pqm4: Benchmarking NIST Additional Post-Quantum Signature Schemes on Microcontrollers

https://youtu.be/S2vhHb_rpL8

정보컴퓨터공학과 송경주

pqm4: Benchmarking NIST Additional Post-Quantum Signature Schemes on Microcontrollers

- 해당 문서는 5번째 PQC 워크샵에서 발표된 자료
- pqm4: 마이크로컨트롤러에서 NIST PQC 추가 서명 벤치마킹
  - **테스트 기본 플랫폼: Arm Cortex-M4(STM32L4R5ZI)**

- PQM4는 해당 플랫폼 환경에서 PQC가 원활히 동작하도록 함
  - **Keccak, SHA-2, AES를 최적화된 코드로 대체, 동적 메모리 할당 수정**

- 발표내용: NIST addition Signature의 초기 벤치마킹
  - 많은 서명들이 아직 Cortex-M4에 최적화되지 않았음
  - **대회 초반이므로 레퍼런스 구현 성능은 큰 의미가 없음**

임베디드 PQC

- 암호화는 **다양한 플랫폼**에서 잘 작동해야 함
- NIST additional Signatures: 작은 서명, 빠른 검증
  - 빠른 검증은 작은 플랫폼에서 특히 중요함
  - Dilithium의 성능은 큰 CPU에서 문제 없음

- NIST: 주요 마이크로 컨트롤러 최적화 대상으로 **Arm Cortex-M4 선택**
  - 강력한 명령어 집합 → UMAAL 및 더 많은 곱셈 명령어
  - 저렴하고 널리 사용 가능 → $20 개발 보드 (팬데믹이 아닌 경우)
  - 큰 코어 (640 KB SRAM 사용 가능) → 많은 PQC에 적합
  - 최적화하는 재미 → 단순한 파이프라인 덕분에 교육용으로도 좋음

- 발표내용: NIST addition Signature의 초기 벤치마킹
  - 많은 서명들이 아직 Cortex-M4에 최적화되지 않았음

삶의 이상과 현실

Ideal

- NIST PQC: 암호학자들은 좋은 레퍼런스 코드가 중요하다는 것을 깨달음
- **PQClean 같은 프로젝트가 품질 기준을 높임**
  - 유용한 SW 개발 도구에 대한 인식 제고
  - 다양한 플랫폼에서의 테스트 자동화
  - -Wall -Wextra -Wpedantic –Werror
- 깨달은 것을 바탕으로 논문 작성[1]
  - NIST 추천 목록 포함
- **Exception**
  - **2018년 보다 더 쉬운 작업환경 기대**

Real

- **NIST: 소프트웨어 요구사항 변경 없음**
- 많은 컴파일러 경고, 코드가 정화되지 않음
  - 많은 버그 발견
- 정적 메모리 사용
  - 큰 사전계산을 통해 성능을 속임
- 40개의 제출물 중 20개는 동적 메모리 할당 사용
  - 실제 필요 없이 종종 사용
- **Reality**
  - **NIST PQC I와 비교하여 아무런 개선도 이루어지지 않았음**

[1] Kannwischer, Matthias J., et al. "Improving software quality in cryptography standardization projects." 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). IEEE, 2022.

pqm4: Platform and framework 최신 변경사항

- **기본 플랫폼을 STM32L4R5ZI로 변경**
  - 640 KB RAM, 2 MB 플래시
  - STM32F407과 동일한 명령어 타이밍(메모리 로드 차이 제외)
  - 지원 플랫폼: STM32F407, STM32L476RG, STM32F303RCT7 (ChipWhisperer F3), MPS2-AN386 (qemu)

- **20% 더 빨라진 Keccak**
  - Adomnicai의 “An update on Keccak performance on ARMv7-M”에 설명됨
  - https://eprint.iacr.org/2023/773
  - https://github.com/mupq/pqm4/pull/254

알고리즘 제외 기준

- **취약성** : 알고리즘의 취약성
- **큰 공개키 크기** : 공개 키+개인 키+서명이 **640KB RAM** 이내여야 함
- **너무 큰 메모리 사용량** : 키 + 메모리 소비가 **640KB RAM** 이내여야 함
- **외부 라이브러리**: 외부 종속성을 가질 수 없음(예: gmp, flint)
  →**Keccak, SHA-2, AES를 최적화된 코드로 대체**

- **이식성 부족**: 32비트 플랫폼에서 지원되지 않는 코드 (예: __int128)
- **동적 메모리 할당** : 동적 메모리 할당은 적합하지 않음
  →**간단한 경우, 이를 수정하기 위해 노력 중**

Scheme inclusion

Code	Lattice	MPCitH	MQ	Other
CROSS	~~EagleSign~~	Biscuit	~~3WISE~~	AIMer
~~Enhanced pqsigRM~~	~~EHTv3 and EHTv4~~	MIRA	~~DME-Sign~~	~~ALTEQ~~
~~FuLeeca~~	HAETAE	MiRitH	~~HPPC~~	Ascon-Sign
~~LESS~~	HAWK	MQOM	MAYO	~~eMLE-Sig 2.0~~
MEDS	~~HuFu~~	PERK	~~PROV~~	FAEST
~~Wave~~	~~Raccoon~~	RYDE	~~QR-UOV~~	~~KAZ-SIGN~~
	~~SQUIRRELS~~	~~SDitH~~	SNOVA	~~Preon~~
			~~TUOV~~	SPHINCS-alpha
			UOV	~~SQIsign~~
			~~VOX~~	~~Xifrat1-Sign.I~~

취약성 (9) 큰 공개키 크기 (4) 너무 큰 메모리 사용량 (7) 이식성 부족/동적 메모리 할당 (3)

Dynamic memory allocations (동적 메모리 할당)

- **임베디드 구현에서 동적 메모리할당을 피해야 함**
  - 비용이 크고, 제대로 지원하지 않음
- 실제 동적 메모리 할당이 필요한 암호(4MB 이상 메모리 사용)는 적합하지 않음
- **pqm4: 동적 메모리 할당 허용x**
- 동적 메모리 할당 방식 사용(20개)
  - **Enhanced pqsigRM, LESS, Wave, SQIsign, EHTv3 and EHTv4, HuFu, MIRA, MQOM, RYDE, SDitH, PROV, QR-UOV, TUOV, VOX, AIMer, FAEST, ALTEQ, eMLE-Sig 2.0, KAZ-SIGN, Preon**
- 쉬운 수정: MQOM, AIMer
- ~~동적 메모리 할당 때문에 제외(3개): MIRA, RYDE, FAEST~~

