.003
MQDSS 48	10.30	0.147	7.25	0.100	10.25	0.181	7.40	0.110
Picnic $L1FS$	4.09	0.050	3.25	0.049	3.17	0.051	2.39	0.044
SPHINCS$^+$ SHA256-128f-simple	93.37	0.654	3.92	0.043	62.7	0.548	2.50	0.037
Rainbow Ia	0.34	0.015	0.83	0.036	0.25	0.020	0.48	0.044
Dilithium IV	1.25	0.021	0.30	0.012	0.46	0.019	0.23	0.010
Falcon 1024	11.37	0.102	0.11	0.005	14.20	0.156	0.14	0.005

The 2nd Round
of the NIST PQC Standardization Process

Dustin Moody

NIST National Institute of Standards and Technology U.S. Department of Commerce

Performance

- We have internal numbers, based on implementations sent to us
  - We strongly prefer code that is constant time
- Performance will play a larger role in the 2nd Round
  - We encourage benchmarking on a variety of platforms
  - We are looking for mature schemes – beyond just proof of concept
- Implementations can always be updated
  - We won't change the implementations on our Round 2 webpage
  - Teams should feel free to advertise results on the pqc-forum, and on their own websites

본 발표에서 다루는 내용

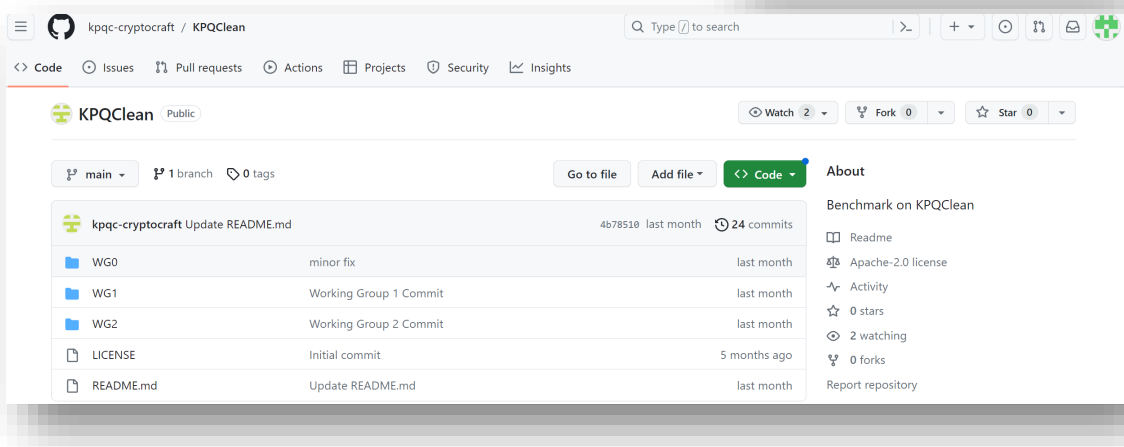
- **KpqC Round 1 후보 알고리즘들에 대한 벤치마킹**
  - 현재 각 알고리즘이 제시하는 벤치마크 결과는 **각기 다른 컴퓨터 환경에서 측정**
  - **동일한 환경에서 알고리즘들을 벤치마크**하여 객관성을 확보하는 것이 목표
  - 내부 모듈 (AES, SHA, AVX2)을 통일하여 모듈에 따른 성능 차이를 최소화
- **공정한 벤치마크 결과 제공**
  - 개발 및 운영 환경 차이, 모듈 차이로 인한 성능 격차 최소화
  - 동일한 환경에서 측정한 성능으로 공정한 성능 비교가 진행되도록 함
- **KpqC 후보 알고리즘의 의존성 제거**
  - NIST 양자내성암호와 관련된 PQClean 프로젝트에서 추구한 방향성
  - 의존성 제거 시 **플랫폼에 구애받지 않고 알고리즘 가동 용이**
  - 개발 환경 설정의 난이도를 낮춰서 양자내성 알고리즘 벤치마크 진입장벽 완화

현재 벤치마크 관련 진행된 사항

https://github.com/kpqc-cryptocraft/KPQClean/tree/main

프로젝트 코드 주소

- **OpenSSL 의존성 제거**
  - 모든 알고리즘에서 **OpenSSL 의존성 제거**
  - gcc 컴파일러를 사용가능한 환경이면 소스코드 실행이 설정없이 가능
  - **추가적인 외부 라이브러리 의존성 제거 진행 중**



- **1차 벤치마크 수행 사항 및 추후 계획**
  - **Ryzen CPU (Intel과 유사) 상에서 벤치마크 진행 완료**
  - **Ubuntu OS를 사용한 벤치마크**
  - 다수의 디바이스에서 추가적인 벤치마크 수행 예정

KpqC 1라운드 선정 알고리즘

양자 내성 암호 연구단은 양자컴퓨터에 의한 보안 위협에 대비하기 위해 PQC 국내 표준을 선정하고자 양자내성암호 (KpqC) 국가공모전을 진행 중

KpqC	
2021.11.25	공모전 공지
2022.02.18	제출마감
2022.03.15	1차 평가
2022.12	**1라운드 결과 발표 2라운드 후보 목록 공개**
2023.09	2라운드 알고리즘 발표
2024.09	KpqC 공모전 최종결과 발표 예정

- KEM: 7종류
- Digital Signature: 9종류

Type	Code-based	Lattice-based	Multivariate Quadratic-based	Isogeny-based	Zero-knowledge
Encryption/Key-establishment	IPCC	NTRU+	-	-	-
	Layered ROLLO	SMAUG			
	PALOMA	TiGER			
	REDOG				
Digital Signature	Enhanced pqsigRM	GCKSign	MQ-Sign	FIBS	AIMer
		HAETAE			
		NCC-Sign			
		Peregrine			
		SOLMAE			

공개키 암호화 암호문 크기

암호문과 공개키의 크기는 네트워크 트래픽에 영향을 미침

- **Classic McEliece (NIST Round 4 후보)가 가장 작은 암호문 크기를 지님**
- **KpqC 알고리즘으로 한정지으면, PALOMA 알고리즘이 가장 작은 암호문 크기를 가짐**
  - IPCC는 보안 강도에 따른 암호문 크기 차이가 존재하지 않음
  - 국제 표준인 Kyber를 기준으로 한다면, 2,048 바이트 정도 까지가 안정적인 암호문 크기라고 할 수 있음

KpqC standards/candidates

NIST standards/candidates

Rank	PKE Scheme	Security level	Cipher size(Bytes)	Rank	PKE Scheme	Security level	Cipher size(Bytes)
1	mceliece348864	1	96	21	KYBER-768	3	1,088
2	PALOMA-128	1	136	22	NTRU+-768	1	1,152
3	mceliece460896	3	156	23	Layered ROLLO-I-256	5	1,286
4	mceliece6960119	5	194	24	NTRU+-864	3	1,296
5	mceliece6688128	5	208	25	REDOG-II	3	1,440
	mceliece8192128	5	208	26	SMAUG-256 (revised)	5	1,472
7	PALOMA-192	2	240	27	KYBER-1024	5	1,568
	PALOMA-256	3	240	28	NTRU+-1152	5	1,728
9	~~SIKE434~~	1	374	29	TIGER-192	3	2,048
10	~~SIKE503~~	2	434		TIGER-256	5	2,048
11	~~SIKE610~~	3	524	31	REDOG-III	5	2,230
11	Layered ROLLO-I-128	1	620	32	HQC-128	1	4,497
13	~~SIKE751~~	5	644	33	HQC-192	3	9,042
14	SMAUG-128 (revised)	1	672	34	BIKE-I	1	12,579
15	KYBER-512	1	768	35	HQC-256	5	14,485
16	REDOG-1	1	830	36	BIKE-III	3	24,915
17	Layered ROLLO-I-192	3	850	37	BIKE-V	5	41,229
18	NTRU+-576	1	864	38	IPCC-f1	1	322,000
19	TIGER-128	1	1,024		IPCC-f3	3	322,000
	SMAUG-192 (revised)	3	1,024		IPCC-f4	4	322,000

공개키 암호화 공개키 크기

암호문과 공개키의 크기는 네트워크 트래픽에 영향을 미침

- ~~SIKE (NIST Round 4 후보)~~가 가장 작은 공개키 크기를 지님
- KpqC 알고리즘으로 한정지으면, TiGER 알고리즘이 가장 작은 공개키 크기를 가짐
  - IPCC는 보안 강도에 따른 암호문 공개키 크기 차이가 존재하지 않음
  - PALOMA는 KpqC 후보군 중에서 가장 큰 공개키 크기를 가지고 있으며, 이는 4라운드 후보인 McEliece와 비슷함

KpqC standards/candidates

NIST standards/candidates

Rank	PKE Scheme	Security level	Public key(byte)	Rank	PKE Scheme	Security level	Public key(byte)
1	~~SIKE434~~	1	330	21	Layered-ROLLO-I-256	5	2,571
2	~~SIKE503~~	2	378	22	IPCC-f1	1	3,600
3	~~SIKE610~~	3	462		IPCC-f3	3	3,600
4	TiGER-128	1	544		IPCC-f4	4	3,600
5	~~SIKE751~~	5	564	25	HQC-192	3	4,522
6	SMAUG-128 (revised)	1	672	26	HQC-256	5	7,245
7	KYBER-512	1	800	27	BIKE-I	1	12,323
8	NTRU+-576	1	864	28	REDOG-I	1	14,250
9	TiGER-192	3	1,056	29	BIKE-III	3	24,659
	TiGER-256	5	1,056	30	REDOG-II	3	32,840
11	SMAUG-192 (revised)	3	1,088	31	BIKE-V	5	40,973
12	NTRU+-768	1	1,152	32	REDOG-III	5	62,980
13	KYBER-768	3	1,184	33	mceliece348864	1	261,120
14	Layered-ROLLO-I-128	1	1,240	34	PALOMA-128	1	319,488
15	NTRU+-864	3	1,296	35	mceliece460896	3	524,160
16	KYBER-1024	5	1,568	36	PALOMA-192	2	812,032
17	Layered-ROLLO-I-192	3	1,699	37	PALOMA-256	3	1,025,024
18	NTRU+-1152	5	1,728	38	mceliece6688128	5	1,044,992
19	SMAUG-256 (revised)	5	1,792	39	mceliece6960119	5	1,047,319
20	HQC-128	1	2,249	40	mceliece8192128	5	1,357,824

