pqm4: NISTPQC Round 3 Results on the Cortex-M4

Matthias J. Kannwischer[1] and Richard Petri[2]
matthias@kannwischer.eu, rp@rpls.de

07 June, 2021, 3rd NIST PQC Standardization Conference

[1]Max Planck Institute for Security and Privacy, Bochum, Germany ⟶ Academia Sinica, Taipei, Taiwan
[2]Fraunhofer Institute for Secure Information Technology, Darmstadt, Germany

pqm4?

- https://github.com/mupq/pqm4
- pqm4 is a benchmarking and testing framework for NISTPQC candidates
- Targeting the Cortex-M4 (with all options)
- Our reference platform: STM32F407 (M4F, 128 KiB contiguous RAM, 1 MiB of flash)
  - You can get a discovery board for < $20
- pqm4 is also a collection of the fastest (open source) implementations
  - If you have something faster, please open a pull request
- pqm4 was presented at the 2nd NIST PQC standardization conference
  - Slides are available at https://kannwischer.eu/talks/20190824_nistpqc.pdf
  - This talk will focus on the changes since then

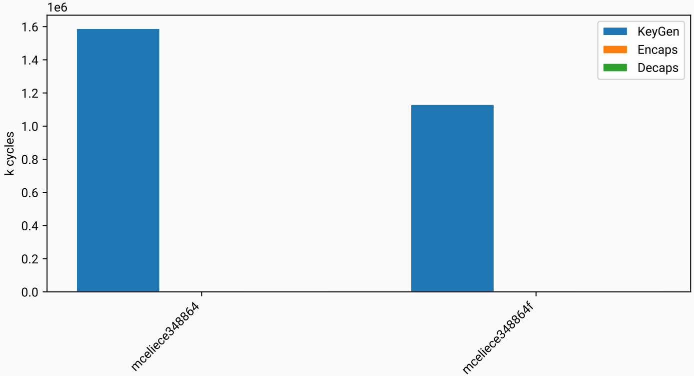
- Changed default AES implementation from t-table to bitsliced
  - Some Cortex-M4 platforms may have a cache → cache attacks possible
  - New bitsliced implementation by Adomnicai and Peyrin (ia.cr/2020/1123)
  - Slows down HQC, NTRUPrime, Kyber-90s
  - Implementations can still use faster t-table implementation for `_publicinputs()`
  - No change for FrodoKEM
- Multi-platform support
  - Improved and unified build system; fully based on `make` now; modular
  - Also supporting NUCLEO-L476R (STM32L476RG), ChipWhisperer LITEARM (STM32F303), QEMU simulator (ARM AN386)
  - Adding more hardware support should be easy
- Round 3 parameter changes for many of the candidates
- Various new record-breaking implementations merged

Implementation updates: KEMs

	pqm4	M4 implementation	new since Aug 2019	implementation
Classic McEliece	✗	✓	✓	ia.cr/2021/492
KYBER	✓	✓	✓	ia.cr/2020/012
NTRU	✓	✓	✓	ia.cr/2020/1397
SABER	✓	✓	✓	ia.cr/2020/1397
BIKE	✓	✓	✓	ia.cr/2021/493
FrodoKEM	✓	✓	✗	ia.cr/2018/1116
HQC	✓	✗	✗	ref
NTRU Prime	✓	✓	✓	ia.cr/2020/1216
SIKE	✓	✓	✓	ia.cr/2021/115