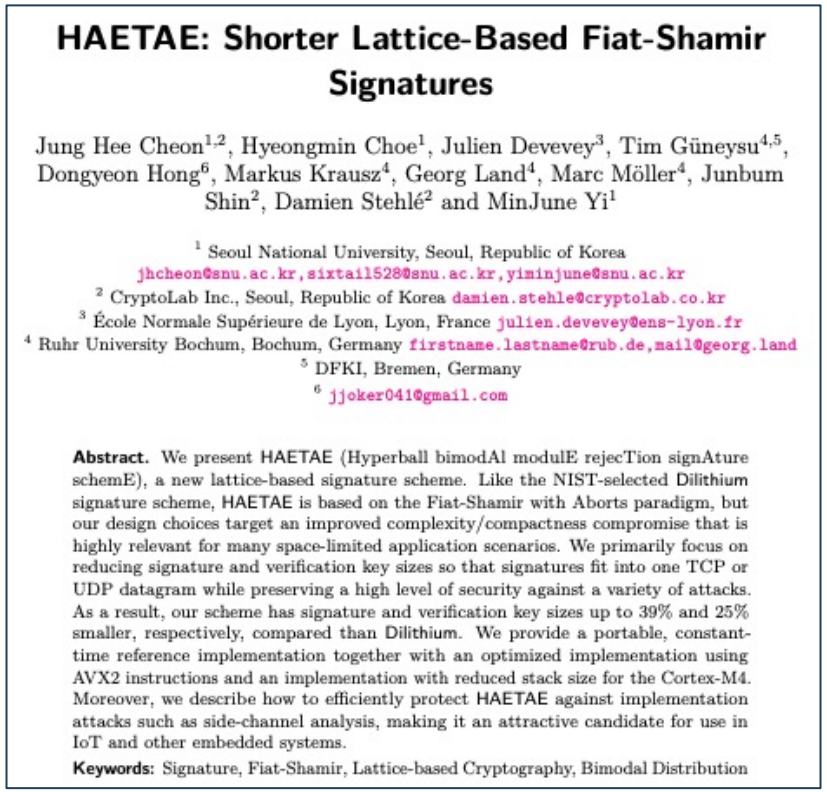
	PR	ref	m4f	params	eprint
CROSS	#309	✓		12/24	
MEDS	#324	✓		2/6	
HAETAE	#313	✓	✓	3/3	ia.cr/2023/624
HAWK	#305	✓		3/3	
Biscuit	#314	✓		3/6	
MiRitH	#315	✓	✓	16/32	ia.cr/2023/1666
MQOM	#322	✓		2/12	
PERK	#318	✓	✓	12/12	ia.cr/2024/088
MAYO	#302	✓	✓	3/4	ia.cr/2023/1683
SNOVA	#311	✓		7/18	
UOV	#300	✓	✓	3/12	ia.cr/2023/059
AIMer	#323	✓		3/12	
Ascon-Sign	#308	✓		8/8	
SPHINCS-alpha	#312	✓		6/24	
		14	5		

<Included implementation>

HAETAE

- **레퍼런스 코드와 M4 최적화 코드 모두 프로젝트에 통합됨**
- HAETAE 팀이 함께 기여함
- HAETAE: Shorter Lattice-Based Fiat-Shamir Signatures 에서 자세히 설명됨
  - https://eprint.iacr.org/2023/624

- Parameter sets: HAETAE-{2,3,5}
- 원래의 사양과 호환x



HAETAE: Shorter Lattice-Based Fiat-Shamir Signatures

Jung Hee Cheon[1,2], Hyeongmin Choe[1], Julien Devevey[3], Tim Güneysu[4,5], Dongyeon Hong[6], Markus Krausz[4], Georg Land[4], Marc Möller[4], Junbum Shin[2], Damien Stehlé[2] and MinJune Yi[1]

[1] Seoul National University, Seoul, Republic of Korea
jhcheon@snu.ac.kr, sixtail528@snu.ac.kr, yiminjune@snu.ac.kr
[2] CryptoLab Inc., Seoul, Republic of Korea damien.stehle@cryptolab.co.kr
[3] École Normale Supérieure de Lyon, Lyon, France julien.devevey@ens-lyon.fr
[4] Ruhr University Bochum, Bochum, Germany firstname.lastname@rub.de, mail@georg.land
[5] DFKI, Bremen, Germany
[6] jjoker041@gmail.com

Abstract. We present HAETAE (Hyperball bimodAl modulE rejecTion signAture schemE), a new lattice-based signature scheme. Like the NIST-selected Dilithium signature scheme, HAETAE is based on the Fiat-Shamir with Aborts paradigm, but our design choices target an improved complexity/compactness compromise that is highly relevant for many space-limited application scenarios. We primarily focus on reducing signature and verification key sizes so that signatures fit into one TCP or UDP datagram while preserving a high level of security against a variety of attacks. As a result, our scheme has signature and verification key sizes up to 39% and 25% smaller, respectively, compared than Dilithium. We provide a portable, constant-time reference implementation together with an optimized implementation using AVX2 instructions and an implementation with reduced stack size for the Cortex-M4. Moreover, we describe how to efficiently protect HAETAE against implementation attacks such as side-channel analysis, making it an attractive candidate for use in IoT and other embedded systems.

Keywords: Signature, Fiat-Shamir, Lattice-based Cryptography, Bimodal Distribution

MiRitH

- **레퍼런스 코드와 M4 최적화 코드 모두 프로젝트에 통합됨**
- MiRitH: Efficient Post-Quantum Signatures from MinRank in the Head에서 자세히 설명됨
  - https://eprint.iacr.org/2023/1666

- 레퍼런스: 32개의 매개변수 세트 중 16개 작동 중
- M4-optimized: mirith_hypercube_Ia_{fast,short}

MiRitH: Efficient Post-Quantum Signatures from MinRank in the Head

Gora Adj[1], Stefano Barbero[2], Emanuele Bellini[1], Andre Esser[1]*, Luis Rivera-Zamarripa[1], Carlo Sanna[2]†, Javier Verbel[1] and Floyd Zweydinger[1]

[1] Technology Innovation Institute, Abu Dhabi, UAE
{gora.adj,emanuele.bellini,andre.esser,
luis.zamarripa,javier.verbel,floyd.zweydinger}@tii.ae
[2] Politecnico di Torino, Torino, Italy
stefano.barbero,carlo.sanna@polito.it