공개키 암호화 비밀키 크기

- **Layered ROLLO가 가장 작은 비밀키 크기**를 지님
  - ROLLO의 암호문과 공개키 크기는 약간 작거나 중간 정도
- NIST 표준인 Kyber를 기준으로 한다면, **대부분의 알고리즘이 이상적인 비밀키** 크기를 지닌다고 할 수 있음
  - 예외로 PALOMA는 매우 큰 비밀키를 가지며, 특히 보안강도 1에 비해 2, 3의 비밀키가 매우 큼
  - IPCC는 모든 스킴에서 비밀키 크기가 동일함

KpqC standards/candidates

NIST standards/candidates

Rank	PKE Scheme	Security level	Private key(byte)	Rank	PKE Scheme	Security level	Private key(byte)
1	Layered-ROLLO-I-128	1	120	21	HQC-128	1	2,289
	Layered-ROLLO-I-192	3	120	22	NTRU+-768	1	2,304
	Layered-ROLLO-I-256	5	120	**23**	**KYBER-768**	**3**	**2,400**
4	SMAUG-128 (revised)	1	174	24	REDOG-II	3	2,520
5	SMAUG-256 (revised)	5	208	25	NTRU+-864	3	2,592
6	SMAUG-192 (revised)	3	230	**26**	**KYBER-1024**	**5**	**3,168**
7	~~SIKE434~~	1	374	27	BIKE-III	3	3,346
8	IPCC-f1	1	400	28	NTRU+-1152	5	3,456
	IPCC-f3	3	400	29	REDOG-III	5	3,890
	IPCC-f4	4	400	30	HQC-192	3	4,562
11	~~SIKE503~~	2	434	31	BIKE-V	5	4,640
12	~~SIKE610~~	3	524	32	mceliece348864	1	6,492
13	TiGER-128	1	528	33	HQC-256	5	7,285
14	~~SIKE751~~	5	644	34	mceliece460896	3	13,608
15	TiGER-192	3	1,056	35	mceliece6688128	5	13,932
	TiGER-256	5	1,056	36	mceliece6960119	5	13,948
17	REDOG-I	1	1,450	37	mceliece8192128	5	14,120
18	**KYBER-512**	**1**	**1,632**	38	PALOMA-128	1	93,008
19	NTRU+-576	1	1,728	39	PALOMA-192	2	355,400
20	BIKE-I	1	2,244	40	PALOMA-256	3	357,064

전자서명 서명 크기

서명과 공개키의 크기는 네트워크 트래픽에 영향을 미침

- **MQSign이 가장 작은 서명 크기**를 지님
  - 모든 서명 스킴이 다른 알고리즘에 비해서 가장 작은 크기를 가짐
- **NIST Standard 중에서 Dilithium이 서명 크기가 큰 편에 속함** (연산효율성 관점에서 네트워크 전송속도 지연보다 연산 효율성이 전체 성능을 좌우)
  - **KpqC 알고리즘은 대체로 Dilithium보다 작은 서명 크기를 가짐**
  - 따라서 KpqC 서명 후보군의 대부분은 안정적인 서명 크기를 지닌다 할 수 있음

KpqC standards/candidates

NIST standards/candidates

Rank	Digital Signature Scheme	Security level	Sig size(Bytes)	Rank	Digital Signature Scheme	Security level	Sig size(Bytes)
1	MQSign-72-46	1	134	19	GCKSign-V	5	3,104
2	MQSign-112-72	3	200	20	NCCSign-I (conserparam)	1	3,186
3	MQSign-148-96	5	260	**21**	**Dilithium-III**	**3**	**3,293**
4	pqsigRM-613	5	528	22	NCCSign-III (original)	3	3,605
5	Falcon-512	1	666	23	NCCSign-III (conserparam)	3	4,251
	SOLMAE-512	1	666	**24**	**Dilithium-V**	**5**	**4,595**
	Peregrine-512	1	666	25	NCCSign-V (original)	5	5,055
8	pqsigRM-612	1	1,040	26	NCCSign-V (conserparam)	5	5,385
9	Falcon-1024	5	1,280	27	AlMer-I (revised, PARAM1)	1	5,904
	Peregrine-1024	5	1,280	28	SPHINCS+ 128s	1	7,856
11	SOLMAE-1024	5	1,375	29	AlMer-III (revised, PARAM1)	3	13,080
12	HAETAE-II (revised)	2	1,463	30	SPHINCS+ 192s	3	16,224
13	GCKSign-II	2	1,952	31	SPHINCS+ 128f	1	17,088
14	GCKSign-III	3	2,080		FIPS	1	17,088
15	HAETAE-III (revised)	3	2,337	32	AlMer-V (revised, PARAM1)	5	25,152
16	**Dilithium-II**	**2**	**2,420**	33	SPHINCS+ 256s	5	29,792
17	NCCSign-I (original)	1	2,458	34	SPHINCS+ 192f	3	35,664
18	HAETAE-V (revised)	5	2,908	35	SPHINCS+ 256f	5	49,856

전자서명 공개키 크기

서명과 공개키의 크기는 네트워크 트래픽에 영향을 미침

- SPHINCS(NIST Round 4 후보)와 **FIPS가 가장 작은 공개키 크기를 지님**
- **AIMer 알고리즘도 거의 비슷한 크기를 가짐**
  - AIMer는 KpqC 후보군 중에서 **가장 큰 서명 크기를 가지지만, 공개키 크기는 가장 작음**
  - **FIPS도 매우 큰 서명 크기를 가짐**
  - 파라미터가 크다고 평가되는 Dilithium을 고려한다면, 2,048~4,096 바이트의 공개키가 적절한 크기로 생각할 수 있음
  - **MQSign은 서명 크기는 가장 작았으나 공개키 크기는 가장 큰 편**에 속함

KpqC standards/candidates

NIST standards/candidates

Rank	Digital Signature Scheme	Security level	Public key(byte)	Rank	Digital Signature Scheme	Security level	Public key(byte)
1	SPHINCS+128	1	32	18	FALCON-1024	5	1,793
	FIPS	1	32	**19**	**Dilithium-III**	**3**	**1,952**
	AIMer-I (revised)	1	32	20	GCKSign-III	3	1,952
4	SPHINCS+192	3	48	21	NCCSign-I (conserparam)	1	1,984
	AIMer-III (revised)	3	48	22	NCCSign-III (original)	3	1,997
6	SPHINCS+256	5	64	23	HAETAE-V (revised)	5	2,080
	AIMer-V (revised)	5	64	24	NCCSign-III (conserparam)	3	2,443
8	SOLMAE-512	1	896	**25**	**Dilithium-V**	**5**	**2,592**
9	Peregrine-512	1	897	26	NCCSign-V (original)	5	2,663
	FALCON-512	1	897	27	GCKSign-V	5	3,040
11	HAETAE-II (revised)	2	992	28	NCCSign-V (conserparam)	5	3,091
12	**Dilithium-II**	**2**	**1,312**	29	MQSign-72-46	1	328,411
13	HAETAE-III (revised)	3	1,472	30	MQSign-112-72	3	1,238,761
14	NCCSign-I (original)	1	1,564	31	pqsigRM-613	5	1,285,120
15	GCKSign-II	2	1,760	32	MQSign-148-96	5	2,892,961
16	SOLMAE-1024	5	1,792	33	pqsigRM-612	1	4,194,304
17	Peregrine-1024	1	1,793				

전자서명 비밀키 크기

- **AIMer가 가장 작은 비밀키 크기**를 지님
  - AIMer는 모든 KpqC 후보군 중에서도 가장 작은 공개키 크기를 가짐
  - **MQSign은 서명 크기가 가장 작으나, 비밀키 크기는 가장 큰 편**에 속함

KpqC standards/candidates

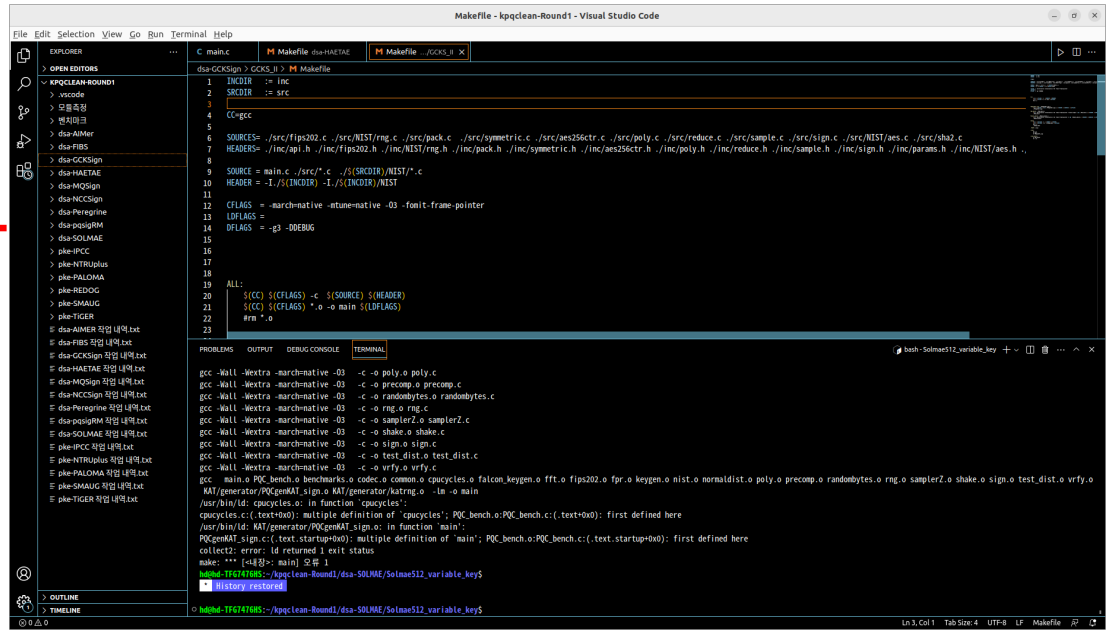
- **파라미터가 크다고 알려진 Dilithium을 기준으로 보면 대부분의 KpqC 알고리즘은 비밀키 크기가 이상적임**
  - 두 알고리즘 (MQSign, pqsigRM)만 비밀키 크기가 큼