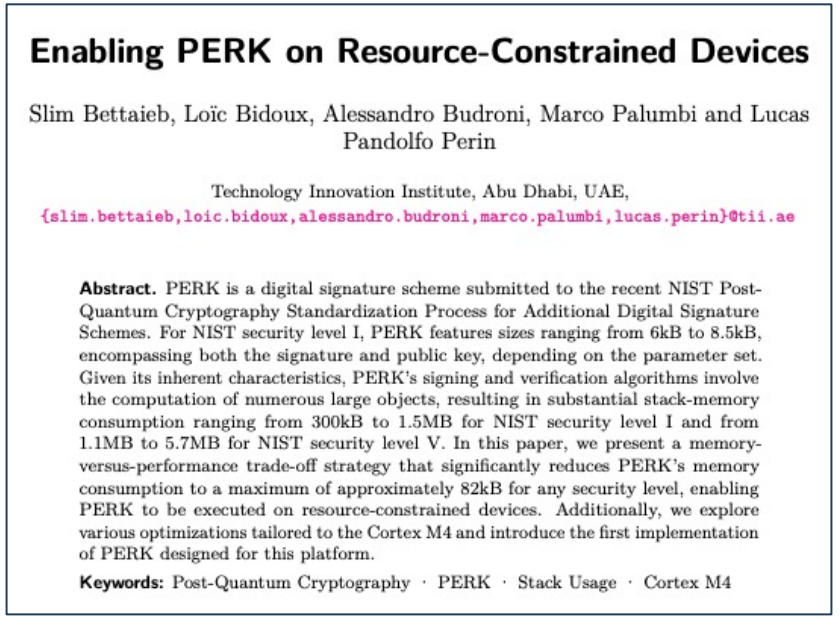
Abstract. Since 2016's NIST call for standardization of post-quantum cryptographic primitives, developing efficient post-quantum secure digital signature schemes has become a highly active area of research. The difficulty in constructing such schemes is evidenced by NIST reopening the call in 2022 for digital signature schemes, because of missing diversity in existing proposals. In this work, we introduce the new post-quantum digital signature scheme MiRitH. As direct successor of a scheme recently developed by Adj, Rivera-Zamarripa and Verbel (Africacrypt '23), it is based on the hardness of the MinRank problem and follows the MPC-in-the-Head paradigm. We revisit the initial proposal, incorporate design-level improvements and provide more efficient parameter sets. We also provide the missing justification for the quantum security of all parameter sets following NIST metrics. In this context we design a novel Grover-amplified quantum search algorithm for solving the MinRank problem that outperforms a naive quantum brute-force search for the solution.
MiRitH obtains signatures of size 5.7 kB for NIST category I security and therefore competes for the smallest signatures among any post-quantum signature following the MPCitH paradigm.
At the same time MiRitH offers competitive signing and verification timings compared to the state of the art. To substantiate those claims we provide extensive implementations. This includes a reference implementation as well as optimized constant-time implementations for Intel processors (AVX2), and for the ARM (NEON) architecture. The speedup of our optimized AVX2 implementation relies mostly on a redesign of the finite field arithmetic, improving over existing implementations as well as an improved memory management.

Keywords: Digital Signature · MinRank · MPCitH · Post-Quantum · ZKPoK · Quantum Analysis

PERK

- **레퍼런스 코드와 M4 최적화 코드 모두 프로젝트에 통합됨**
- PERK 팀이 함께 기여함
- Enabling PERK on Resource-Constrained Devices에서 자세히 설명함
  - https://eprint.iacr.org/2024/088

- 모든 12개의 매개변수 세트가 M4 구현에서 지원됨



Enabling PERK on Resource-Constrained Devices

Slim Bettaieb, Loïc Bidoux, Alessandro Budroni, Marco Palumbi and Lucas Pandolfo Perin

Technology Innovation Institute, Abu Dhabi, UAE,
{slim.bettaieb,loic.bidoux,alessandro.budroni,marco.palumbi,lucas.perin}@tii.ae

Abstract. PERK is a digital signature scheme submitted to the recent NIST Post-Quantum Cryptography Standardization Process for Additional Digital Signature Schemes. For NIST security level I, PERK features sizes ranging from 6kB to 8.5kB, encompassing both the signature and public key, depending on the parameter set. Given its inherent characteristics, PERK's signing and verification algorithms involve the computation of numerous large objects, resulting in substantial stack-memory consumption ranging from 300kB to 1.5MB for NIST security level I and from 1.1MB to 5.7MB for NIST security level V. In this paper, we present a memory-versus-performance trade-off strategy that significantly reduces PERK's memory consumption to a maximum of approximately 82kB for any security level, enabling PERK to be executed on resource-constrained devices. Additionally, we explore various optimizations tailored to the Cortex M4 and introduce the first implementation of PERK designed for this platform.

Keywords: Post-Quantum Cryptography · PERK · Stack Usage · Cortex M4

MAYO

- **레퍼런스 코드와 M4 최적화 코드 모두 프로젝트에 통합됨**
- Nibbling MAYO: Optimized Implementations for AVX2 and Cortex-M4에서 자세히 설명됨
  - https://eprint.iacr.org/2023/1683
- Supported parameter sets: MAYO{1,2,3} (MAYO5는 현재 메모리가 커서 지원되지 않음)
- 현재의 pqm4 구현은 round-1 사양과 호환됨
- 다른 방법을 사용할 때 약간 더 빠른 성능을 낼 수 있음 (Nibbling MAYO 발표)

Nibbling MAYO: Optimized Implementations for AVX2 and Cortex-M4

Ward Beullens[1], Fabio Campos[2], Sofía Celi[3], Basil Hess[1] and Matthias J. Kannwischer[4]

[1] IBM Research Europe, Zurich, Switzerland
[2] RheinMain University of Applied Sciences, Wiesbaden, Germany
[3] Brave Software
[4] Quantum Safe Migration Center, Chelpis Quantum Tech, Taipei, Taiwan†
contact@pqmayo.org

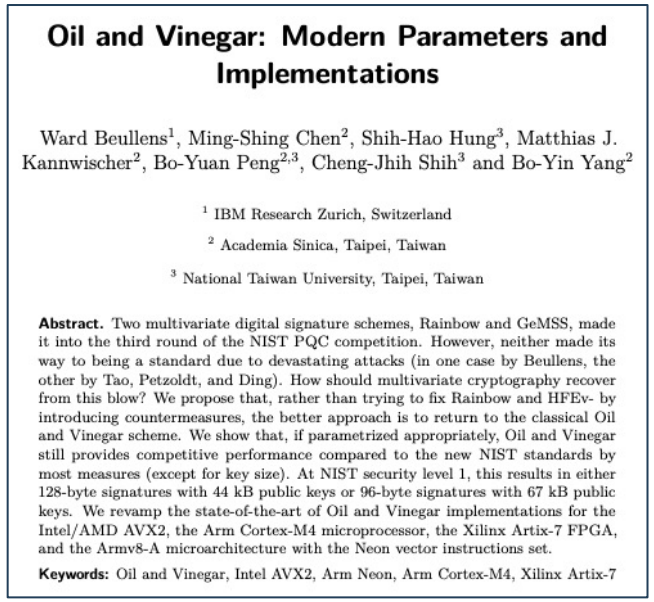
Abstract. MAYO is a popular high-calorie condiment as well as an auspicious candidate in the ongoing NIST competition for additional post-quantum signature schemes achieving competitive signature and public key sizes. In this work, we present high-speed implementations of MAYO using the AVX2 and Armv7E-M instruction sets targeting recent x86 platforms and the Arm Cortex-M4. Moreover, the main contribution of our work is showing that MAYO can be even faster when switching from a bitsliced representation of keys to a nibble-sliced representation. While the bitsliced representation was primarily motivated by faster arithmetic on microcontrollers, we show that it is not necessary for achieving high performance on Cortex-M4. On Cortex-M4, we instead propose to implement the large matrix multiplications of MAYO using the Method of the Four Russians (M4R), which allows us to achieve better performance than when using the bitsliced approach. This results in up to 21% faster signing. For AVX2, the change in representation allows us to implement the arithmetic much faster using shuffle instructions. Signing takes up to 3.2× fewer cycles and key generation and verification enjoy similar speedups. This shows that MAYO is competitive with lattice-based signature schemes on x86 CPUs, and a factor of 2-6 slower than lattice-based signature schemes on Cortex-M4 (which can still be considered competitive).

Keywords: MAYO · Oil and Vinegar · Arm Cortex-M4 · AVX2 · NIST PQC

UOV

- **레퍼런스 코드와 M4 최적화 코드 모두 프로젝트에 통합됨**
- Oil and Vinegar: Modern Parameters and Implementations에서 자세히 설명됨
  - https://eprint.iacr.org/2023/059

- Supported parameter sets: ov-Ip-{,pkc,pkc-skc}
- ov-Is는 640 KB의 RAM에 맞추기 위해 키를 플래시 메모리로 오프로드 해야함
- ov-III와 ov-V는 공개 키 크기 때문에 사용하기 어려움



Oil and Vinegar: Modern Parameters and Implementations

Ward Beullens[1], Ming-Shing Chen[2], Shih-Hao Hung[3], Matthias J. Kannwischer[2], Bo-Yuan Peng[2,3], Cheng-Jhih Shih[3] and Bo-Yin Yang[2]

[1] IBM Research Zurich, Switzerland

[2] Academia Sinica, Taipei, Taiwan

[3] National Taiwan University, Taipei, Taiwan

Abstract. Two multivariate digital signature schemes, Rainbow and GeMSS, made it into the third round of the NIST PQC competition. However, neither made its way to being a standard due to devastating attacks (in one case by Beullens, the other by Tao, Petzoldt, and Ding). How should multivariate cryptography recover from this blow? We propose that, rather than trying to fix Rainbow and HFEv- by introducing countermeasures, the better approach is to return to the classical Oil and Vinegar scheme. We show that, if parametrized appropriately, Oil and Vinegar still provides competitive performance compared to the new NIST standards by most measures (except for key size). At NIST security level 1, this results in either 128-byte signatures with 44 kB public keys or 96-byte signatures with 67 kB public keys. We revamp the state-of-the-art of Oil and Vinegar implementations for the Intel/AMD AVX2, the Arm Cortex-M4 microprocessor, the Xilinx Artix-7 FPGA, and the Armv8-A microarchitecture with the Neon vector instructions set.

Keywords: Oil and Vinegar, Intel AVX2, Arm Neon, Arm Cortex-M4, Xilinx Artix-7

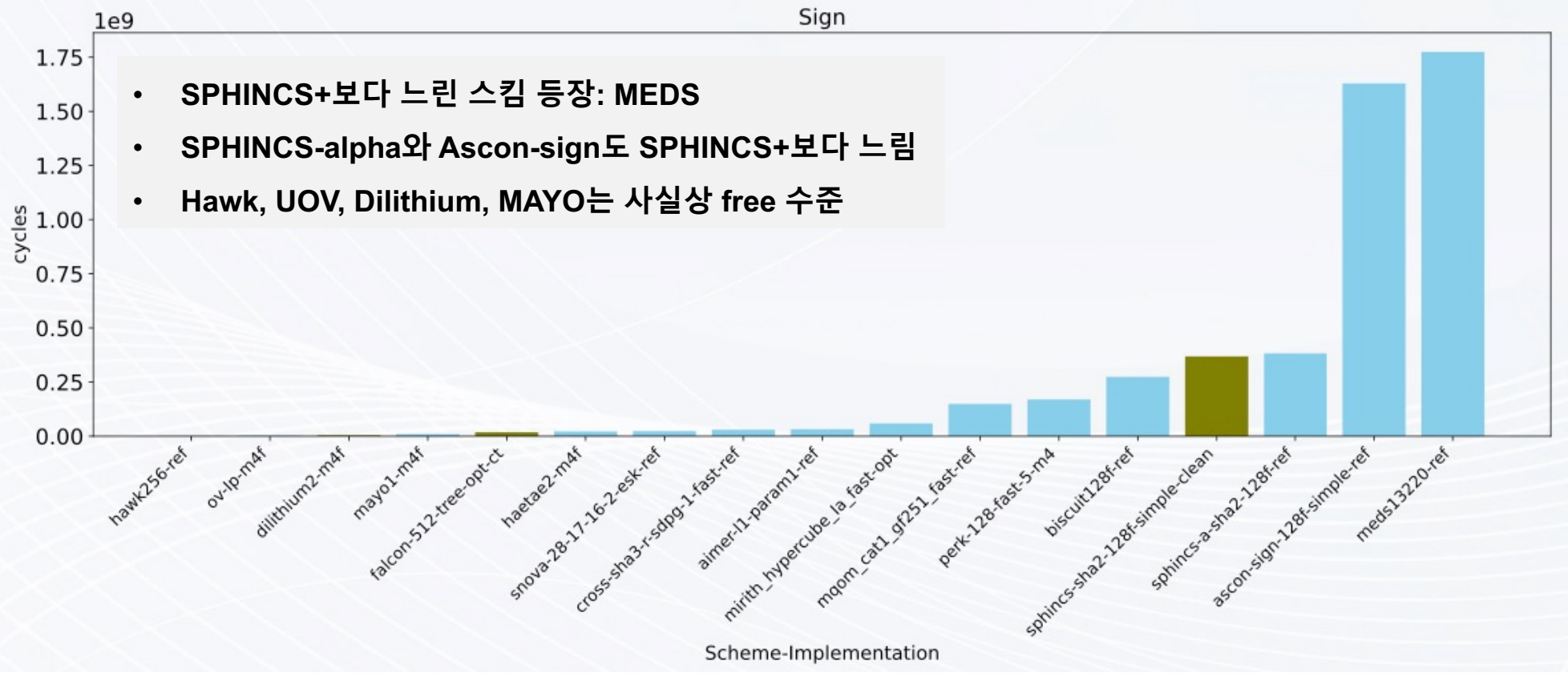
성능

- 몇가지 성능을 수치적으로 살펴봄
- 플래시 액세스 대기 상태를 피하기 위해 장치 클럭을 20 MHz로 낮춤
- 해당 자료에서는 제한된 몇 결과만 제공
  - 전체적인 결과는 pqm4 논문에서 확인가능
  - **Warning: 대회 초반이므로 레퍼런스 구현 성능은 큰 의미가 없음**