NIST standards/candidates

Rank	Digital Signature Scheme	Security level	Private key(byte)	Rank	Digital Signature Scheme	Security level	Private key(byte)
1	AIMer-I (revised)	1	16	18	Peregrine-1024	5	2,305
2	AIMer-III (revised)	3	24		FALCON-1024	5	2,305
3	AIMer-V (revised)	5	32	20	Dilithium-II	2	2,528
4	SPHINCS+128	1	64	21	HAETAE-V (revised)	5	2,720
	FIPS	1	64	22	NCCSign-I (conserparam)	1	2,800
6	SPHINCS+192	3	96	23	NCCSign-III (original)	3	3,312
7	SPHINCS+256	5	128	24	NCCSign-III (conserparam)	3	3,914
8	GCKSign-II	2	288	25	Dilithium-III	3	4,000
	GCKSign-III	3	288	26	NCCSign-V (original)	5	4,402
10	GCKSign-V	5	544	27	Dilithium-V	5	4,864
11	SOLMAE-512	1	1,281	28	NCCSign-V(conserparam)	5	4,940
	Peregrine-512	1	1,281	29	MQSign-72-46	1	15,561
	FALCON-512	1	1,281	30	pqsigRM-612	1	24,592
14	HAETAE-II (revised)	2	1,376	31	MQSign-112-72	3	37,729
15	HAETAE-III (revised)	3	2,080	32	MQSign-148-96	5	66,421
16	NCCSign-I (original)	1	2,266	33	pqsigRM-613	5	331,074
17	SOLMAE-1024	5	2,305				

알고리즘 성능 측정 결과

- **테스트 환경**
  - Ubuntu 22.04, Ryzen 7 4800H, RTX 3060, 16GB RAM
  - Ubuntu 22.04, Intel i5-8259U, Coffee Lake GT3e, 16GB RAM
  - **IDE: Visual Studio Code**
    - 컴파일러가 없지만 대신 터미널이 내장된 편집기
    - 각종 확장을 통해 편리한 코딩이 가능
  - **Compiler: gcc 11.3.0**
  - **Optimization Level: O2와 O3 모두 테스트**



알고리즘 성능 측정 결과

- **Optimization Level에 대한 비교**
  - **효율성은 높을수록 좋음 (안전성면에서는 낮을수록 좋음)**

O2: space-speed tradeoff를 활용하지 않는 모든 최적화 지원

```
-falign-functions  -falign-jumps
-falign-labels  -falign-loops
-fcaller-saves
-fcode-hoisting
-fcrossjumping
-fcse-follow-jumps  -fcse-skip-blocks
-fdelete-null-pointer-checks
-fdevirtualize  -fdevirtualize-speculatively
-fexpensive-optimizations
-ffinite-loops
-fgcse  -fgcse-lm
-fhoist-adjacent-loads
-finline-functions
-finline-small-functions
-findirect-inlining
-fipa-bit-cp  -fipa-cp  -fipa-icf
-fipa-ra  -fipa-sra  -fipa-vrp
-fisolate-erroneous-paths-dereference
```

```
-flra-remat
-foptimize-sibling-calls
-foptimize-strlen
-fpartial-inlining
-fpeephole2
-freorder-blocks-algorithm=stc
-freorder-blocks-and-partition  -freorder-functions
-frerun-cse-after-loop
-fschedule-insns  -fschedule-insns2
-fsched-interblock  -fsched-spec
-fstore-merging
-fstrict-aliasing
-fthread-jumps
-ftree-builtin-call-dce
-ftree-loop-vectorize
-ftree-pre
-ftree-slp-vectorize
-ftree-switch-conversion  -ftree-tail-merge
-ftree-vrp
-fvect-cost-model=very-cheap
```

O3: O2보다 조금 더 최적화

```
-fgcse-after-reload
-fipa-cp-clone
-floop-interchange
-floop-unroll-and-jam
-fpeel-loops
-fpredictive-commoning
-fsplit-loops
-fsplit-paths
-ftree-loop-distribution
-ftree-partial-pre
-funswitch-loops
-fvect-cost-model=dynamic
-fversion-loops-for-strides
```

알고리즘 성능 측정 결과

- **알고리즘 정리 상황**
  - **Group 0: OpenSSL 의존성 제거, 모듈 통일화 작업 완료**
  - Group 1: OpenSSL 의존성 제거
  - Group 2: 작업 진행중 또는 진행 불가

	Working Group 0	Working Group 1	Working Group 2
PKE/KEM	IPCC NTRUplus PALOMA SMUAG (revised) TiGER	-	Layered-ROLLO (진행 중) REDOG (Python)
Digital Signature	AIMer (legacy) GCKSign HAETAE (revised) MQSign NCCSign Peregrine	SOLMAE pqsigRM	FIBS (느린 속도)

알고리즘 성능 측정 결과 (공개키 암호화)

- **공개키 암호 알고리즘의 성능 측정**
  - 측정 기준: 반복 10,000회, 최적화 옵션 -O2, -O3 사용
- **전체적인 성능 요약 (암/복호화 기준, -O3 기준)**
  - **TiGER > SMAUG > NTRU+ > PALOMA > ROLLO > IPCC**
  - **TiGER > SMAUG > PALOMA > NTRU+ > IPCC > ROLLO (AVX 패널티 적용: 성능*3)**
- **특이사항**
  - **IPCC:** IPCC-f1의 암호화 시간이 **느리며 일부 매개변수가 문서와 다름**
  - **NTRU+:** 측정한 알고리즘 중 **연산 속도가 빠름**
  - **SMAUG:** **연산 속도는 빠른 편이며 키 크기는 가장 작음**
  - **TiGER:** SMAUG와 비슷한 성능을 보이지만 **키 크기가 SMAUG에 비해 큰 편**
  - **PALOMA:** 키 생성에 **소요되는 시간이 매우 오래 걸림**
  - **ROLLO:** 암호화에 비해 복호화가 느린편 (-O2 설정은 측정 예정)
  - **REDOG:** **Python 의존성으로 인해 성능 측정 제외**

~~취소선~~: 성능 측정 중 다소 이상한 부분이 있어서 재측정 필요 (성능의 일관성 확인이 안됨)

녹색 이름 알고리즘: AVX 적용

Ryzen –O2

Unit: clock cycles

Algorithm	Keygen (Med.)	Encapsulation (Med.)	Decapsulation (Med.)	Keygen (Avr.)	Encapsulation (Avr.)	Decapsulation (Avr.)
IPCC_f1	14,362,627	~~164,892,550~~	2,484,981	14,376,006	~~239,300,607~~	2,501,514
IPCC_f3	14,170,647	898,710	2,619,570	14,178,752	941,012	2,633,907
IPCC_f4	14,209,594	1,075,059	2,904,524	14,245,778	1,135,740	2,935,161
NTRUplus-576	208,742	111,998	128,093	286,443	112,614	128,670
NTRUplus-768	279,386	148,480	181,250	298,193	154,346	185,298
NTRUplus-864	304,819	179,858	224,953	306,436	180,793	225,978
NTRUplus-1152	444,744	223,619	278,690	801,975	224,602	279,606
PALOMA-128	125,800,419	510,922	35,496	125,630,139	513,098	36,061
PALOMA-192	125,360,779	514,228	34,220	125,242,970	516,579	34,419
PALOMA-256	125,294,065	510,284	34,713	125,174,502	512,685	34,978
SMAUG-128 (revised)	171,477	154,483	178,205	181,007	156,512	181,950
SMAUG-192 (revised)	250,096	229,999	277,298	260,837	230,994	279,080
SMAUG-256 (revised)	479,138	385,178	438,364	490,702	387,420	439,345
TiGER-128	273,470	466,755	628,778	286,082	471,567	632,066
TiGER-192	288,550	518,491	674,192	293,000	524,467	691,838
TiGER-256	536,152	~~1,088,747~~	~~1,477,318~~	541,935	~~1,092,191~~	~~1,276,848~~

취소선: 성능 측정 중 다소 이상한 부분이 있어서 재측정 필요 (성능의 일관성 확인이 안됨)

녹색 이름 알고리즘: AVX 적용

Ryzen –O3

Unit: clock cycles

Algorithm	Keygen (Med.)	Encapsulation (Med.)	Decapsulation (Med.)	Keygen (Avr.)	Encapsulation (Avr.)	Decapsulation (Avr.)
IPCC_f1	13,940,097	~~160,111,204~~	~~16,360,164~~	13,969,607	~~232,561,407~~	2,408,173
IPCC_f3	13,996,024	926,492	2,512,836	14,036,005	967,543	2,532,438
IPCC_f4	13,989,832	1,106,031	2,714,531	14,007,544	1,165,274	2,732,139
NTRUplus-576	202,652	110,026	121,742	287,810	110,910	123,929
NTRUplus-768	270,512	146,566	174,435	281,685	147,174	175,018
NTRUplus-864	297,192	168,113	204,537	302,876	168,857	206,046
NTRUplus-1152	435,305	222,459	266,626	772,442	223,429	268,110
PALOMA-128	122,325,408	498,365	34,307	122,253,994	500,446	34,484
PALOMA-192	122,290,738	503,266	34,278	122,173,457	506,366	34,468
PALOMA-256	122,321,957	497,959	34,249	122,254,172	500,026	34,420
SMAUG-128 (revised)	72,790	57,246	50,460	82,292	57,466	50,708
SMAUG-192 (revised)	105,966	82,940	80,475	108,491	83,648	81,029
SMAUG-256 (revised)	158,021	139,925	135,749	161,110	141,573	138,010
TiGER-128	65,482	48,749	51,214	68,866	49,105	51,589
TiGER-192	69,426	63,510	57,739	79,105	63,805	59,383
TiGER-256	81,316	87,551	93,090	90,989	88,218	93,436
Layered ROLLO I-128	285,940	83,346	788,104	296,880	84,198	805,790
Layered ROLLO I-192	320,958	136,503	1,014,203	345,689	149,843	1,110,378
Layered ROLLO I-256	687,721	201,913	1,945,871	700,284	207,030	1,948,662

취소선: 성능 측정 중 다소 이상한 부분이 있어서 재측정 필요 (성능의 일관성 확인이 안됨)

녹색 이름 알고리즘: AVX 적용

Intel –O2

Unit: clock cycles

Algorithm	Keygen (Med.)	Encapsulation (Med.)	Decapsulation (Med.)	Keygen (Avr.)	Encapsulation (Avr.)	Decapsulation (Avr.)
IPCC_f1	13,792,887	~~159,126,951~~	1,196,157	13,896,694	~~231,010,613~~	1,259,215
IPCC_f3	13,754,219	870,059	1,235,991	13,864,988	922,755	1,307,538
IPCC_f4	13,754,687	1,050,451	1,318,173	13,851,205	1,151,306	1,380,740
NTRUplus-576	186,944	105,686	120,194	271,460	121,722	132,428
NTRUplus-768	246,616	139,310	166,938	265,516	154,868	174,788
NTRUplus-864	270,494	160,789	200,702	288,025	180,014	206,856
NTRUplus-1152	698,490	202,678	257,114	744,381	212,073	267,046
PALOMA-128	118,204,341	499,914	39,724	118,365,137	511,837	41,693
PALOMA-192	118,310,371	499,302	38,846	118,490,933	514,299	41,016
PALOMA-256	118,366,206	503,814	43,174	118,507,160	518,903	45,385
SMAUG-128 (revised)	158,149	164,598	196,470	165,414	169,648	203,288
SMAUG-192 (revised)	244,736	225,490	272,132	265,142	236,227	285,366
SMAUG-256 (revised)	435,790	411,917	465,572	448,654	422,602	486,263
TiGER-128	163,856	209,168	311,924	182,794	217,723	325,532
TiGER-192	171,578	214,126	312,702	181,774	221,613	324,412
TiGER-256	444,558	433,462	673,105	461,623	448,364	714,429

취소선: 성능 측정 중 다소 이상한 부분이 있어서 재측정 필요 (성능의 일관성 확인이 안됨)

녹색 이름 알고리즘: AVX 적용

Intel –O3

Unit: clock cycles

Algorithm	Keygen (Med.)	Encapsulation (Med.)	Decapsulation (Med.)	Keygen (Avr.)	Encapsulation (Avr.)	Decapsulation (Avr.)
IPCC_f1	12,643,392	~~145,233,220~~	1,159,273	12,712,124	~~210,977,105~~	1,185,580
IPCC_f3	12,795,377	874,663	1,206,585	12,874,291	922,533	1,267,783
IPCC_f4	13,078,917	1,037,485	1,310,503	13,250,237	1,107,017	1,368,035
NTRUplus-576	177,748	102,296	111,820	258,761	117,949	124,783
NTRUplus-768	239,546	137,135	161,970	257,560	165,057	177,077
NTRUplus-864	260,672	153,481	186,386	272,001	163,794	197,686
NTRUplus-1152	568,556	201,226	246,050	698,764	209,569	256,800
PALOMA-128	108,402,198	459,846	40,838	108,597,537	473,532	42,840
PALOMA-192	108,206,652	460,374	40,688	108,344,570	472,432	42,798
PALOMA-256	108,216,713	459,880	40,886	108,461,853	465,766	41,780
SMAUG-128 (revised)	63,020	49,324	39,196	65,919	55,873	42,528
SMAUG-192 (revised)	92,658	69,739	67,691	95,436	74,836	70,950
SMAUG-256 (revised)	135,202	122,766	115,096	142,842	128,734	118,789
TiGER-128	62,490	45,398	53,248	66,987	48,285	56,591
TiGER-192	66,512	60,238	58,572	71,626	71,973	71,967
TiGER-256	78,772	82,776	89,902	83,770	90,129	98,287
Layered ROLLO I-128	203,181	66,529	558,503	231,523	77,774	602,966
Layered ROLLO I-192	227,813	102,758	671,605	255,243	125,567	761,739
Layered ROLLO I-256	375,056	136,052	1,245,346	455,911	146,919	1,337,504

알고리즘 성능 측정 결과 (전자서명)

- **전자서명 알고리즘의 성능 측정**
  - 측정 기준: 반복 10,000회, 최적화 옵션 -O2, -O3 사용
    - pqsigRM은 연산속도 문제로 **100회 반복** 후 측정
- **전체적인 성능 요약 (서명 생성/검증 기준, -O3 기준)**
  - **Peregrine > SOLMAE> pqsig > GCKSign > MQSign > HAETAE > AIMer > NCC**
  - **Peregrine > pqsig > GCKSign > SOLMAE> HAETAE > MQSign > AIMer > NCC (AVX 패널티 적용: 성능*3)**

- **특이사항**
  - AIMer: 서명 생성/검증 성능은 느리지만 **키 생성이 빠르고 키 크기가 작음**
  - GCKSign: **검증 속도가 빠름**
  - MQSign: 서명 크기가 가장 작지만 **키 크기가 큼**, **특히 공개키 크기가 매우 큼**
  - Peregrine: 전체적인 성능이 우수하며 **검증 속도는 빠름**
  - pqsigRM: 보안강도 3 알고리즘의 **키 생성 속도가 느림**
  - FIBS: **무한 루프로 인해 측정 불가**

취소선: 성능 측정 중 다소 이상한 부분이 있어서 재측정 필요 (성능의 일관성 확인이 안됨) 녹색 이름 알고리즘: AVX 적용 Ryzen –O2 Unit: clock cycles

Algorithm	Keygen (Med.)	Sign (Med.)	Verify (Med.)	Keygen (Avr.)	Sign (Avr.)	Verify (Avr.)
AIMER-I	145,058	3,912,361	3,669,834	156,204	3,966,517	3,701,498
AIMER-III	296,496	8,001,274	7,550,063	315,810	8,033,590	7,548,322
AIMER-V	710,442	18,068,276	17,415,022	730,960	18,077,211	17,421,527
GCKSign-II	179,771	601,707	176,987	181,822	848,504	178,229
GCKSign-III	186,673	649,049	183,367	198,852	899,646	185,793
GCKSign-V	252,822	917,415	277,733	255,206	1,099,271	284,217
HAETAE-II (revised)	798,312	4,605,461	147,494	1,091,637	5,704,780	148,078
HAETAE-III (revised)	1,533,941	11,474,155	257,926	2,127,683	12,068,749	259,846
HAETAE-V (revised)	846,713	3,902,298	305,428	1,104,472	5,214,861	306,973
MQSign-72/46	94,788,559	516,954	1,461,281	94,829,257	518,651	1,465,923
MQSign-112/72	488,913,828	1,493,703	5,211,909	490,448,324	1,513,132	5,258,218
MQSign-148/96	1,488,480,956	3,162,943	12,036,827	1,488,377,972	3,164,654	12,041,118
NCCSign-II(con)	2,650,542	10,404,301	5,232,079	2,670,083	10,419,012	5,244,741
NCCSign-III(con)	4,477,513	17,657,839	8,867,243	4,497,436	17,666,605	8,869,094
NCCSign-V(con)	7,240,343	64,377,767	14,358,074	7,257,655	64,387,183	14,375,040
NCCSign-II(ori)	1,869,079	23,762,252	3,681,057	1,882,892	23,763,293	3,684,640
NCCSign-III(ori)	3,655,334	39,587,190	7,241,808	3,675,996	39,635,337	7,246,465
NCCSign-V(ori)	6,263,739	179,281,596	12,418,902	6,268,503	179,337,534	12,422,702
Peregrine-512	12,401,256	329,933	37,294	12,609,569	332,600	37,505
Peregrine-1024	39,405,505	709,848	80,243	42,160,344	722,426	81,200
pqsigRM-613	6,013,112,315	7,210,560	2,223,401	5,970,970,554	9,823,994	2,303,399
pqsigRM-612	58,238,108,879	1,864,512	1,053,034	58,669,322,672	2,650,133	1,064,763
SOLMAE-512	23,848,774	378,392	43,935	29,181,985	385,719	44,109
SOLMAE-1024	55,350,546	760,380	141,375	70,141,847	764,304	142,357