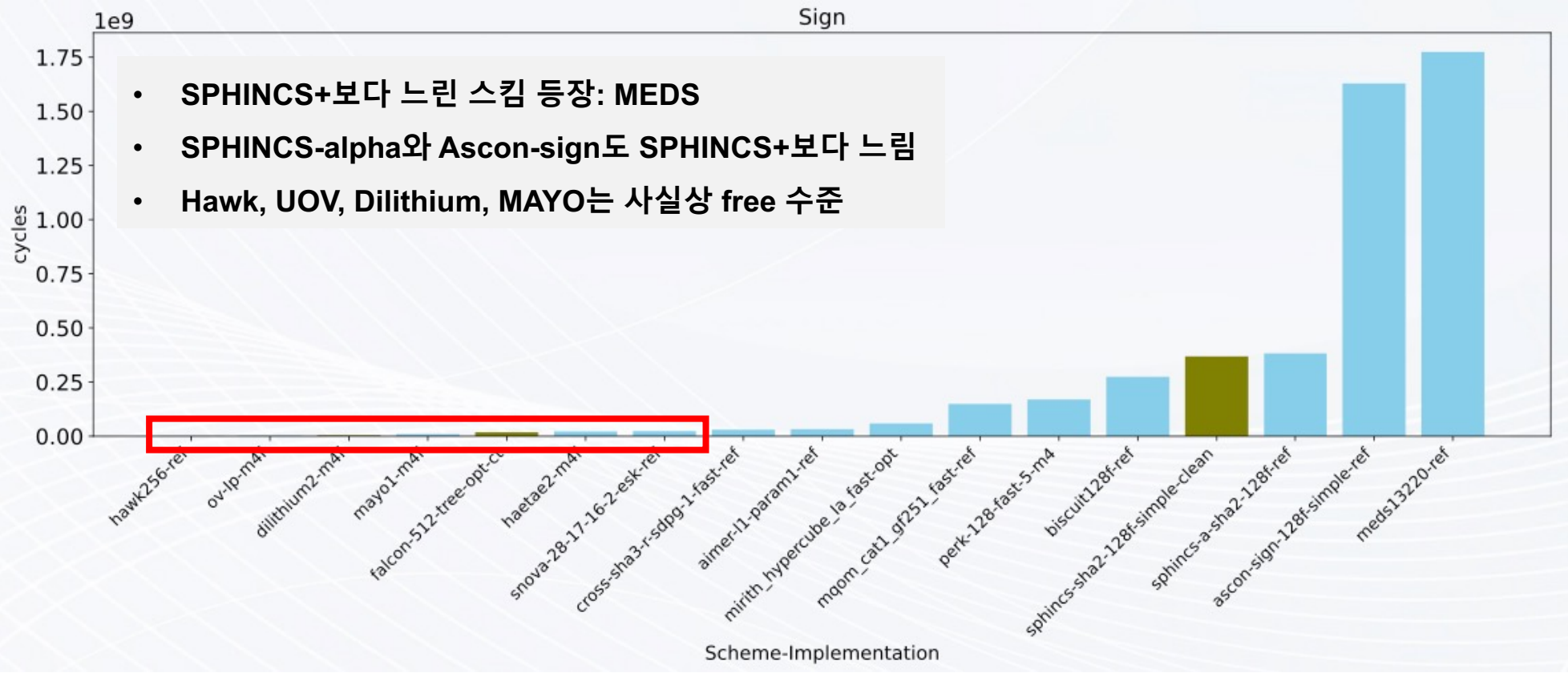
- **선택기준**
  - **각 제출물 중 가장 빠른 파라미터 세트 사용**
  - 서명 및 검증에 동일한 파라미터를 사용하지 않을 수 있음
  - 대부분 보안 수준 1 (예외: sphincs-a-sha2-192f)

- **추가: SQISign 검증**
  - Décio Luiz Gazzoni Filho와 Krijn Reijnders가 진행 중인 작업

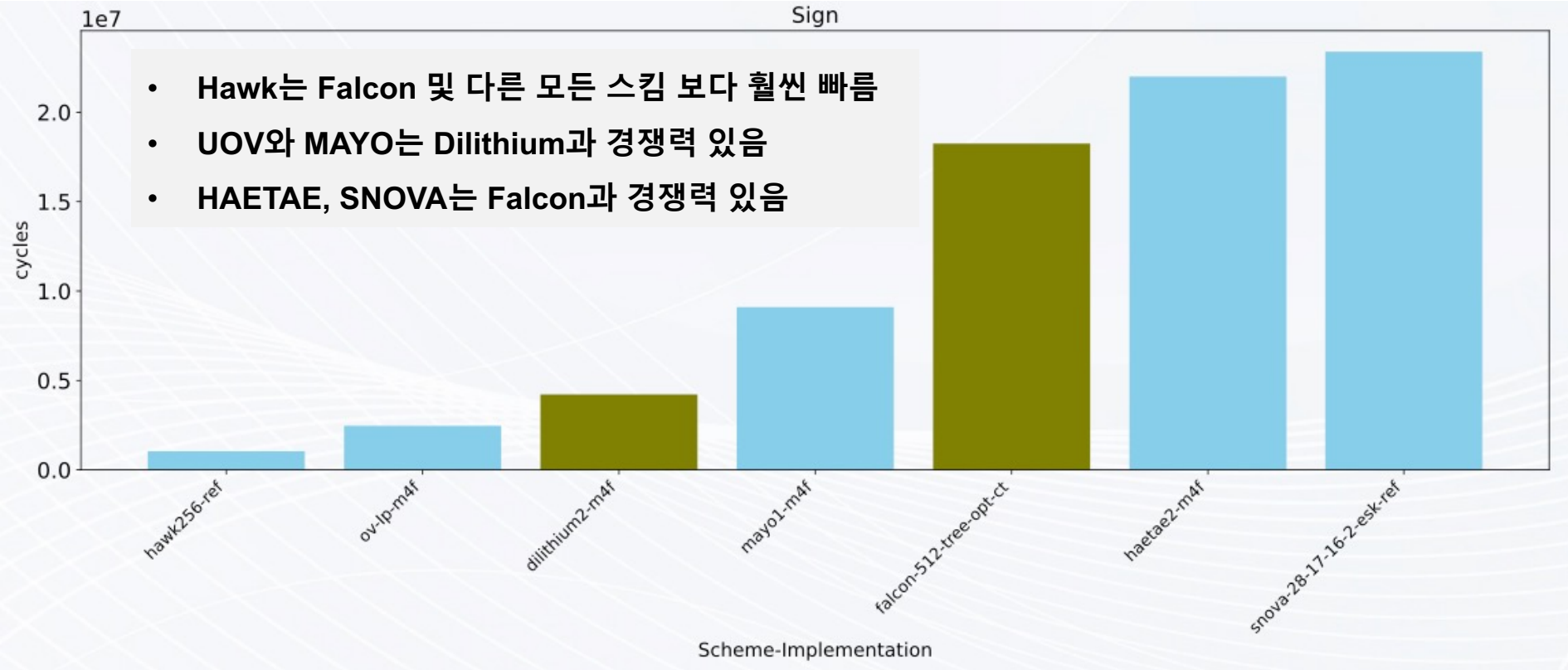
성능: 서명


Sign
1e9
cycles
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
• SPHINCS+보다 느린 스킴 등장: MEDS
• SPHINCS-alpha와 Ascon-sign도 SPHINCS+보다 느림
• Hawk, UOV, Dilithium, MAYO는 사실상 free 수준
hawk256-ref
ov-Ip-m4f
dilithium2-m4f
mayo1-m4f
falcon-512-tree-opt-ct
haetae2-m4f
snova-28-17-16-2-esk-ref
cross-sha3-r-sdpg-1-fast-ref
aimer-l1-param1-ref
mirith_hypercube_la_fast-opt
mqom_cat1_gf251_fast-ref
perk-128-fast-5-m4
biscuit128f-ref
sphincs-sha2-128f-simple-clean
sphincs-a-sha2-128f-ref
ascon-sign-128f-simple-ref
meds13220-ref
Scheme-Implementation