취소선: 성능 측정 중 다소 이상한 부분이 있어서 재측정 필요 (성능의 일관성 확인이 안됨) 녹색 이름 알고리즘: AVX 적용 Ryzen –O3 Unit: clock cycles

Algorithm	Keygen (Med.)	Sign (Med.)	Verify (Med.)	Keygen (Avr.)	Sign (Avr.)	Verify (Avr.)
AIMER-I	145,986	3,878,272	3,672,923	156,213	4,077,840	4,384,331
AIMER-III	296,032	8,087,462	7,678,098	307,498	8,364,809	7,740,701
AIMER-V	713,922	17,983,857	17,361,691	817,056	18,096,797	17,472,521
GCKSign-II	164,836	537,675	159,674	175,216	765,476	160,447
GCKSign-III	166,199	581,189	161,646	180,908	806,260	162,452
GCKSign-V	231,797	895,549	279,009	242,850	1,068,798	280,118
HAETAE-II (revised)	688,083	3,429,265	131,805	957,268	4,247,185	132,462
HAETAE-III (revised)	1,329,157	8,734,670	228,578	1,843,459	9,183,604	229,703
HAETAE-V (revised)	723,318	2,790,612	272,542	946,202	3,700,449	273,840
MQSign-72/46	39,040,917	311,112	512,227	39,057,616	312,293	514,042
MQSign-112/72	115,942,827	669,465	1,143,296	116,040,569	672,499	1,147,066
MQSign-148/96	235,289,035	1,186,622	1,943,667	235,425,321	1,190,984	1,952,355
NCCSign-II(con)	2,619,295	10,301,902	5,171,686	2,639,164	10,308,375	5,175,958
NCCSign-III(con)	4,379,261	86,475,941	8,685,877	4,405,049	86,515,726	8,685,125
NCCSign-V(con)	7,178,921	42,637,366	14,245,148	7,194,274	42,681,718	14,247,358
NCCSign-II(ori)	1,843,356	50,520,712	3,636,803	1,860,128	50,540,655	3,643,639
NCCSign-III(ori)	3,618,997	21,416,384	7,170,903	3,646,841	21,437,009	7,169,406
NCCSign-V(ori)	6,149,059	151,973,282	12,196,791	6,162,746	152,011,122	12,213,326
Peregrine-512	11,953,307	253,402	25,462	12,146,320	254,228	25,634
Peregrine-1024	38,366,232	535,920	53,621	41,014,591	538,260	53,946
pqsigRM-613	6,139,551,981	4,610,319	2,278,806	6,144,274,759	6,276,554	2,376,095
pqsigRM-612	54,994,439,928	714,647	225,577	55,073,661,751	967,439	234,553
SOLMAE-512	23,053,028	349,566	40,513	28,233,370	355,950	40,812
SOLMAE-1024	53,966,332	698,581	135,256	68,603,714	702,006	136,193

취소선: 성능 측정 중 다소 이상한 부분이 있어서 재측정 필요 (성능의 일관성 확인이 안됨) 녹색 이름 알고리즘: AVX 적용 Intel –O2 Unit: clock cycles

Algorithm	Keygen (Med.)	Sign (Med.)	Verify (Med.)	Keygen (Avr.)	Sign (Avr.)	Verify (Avr.)
AIMER-I	145,566	3,691,256	3,713,173	159,391	3,845,843	4,018,047
AIMER-III	274,358	7,771,108	7,366,672	304,919	7,863,536	7,438,953
AIMER-V	790,456	18,394,069	17,662,359	899,383	18,802,192	18,080,181
GCKSign-II	171,176	640,093	167,116	190,739	845,502	173,676
GCKSign-III	173,252	698,964	168,824	185,265	943,376	176,768
GCKSign-V	248,629	945,815	273,631	264,811	1,151,316	282,979
HAETAE-II (revised)	700,875	4,173,002	142,584	979,130	5,274,158	150,759
HAETAE-III (revised)	1,352,577	10,615,663	250,534	1,940,364	11,445,286	262,470
HAETAE-V (revised)	752,413	3,418,728	311,986	983,382	4,622,966	328,866
MQSign-72/46	87,038,447	509,630	1,377,392	87,156,508	527,234	1,411,202
MQSign-112/72	448,271,119	1,472,032	4,808,216	448,141,266	1,500,297	4,875,532
MQSign-148/96	1,326,638,494	3,128,536	11,091,036	1,328,649,536	3,150,219	11,143,601
NCCSign-II(con)	2,296,351	15,914,954	4,519,308	2,412,156	16,002,740	4,622,316
NCCSign-III(con)	4,009,717	16,015,734	7,996,462	4,169,703	16,116,734	8,085,000
NCCSign-V(con)	6,561,582	26,019,063	13,005,536	6,639,348	26,080,187	13,084,234
NCCSign-II(ori)	1,704,190	27,083,021	3,344,228	1,799,742	27,354,886	3,460,388
NCCSign-III(ori)	3,271,119	65,455,745	6,533,931	3,402,118	65,582,525	6,586,857
NCCSign-V(ori)	5,723,169	39,565,842	11,290,884	6,088,747	39,658,546	11,384,040
Peregrine-512	12,073,005	295,128	33,114	12,299,755	305,264	35,943
Peregrine-1024	38,493,479	640,132	71,246	41,112,188	652,620	74,891
pqsigRM-613	4,961,556,899	7,505,040	2,125,125	4,973,260,518	10,823,438	2,645,728
pqsigRM-612	74,021,054,015	2,113,913	1,126,131	73,941,690,821	2,765,068	1,295,161
SOLMAE-512	22,494,902	351,311	64,526	27,556,843	366,508	68,880
SOLMAE-1024	52,388,360	706,028	152,984	65,688,581	729,400	158,540

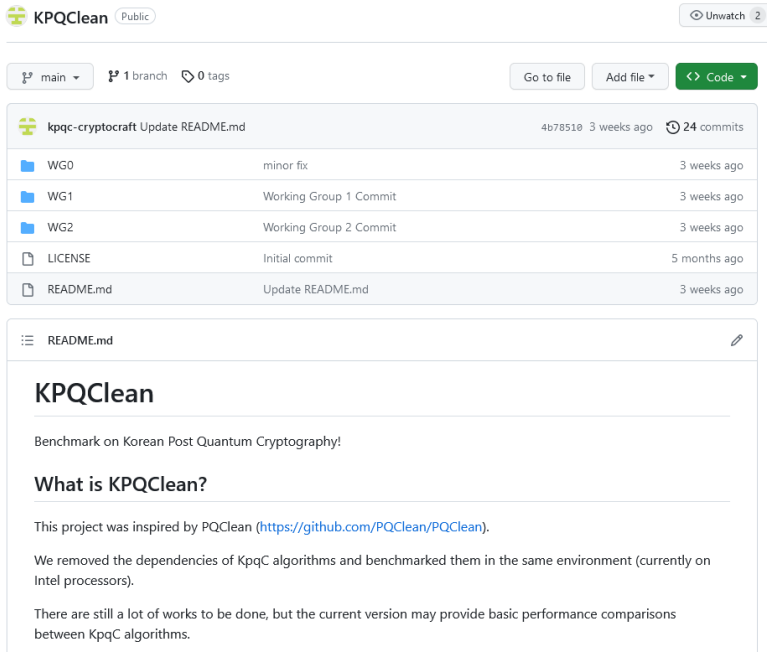
취소선: 성능 측정 중 다소 이상한 부분이 있어서 재측정 필요 (성능의 일관성 확인이 안됨) 녹색 이름 알고리즘: AVX 적용 Intel –O3 Unit: clock cycles

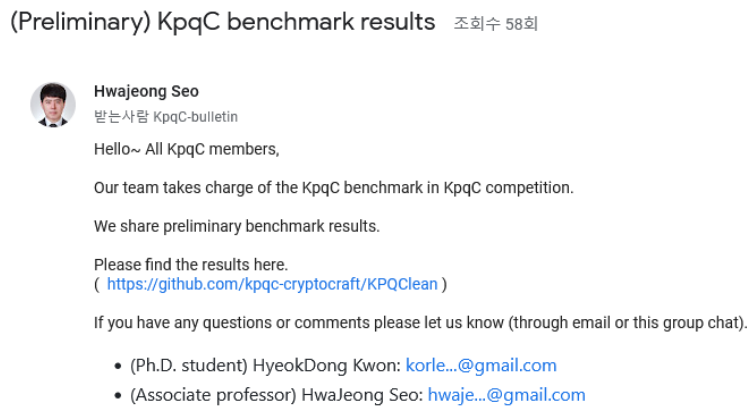
Algorithm	Keygen (Med.)	Sign (Med.)	Verify (Med.)	Keygen (Avr.)	Sign (Avr.)	Verify (Avr.)
AIMER-I	133,130	3,960,345	3,747,101	143,746	4,070,924	3,834,717
AIMER-III	272,484	8,440,184	7,968,982	282,896	8,530,553	8,041,509
AIMER-V	643,253	17,998,305	17,373,174	662,744	18,202,241	17,455,874
GCKSign-II	175,993	597,712	172,893	188,999	869,677	182,127
GCKSign-III	183,987	698,941	179,608	223,689	976,483	186,837
GCKSign-V	238,884	928,251	262,868	259,401	1,228,167	293,133
HAETAE-II (revised)	672,901	3,334,242	126,972	944,910	4,200,552	132,300
HAETAE-III (revised)	1,291,292	8,261,232	227,780	1,828,235	8,769,910	238,478
HAETAE-V (revised)	719,708	2,627,334	270,600	973,865	3,546,813	280,493
MQSign-72/46	38,474,591	298,952	533,676	38,612,360	308,203	547,680
MQSign-112/72	117,049,542	650,928	1,120,124	117,234,338	667,681	1,147,333
MQSign-148/96	236,124,011	1,165,706	1,897,664	236,332,422	1,173,558	1,908,458
NCCSign-II(con)	2,317,555	13,776,448	4,568,006	2,393,641	13,868,809	4,647,302
NCCSign-III(con)	3,981,551	83,521,123	7,935,382	4,209,101	83,634,184	8,001,129
NCCSign-V(con)	6,333,006	25,183,392	12,555,623	6,470,472	25,269,299	12,680,799
NCCSign-II(ori)	1,666,543	16,352,341	3,248,162	1,846,947	16,530,887	3,321,373
NCCSign-III(ori)	3,141,974	34,454,252	6,234,249	3,227,617	34,523,301	6,288,505
NCCSign-V(ori)	5,613,303	167,158,023	11,155,020	5,851,360	167,337,719	11,307,818
Peregrine-512	11,783,005	260,328	26,262	12,032,320	269,678	28,484
Peregrine-1024	37,875,534	551,168	55,654	40,364,494	569,794	58,474
pqsigRM-613	4,702,612,115	4,732,706	2,064,731	4,703,836,987	6,667,564	2,458,625
pqsigRM-612	71,111,088,778	923,513	417,658	71,168,430,985	1,166,665	502,448
SOLMAE-512	22,627,042	332,848	64,838	27,866,035	348,841	67,662
SOLMAE-1024	53,245,753	668,103	149,168	67,369,725	686,523	154,073