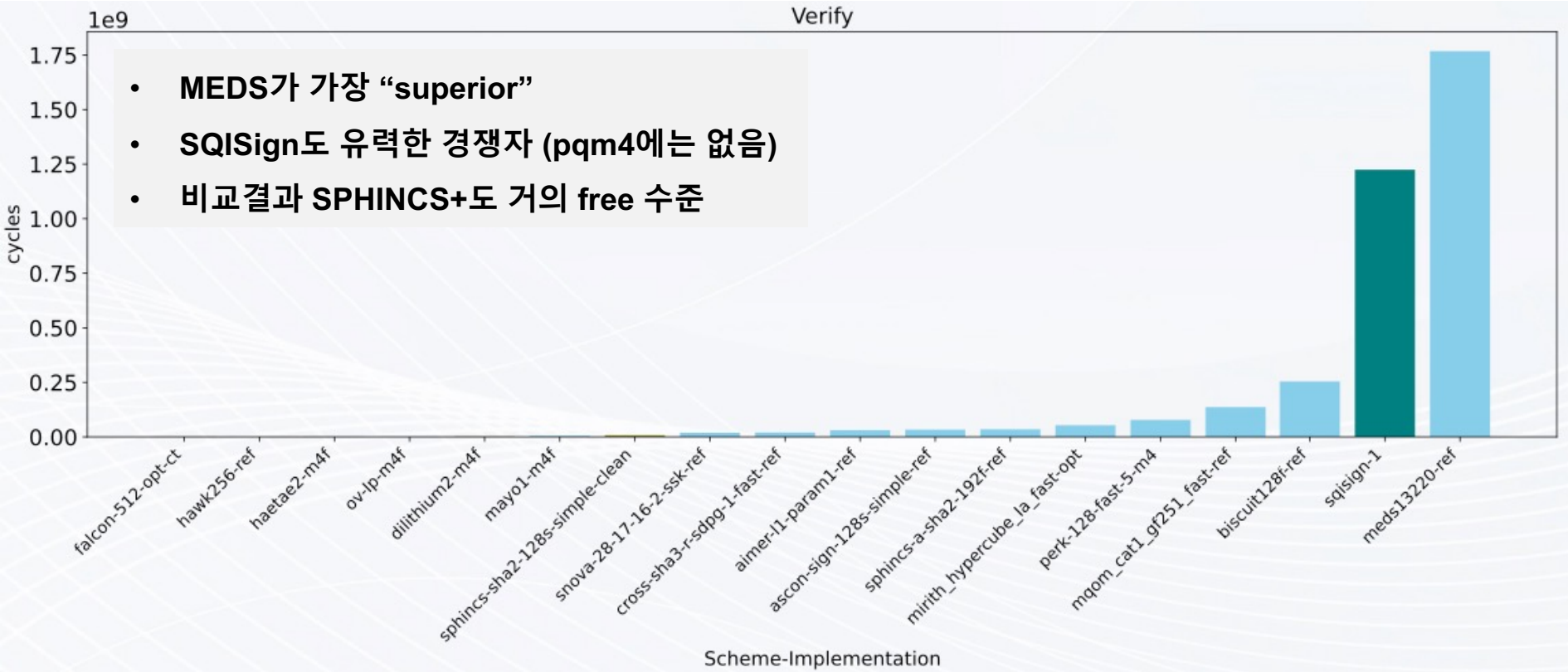
성능: 서명


Sign
1e9
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
cycles
• SPHINCS+보다 느린 스킴 등장: MEDS
• SPHINCS-alpha와 Ascon-sign도 SPHINCS+보다 느림
• Hawk, UOV, Dilithium, MAYO는 사실상 free 수준
hawk256-ref
ov-Ip-m4
dilithium2-m4
mayo1-m4
falcon-512-tree-opt-
haetae2-m4
snova-28-17-16-2-esk-ref
cross-sha3-r-sdpg-1-fast-ref
aimer-l1-param1-ref
mirith_hypercube_la_fast-opt
mqom_cat1_gf251_fast-ref
perk-128-fast-5-m4
biscuit128f-ref
sphincs-sha2-128f-simple-clean
sphincs-a-sha2-128f-ref
ascon-sign-128f-simple-ref
meds13220-ref
Scheme-Implementation

성능: 서명

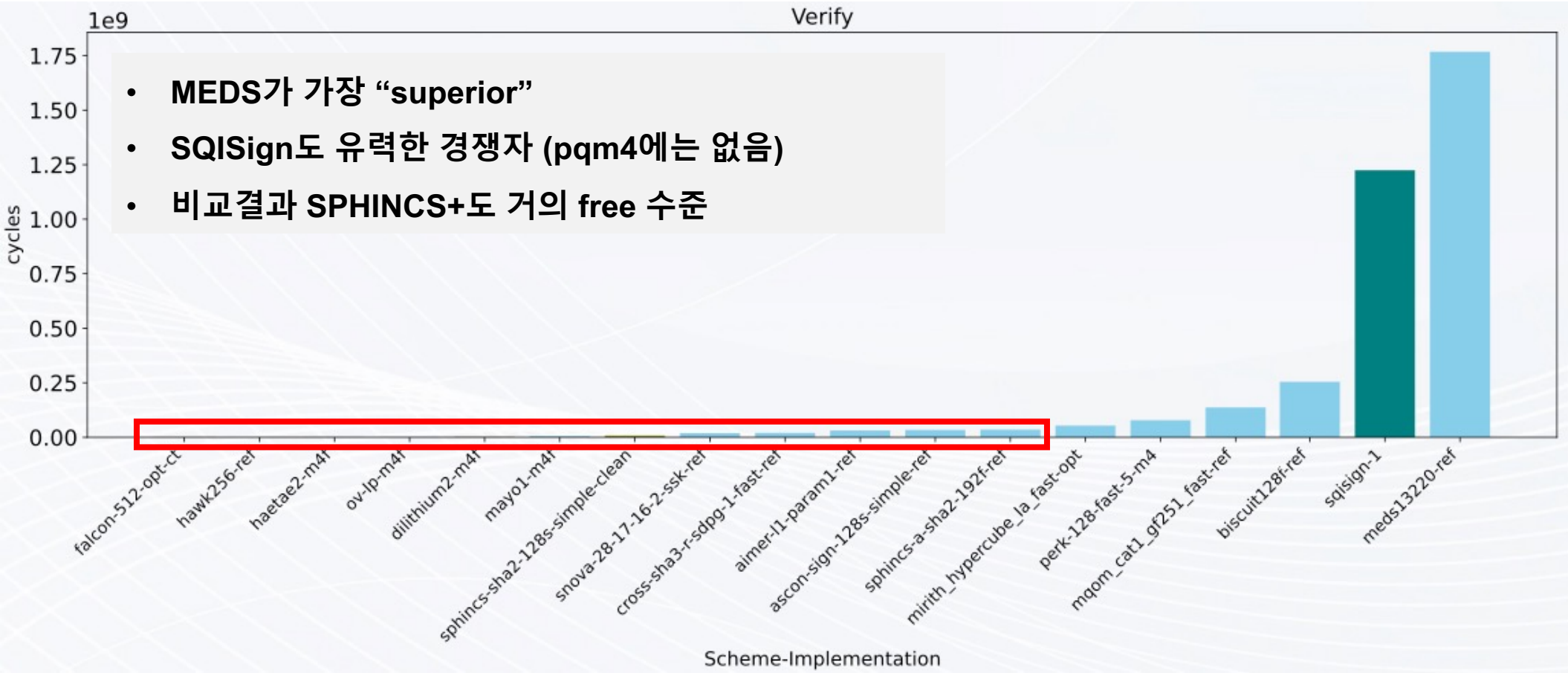

Sign
1e7
cycles
2.0
1.5
1.0
0.5
0.0
• Hawk는 Falcon 및 다른 모든 스킴 보다 훨씬 빠름
• UOV와 MAYO는 Dilithium과 경쟁력 있음
• HAETAE, SNOVA는 Falcon과 경쟁력 있음
hawk256-ref
ov-Ip-m4f
dilithium2-m4f
mayo1-m4f
falcon-512-tree-opt-ct
haetae2-m4f
snova-28-17-16-2-esk-ref
Scheme-Implementation

성능: 검증



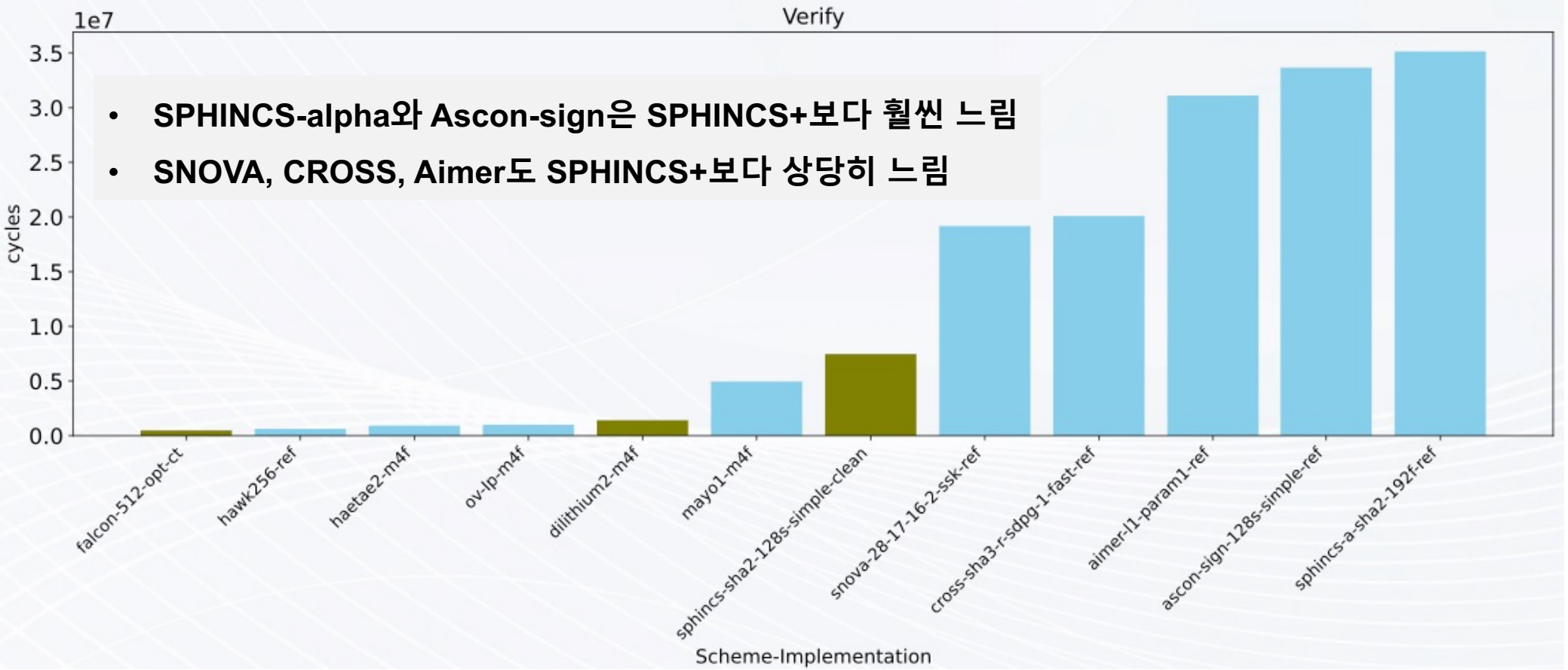
- **MEDS가 가장 “superior”**
- **SQISign도 유력한 경쟁자 (pqm4에는 없음)**
- **비교결과 SPHINCS+도 거의 free 수준**