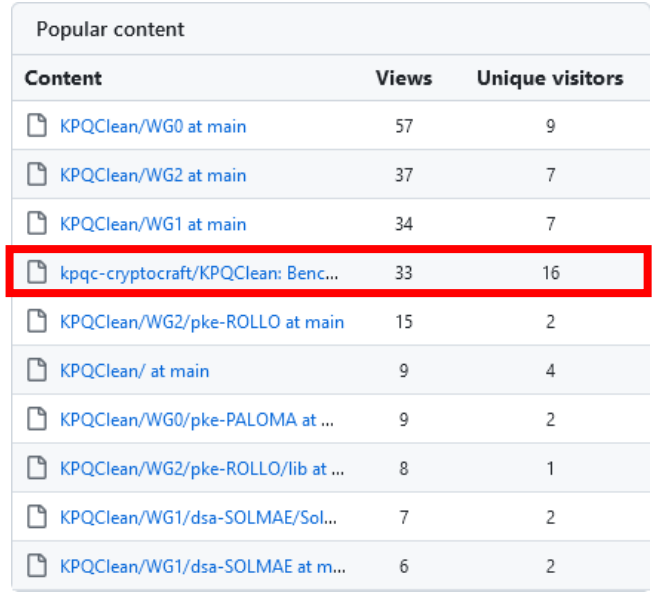
공개사항

- **KpqC github에 벤치마크 관련 모든 자료가 공개됨**
  - KpqC groups에도 공지가 됨
  - https://github.com/kpqc-cryptocraft/KPQClean
  - 아래 이메일을 통해 요청/건의사항을 저희 연구팀에게 연락주시면 확인 후 반영!

korlethean@gmail.com hwajeong84@gmail.com







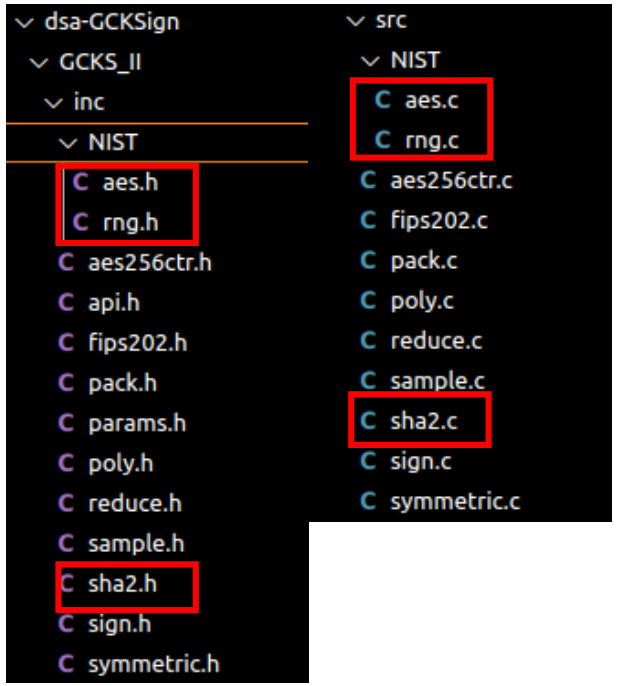
성능 측정을 위한 작업

- **자체 조사 결과 대부분의 알고리즘이 외부 라이브러리 의존성을 지님**
  - ROLLO, REDOG, Peregrine, HAETAE는 OpenSSL 라이브러리 의존성이 없음
- **의존성이 있는 알고리즘은 구현 시에는 용이하지만 실행에는 제약조건이 있음**
  - OpenSSL 설정 과정이 필요 → **사용자의 OpenSSL 관련 지식 요구**
  - 가동 환경마다 OpenSSL 연동 방법이 다름 → **문서화가 까다로움**
  - 서로 다른 버전의 라이브러리를 사용하는 경우 → **성능에 영향!**
- **PQC를 가동하는 입장에서 라이브러리는 불편한 요소로 작용**
  - 따라서 외부 **라이브러리 의존성을 제거하는 작업**을 먼저 수행
  - SSL에 의존하는 알고리즘: AES, SHA-2, SHA-3, SHAKE
  - OpenSSL 의존성을 제거하며 필요한 알고리즘들을 직접 삽입 (PQClean 참조)
  - **라이브러리 연동 없이 plain C를 통해 가동하는 것을 목표**
- **라이브러리 의존성을 제거해서 얻는 장점**
  1. 손쉬운 알고리즘 가동: 구현 환경 설정을 안하기에 누구나 알고리즘 가동 가능
  2. 외부 라이브러리 연동 실수 예방: 다른 버전의 라이브러리 사용이 발생하지 않음

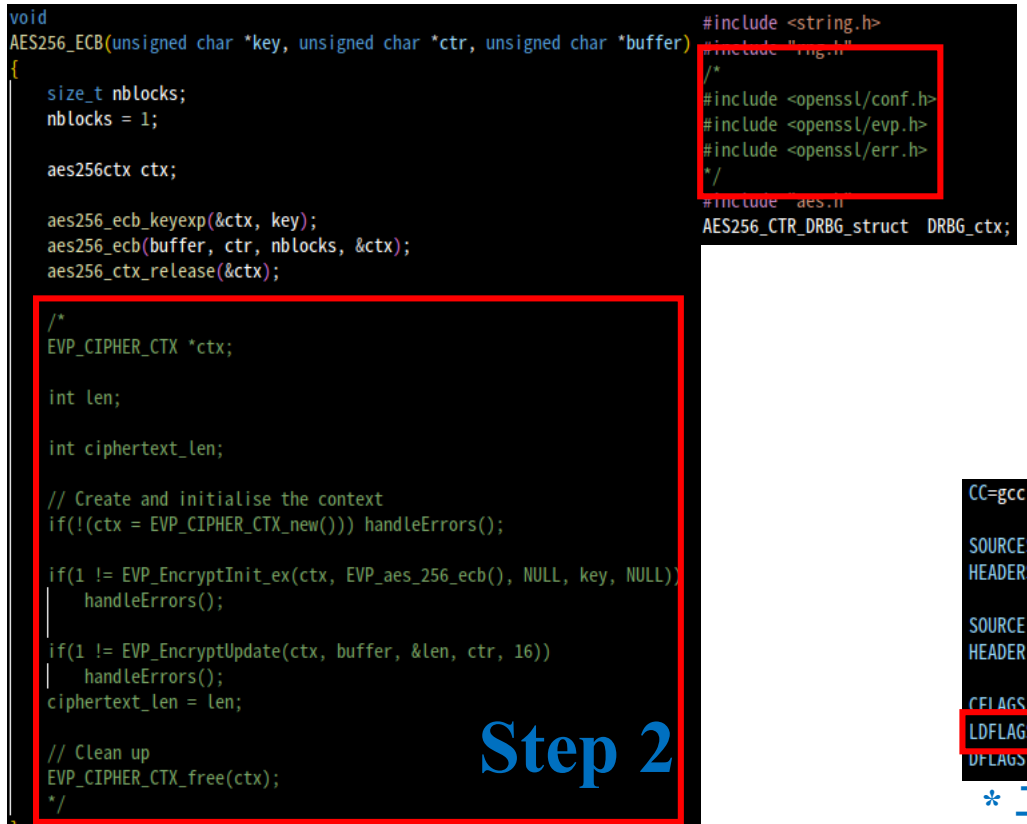
알고리즘명	Openssl 의존
공개키	
IPCC	Y
Layered ROLLO	N
NTRU+	Y
PALOMA	Y
REDOG	N
SMAUG	Y
TIGER	Y
전자서명	
MQSign	Y
GCKSign	Y
AIMer	Y
FIBS	Y
SOLMAE	Y
peregrine	N
HAETAE	N
pqsigRM	Y
NCC_Sign	Y

성능 측정을 위한 작업

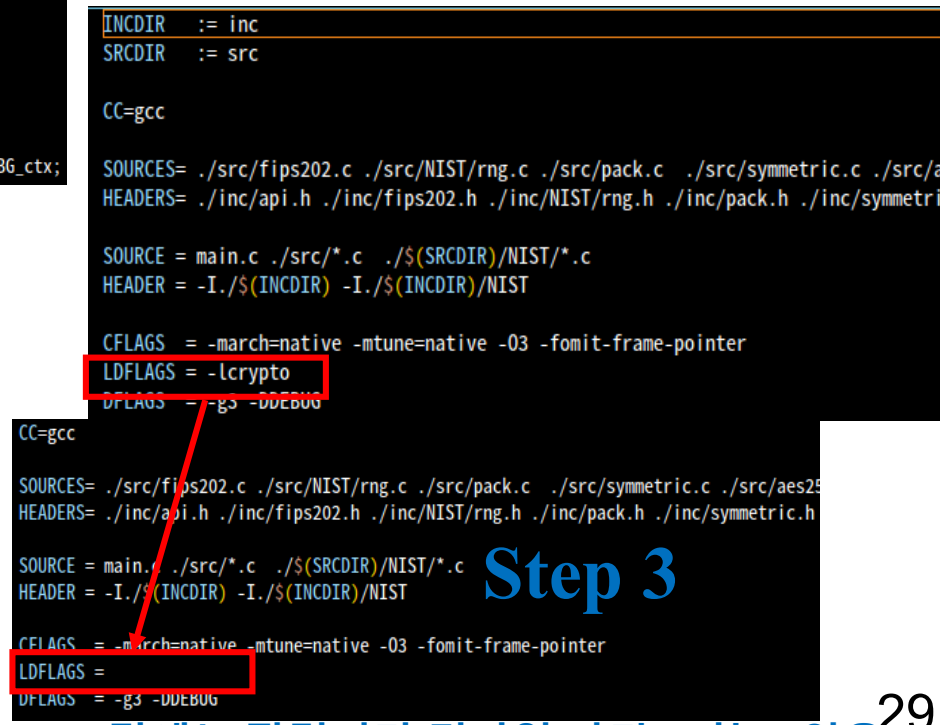
- **의존성은 어떻게 제거하는가?**
  - **알고리즘에 따라 방법이 전부 다름, 하지만 기본적인 골격은 거의 동일**
  1. 필요한 **알고리즘 소스 코드 파일 추가**
  2. 원본 소스코드에서 OpenSSL수정: **헤더 제거, 알고리즘 변경**
  3. Makefile 수정: OpenSSL **의존성 관련 규칙을 전부 제거**, 신규 파일을 **컴파일 타깃**에 포함



Step 1



Step 2



Step 3

* 그림에는 잘렸지만 컴파일 타깃 포함도 있음

성능 측정을 위한 작업

- **성능 측정용 코드 작성: PQC_bench.c**
  - CPU clock cycle을 측정
  - **반복적인 알고리즘 가동 후 평균 값**을 사용
  - **TEST_LOOP:** 반복 횟수
  - **cpucycles 함수:** 사이클을 계산하는 함수.
    Intel 명령어가 사용되기 때문에 **Intel CPU에 의존**
    따라서 **성능 측정은 모두 Intel 프로세서에서 수행**
  - **PQC_bench 함수:** 알고리즘을 가동하는 함수.
    알고리즘을 가동하며 실제 소요된 사이클을 측정 함

```
BENCHMARK ENVIRONMENTS  ============================
CRYPTO_PUBLICKEYBYTES: 1760
CRYPTO_SECRETKEYBYTES: 288
CRYPTO_BYTES: 1952
Number of loop: 10000
KeyGen ////////////////////////////////////////////////
  KeyGen runs in .................   191048 cycles
Sign //////////////////////////////////////////////////
  Sign runs in .................   812492 cycles
Verify ////////////////////////////////////////////////
  Verify runs in .................   175823 cycles
================================================
hd@hd-TFG7476HS:~/kpqclean-Round1/dsa-GCKSign/GCKS_II$
```

결과 화면

- 공개키, 개인키 크기
- 서명: 키 생성, 서명 생성, 검증 시간
- 암호: 키 생성, 암호화, 복호화 시간

```c
#define TEST_LOOP 10000

uint64_t t[TEST_LOOP];

int64_t cpucycles(void)
{
    unsigned int hi, lo;

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

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

int PQC_bench(void)
{
    int r;
    double rejw=.0, rejyz=.0, rejctr=.0, rejctrkg=.0;
    unsigned long long i, j;

    unsigned char pk[CRYPTO_PUBLICKEYBYTES];
    unsigned char sk[CRYPTO_SECRETKEYBYTES];

    unsigned char  m[100] = "kpqc benchmark system";
    unsigned char sm[CRYPTO_BYTES + 200];
```

성능 측정을 위한 작업

- **두 가지 항목에서 각각 두 가지 상황으로 분리**

1. 컴파일 옵션: **-O2 vs -O3**
   - **-O2는 컴파일러의 안정적인 최적화 방식**
   - -O3는 거의 모든 KpqC 알고리즘이 제시한 성능 측정 기준
   - 초기에는 –O2만 진행하였지만, **-O3를 추가로 제시함**
     - 알고리즘 개발자들의 의도를 반영하기 위함
2. 측정 값 설정: **평균값 vs 중앙값**
   - **평균값은 산술 평균으로 주어진 값 범위에서 일반적인 값 표현**
     - 연산 도중 급격히 느리거나 빠른 경우 평균 성능에 영향
   - **중앙값은 위치 평균으로 값 그룹을 두 부분으로 나누는 기준**
     - 속도 측정 중에 매우 느리거나 빠른 경우는 무시될 수 있음
   - 초기 성능 측정은 평균만 진행하였지만, **중앙값도 추가로 측정함**
     - KpqC 알고리즘은 대부분 중앙 값을 사용하므로 이 역시 개발자의 의도를 반영함

성능 측정을 위한 작업

- **Makefile 작성**
  - Makefile을 **처음부터 작성하지 않음**
  - 기존 투고된 **레퍼런스 코드의 Makeflie을 일부 수정**
  - 의존성 제거: OpenSSL 등 외부 종속성 제거
  - 신규 코드 타깃 추가: 새로 추가한 코드(주로 외부 알고리즘)를 타깃에 포함
  - 신규 규칙 추가: **PQC 벤치마킹과 모듈 벤치마킹 규칙 추가**
    - **벤치마크 옵션으로 -O2, -O3 서로 다른 경우를 포함**

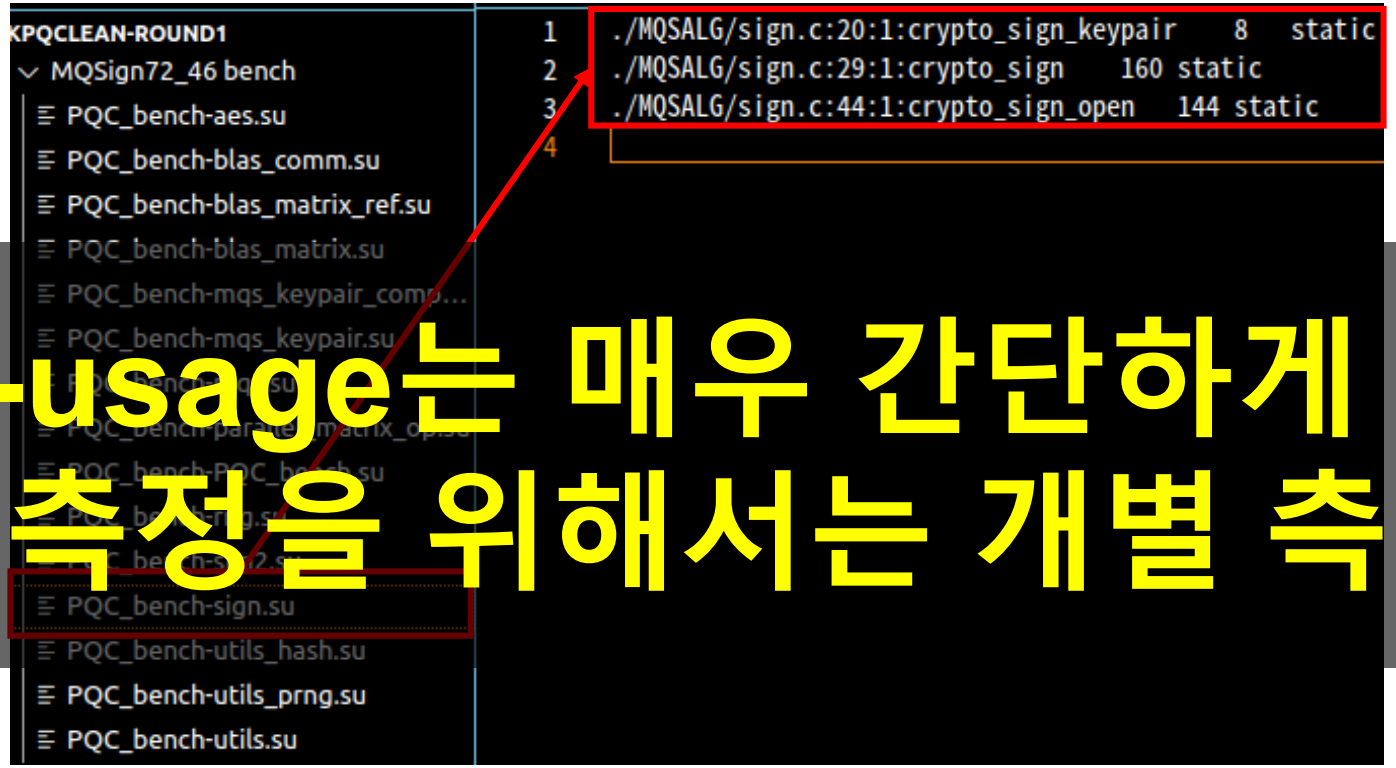
```
PQCgenKAT_sign: ./PQCgenKAT_sign.c
    $(CC) $(CFLAGS) -o $@ ./PQCgenKAT_sign.c $(HEADER) $(SOURCES) $(LDFLAGS)
    ./PQCgenKAT_sign

PQC_bench: ./PQC_bench.c
    $(CC) -march=native -mtune=native -O2 -fomit-frame-pointer -fstack-usage -o $@ ./PQC_bench.c $(HEADER) $(SOURCES) $(LDFLAGS)
    ./PQC_bench

Module_bench: ./Module_bench.c
    $(CC) -march=native -mtune=native -O2 -fomit-frame-pointer -o $@ ./Module_bench.c $(HEADER) $(SOURCES) $(LDFLAGS)
    ./Module_bench
```