성능: 검증

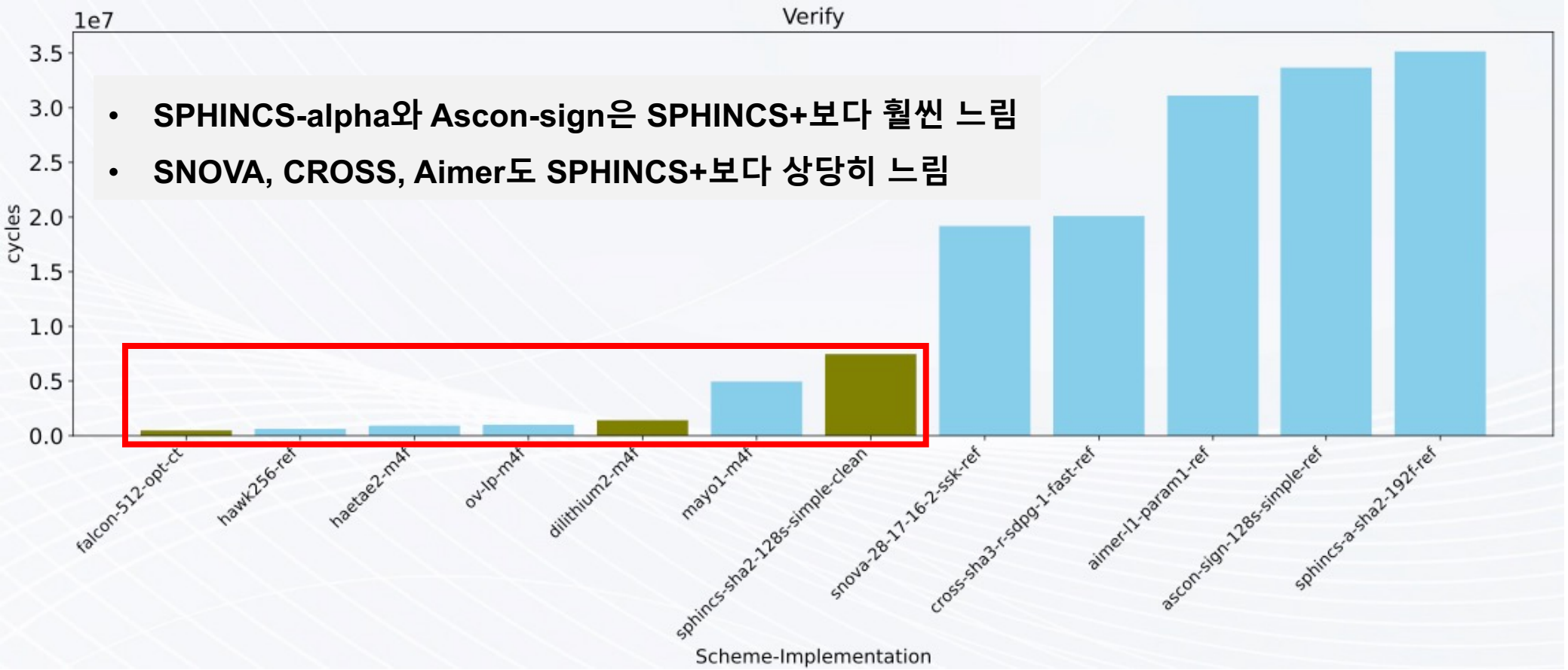


- **MEDS가 가장 “superior”**
- **SQISign도 유력한 경쟁자 (pqm4에는 없음)**
- **비교결과 SPHINCS+도 거의 free 수준**

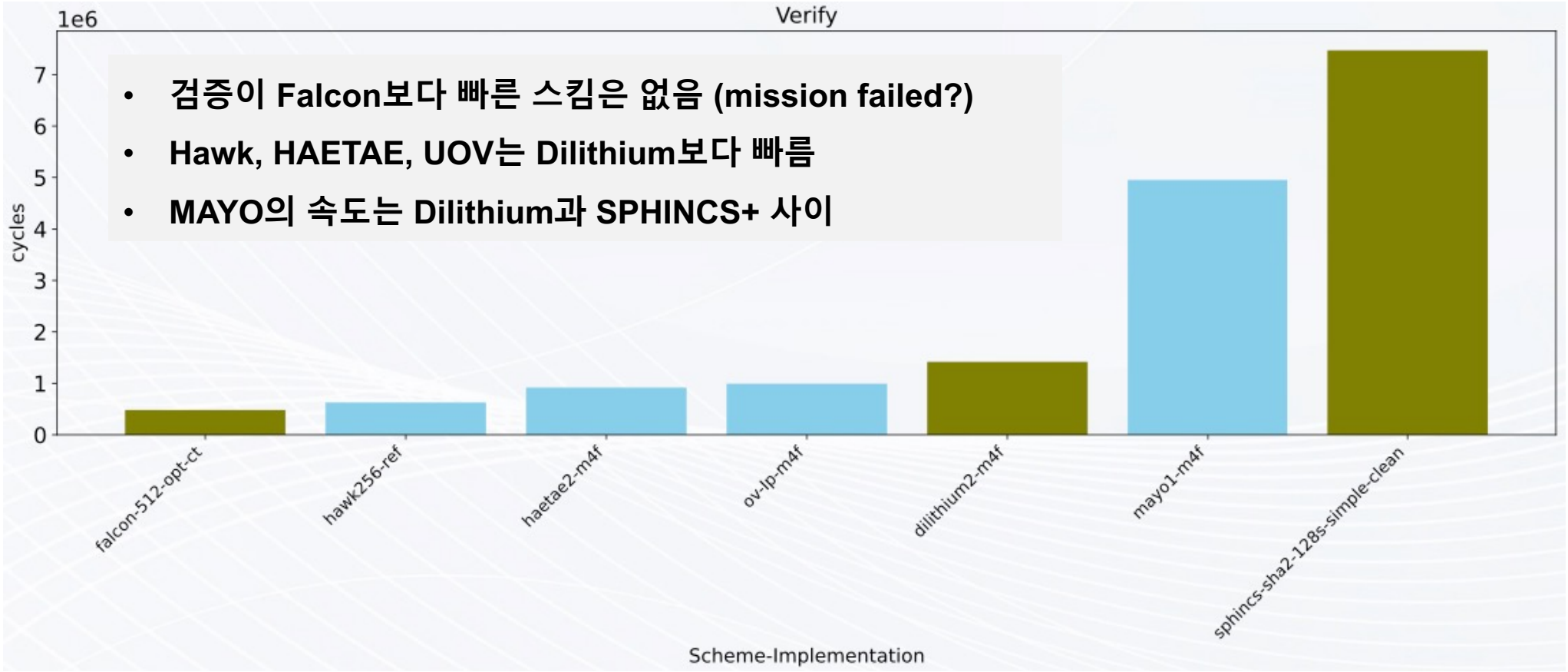
성능: 검증


Verify
1e7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
cycles
• SPHINCS-alpha와 Ascon-sign은 SPHINCS+보다 훨씬 느림
• SNOVA, CROSS, Aimer도 SPHINCS+보다 상당히 느림
falcon-512-opt-ct
hawk256-ref
haetae2-m4f
ov-Ip-m4f
dilithium2-m4f
mayo1-m4f
sphincs-sha2-128s-simple-clean
snova-28-17-16-2-ssk-ref
cross-sha3-r-sdpg-1-fast-ref
aimer-l1-param1-ref
ascon-sign-128s-simple-ref
sphincs-a-sha2-192f-ref
Scheme-Implementation

성능: 검증


Verify
1e7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
cycles
• SPHINCS-alpha와 Ascon-sign은 SPHINCS+보다 훨씬 느림
• SNOVA, CROSS, Aimer도 SPHINCS+보다 상당히 느림
falcon-512-opt-ct
hawk256-ref
haetae2-m4f
ov-Ip-m4f
dilithium2-m4f
mayo1-m4f
sphincs-sha2-128s-simple-clean
snova-28-17-16-2-ssk-ref
cross-sha3-r-sdpg-1-fast-ref
aimer-l1-param1-ref
ascon-sign-128s-simple-ref
sphincs-a-sha2-192f-ref
Scheme-Implementation

성능: 검증



- 검증이 Falcon보다 빠른 스킴은 없음 (mission failed?)
- Hawk, HAETAE, UOV는 Dilithium보다 빠름
- MAYO의 속도는 Dilithium과 SPHINCS+ 사이

결론 및 다음단계

- 일부 암호학자는 빠른 검증에 대해 이상하게 이해하고 있음
- 아직 대회 초기 단계
  - **제거해야 할 많은 low-hanging cycles가 아직 남아있음**

- Interesting question: pqm4에 포함되지 않은 스킴은 어떻게 여기에 맞출 수 있나?
  - **동적 메모리 할당 제거**
  - **외부 라이브러리 대체**
  - **메모리 소비 줄이기**

- 좋아하는 스킴이 포함되지 않았거나 느리다면
  - 메일로 화내지 마세요
  - 빠른 구현을 작성하고 풀 리퀘스트를 제출하세요

- **진행 중인 작업: 부분적 스킴 벤치마킹 (Ex: SQISign 검증)**
- **NIST: 후보 수를 대폭 줄여주세요.**