성능 측정을 위한 작업

- **부가적인 벤치마크로 스택 사용량을 점검함**
  - 컴파일 명령어로 **‘fstack-usage’**를 사용
  - 해당 옵션 사용 시 **각 소스코드 파일마다 스택 사용량을 분석**한 파일을 생성
  - 이를 기반으로 스택 사용량을 체크



fstack-usage는 매우 간단하게 분석함
정밀한 측정을 위해서는 개별 측정 필요

특이사항

- 일부 알고리즘은 소스코드에 정의된 **파라미터가 문서와 다름**
  - **현재는 제공되는 각 알고리즘별 공식 문서를 따름**

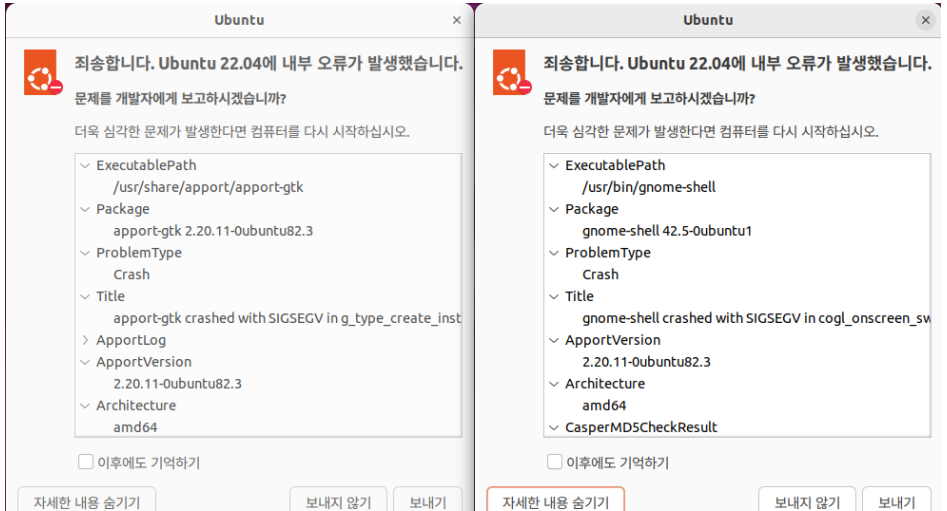
알고리즘	공개키(bytes)	암호문(bytes)	문서상 공개키	문서상 암호문
IPCC	3,600	322,000	4,800	92,000
	3,600	322,000	4,800	92,000
	3,600	322,000	4,800	92,000

알고리즘	공개키(bytes)	암호문(bytes)	문서상 공개키	문서상 암호문
TiGER	544	1,024	480	768
	1,056	2,048	800	1,024
	1,056	2,048	928	1,152

알고리즘	공개키(bytes)	비밀키(bytes)	문서상 공개키	문서상 비밀키
AIMer	33	49	32	16
	49	73	48	24
	65	97	64	32

알고리즘	공개키(bytes)	비밀키(bytes)	서명크기(bytes)	문서상 공개키	문서상 비밀키	문서상 서명크기
HAETAE	1,120	1,536	3,040	1,056	?	3,040
	1,664	2,272	4,544	1,568	?	4,064
	2,208	2,944	5,792	2,080	?	5,792

- 일부 알고리즘은 **가동이 불가**함
  - FIBS: 무한루프… 알고리즘이 멈추질 않음
  - PALOMA: 스택 제한을 풀면 가동 (-ulimit)
    → **그러나 VS code에서는 계속 오류**
    - **MQSign148에서도 동일한 문제 확인**



특이사항

- 간혹 **이상한 벤치마크 결과**가 도출됨
  - IPCC_f1의 암호화 알고리즘은 **다른 알고리즘에 비해 유난히 느림**
  - pqsigRM은 **키 생성이 지나치게 느림**
  - ~~스택 사이즈가 너무 작게 측정되는 경우~~
    - **fstack-usage의 단점으로, 스택을 측정한다면 직접 계산하는 것이 필요함**

알고리즘	보안레벨	키 설정(cycles)	암호화(cycles)	복호화(cycles)
IPCC	1	15,006,320	236,123,169	2,859,824
	3	15,411,372	942,886	2,829,830
	4	15,265,594	1,137,671	3,248,358

알고리즘	보안레벨	키 설정(cycles)	서명생성(cycles)	검증(cycles)
pqsigRM	612	6,175,797,126	14,737,341	2,441,347
	613	58,885,289,264	2,145,199	1,061,605

```
1   ./MQSALG/sign.c:20:1:crypto_sign_keypair    8   static
2   ./MQSALG/sign.c:29:1:crypto_sign    160 static
3   ./MQSALG/sign.c:44:1:crypto_sign_open    144 static
```

개선이 필요한 부분

- **알고리즘 성능 측정의 방법**
  - 현행 방법: **rdtsc 명령어로 CPU 클록 시간 차이를 통한 측정**
  - 10,000회 반복 값을 추출하여 평균과 중앙 값을 계산
  - 간편하고 유용하나, **장비 상태에 영향을 많이 받음**
    - 쓰로틀링, 전력 공급 상황 등
  - 더 정교한 측정 방법에 대한 고찰 필요

Keygen	Enc	Dec			
868115	507761	640378		Keygen 중앙값	273,470
823542	498481	630953		Enc 중앙값	466,755
695971	508979	628285		Dec 중앙값	628,778
546824	517505	629213		Keygen 평균값	286,082
546679	502048	629387		Enc 평균값	471,567
548158	487693	629329		Dec 평균값	632,066
547346	467045	629329			
547955	467741	628691			

- **Clean 라이브러리 확보**
  - **현재는 OpenSSL 라이브러리 의존성만 제거됨**
  - 추가적인 외부 라이브러리 의존성 제거가 필요
  - **어셈블리 명령어, AVX 등 병렬 명령어 제거도 필요**
    - 하지만 해당 명령어 제거는 많은 프로그래밍 노력이 요구됨

향후 작업 계획

1. 모듈 동일성 여부 판단

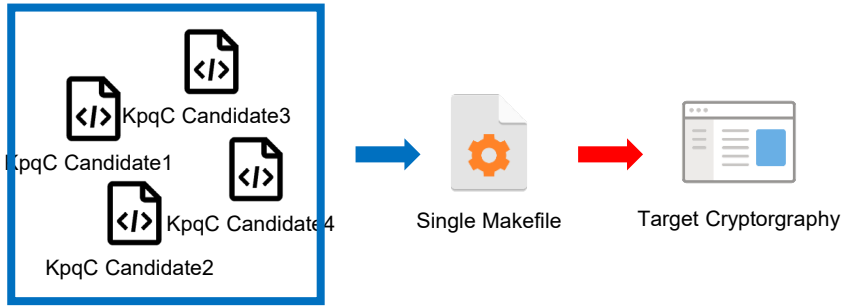
- 각 모듈(AES 등)이 **서로 다른 코드면 모듈 벤치마킹에서 유·불리** 발생
- 모든 알고리즘의 모듈을 동일한 코드로 통일
- 파라미터는 다를 수 있어도, **알고리즘 자체는 동일해야 함**

2. 동적할당 제거

- 동적할당의 경우 임베디드 장비 상에서의 동작 및 테스트에 어려움이 있음

3. 컴파일 규칙 통일

- 현재 작성된 Makefile 규칙은 알고리즘마다 다름
- 기존 레퍼런스 코드의 Makefile의 규칙을 수정한 것이기 때문
- **컴파일 규칙을 통일시켜서 완전히 동일한 컴파일 환경을 제공**



향후 작업 계획

4. 기타 의존성 제거

- 현재 파악된 각종 의존성은 다음과 같음

5. 스택 사용량 확인

- 제대로 측정 되었는지 점검

알고리즘명	Openssl 의존	Openssl 제거	Python 의존	Python 제거	어셈블리 의존	어셈블리 제거	동적할당 의존	동적할당 제거	C++ 의존	C++ 제거
공개키										
IPCC	Y	V	N	-	N	-	Y	X	N	-
Layered ROLLO	N	-	Y	X	N	-	N	-	N	-
NTRU+	Y	V	N	-	Y	X	Y	X	N	-
PALOMA	Y	V	Y(sage)	X	N	-	Y	X	N	-
REDOG	N	-	Y	X	N	-	N	-	Y	X
SMAUG	Y	V	Y(sage)	X	Y	X	Y	X	N	-
TIGER	Y	V	N	-	Y	X	Y	X	N	-
전자서명										
MQSign	Y	V	N	-	Y	X	Y	X	N	-
GCKSign	Y	V	N	-	Y	X	Y	X	N	-
AIMer	Y	V	N	-	Y	X	Y	X	N	-
FIBS	Y	V	N	-	Y	X	Y	X	N	-
SOLMAE	Y	V	N	-	Y	X	Y	X	N	-
peregrine	N	-	N	-	Y	X	Y	X	Y	X
HAETAE	N	-	N	-	Y	X	Y	X	Y	X
pqsigRM	Y	V	N	-	Y	X	Y	X	N	-
NCC_Sign	Y	V	N	-	Y	X	Y	X	N	-

Q & A