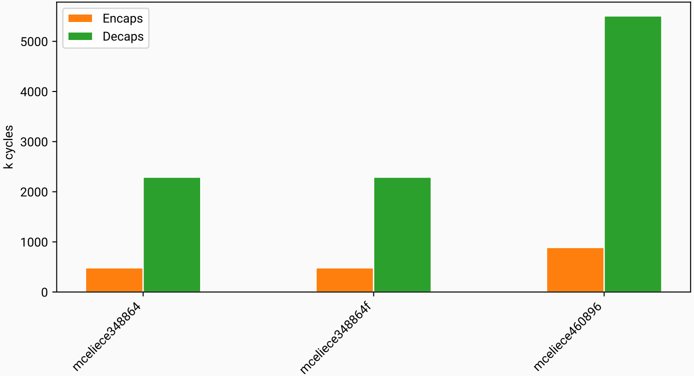
✓: outdated implementation in pqm4

Classic McEliece

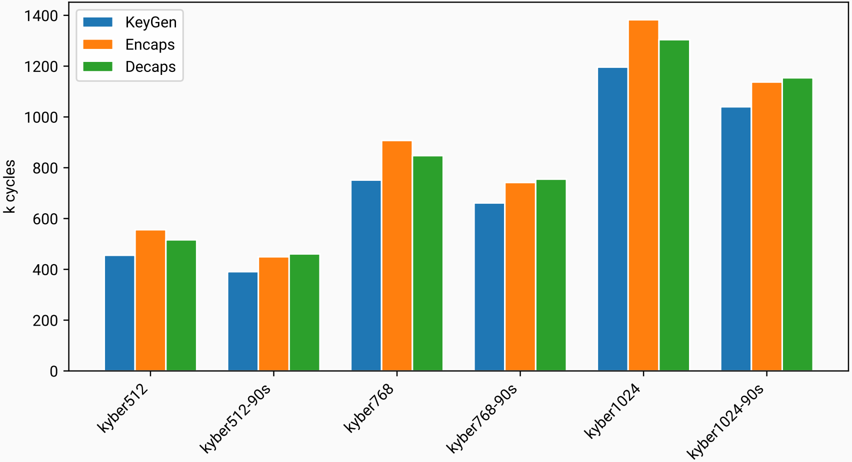


- Keys do not fit in RAM of our target (except the simulated platform)
- New implementation by Chen and Chou (ia.cr/2021/492)
  - Write keys to flash instead
- Not suitable for pqm4

Classic McEliece

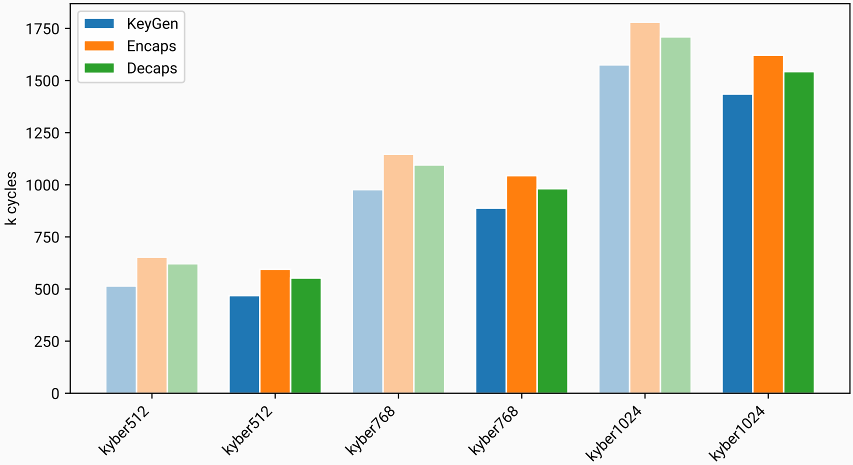


- Keys do not fit in RAM of our target (except the simulated platform)
- New implementation by Chen and Chou ([ia.cr/2021/492](ia.cr/2021/492))
  - Write keys to flash instead
- Not suitable for pqm4

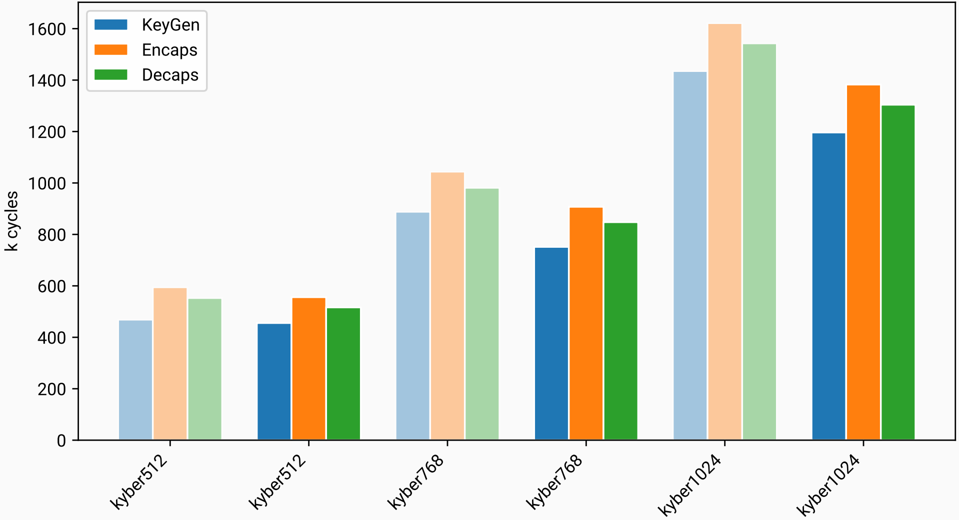


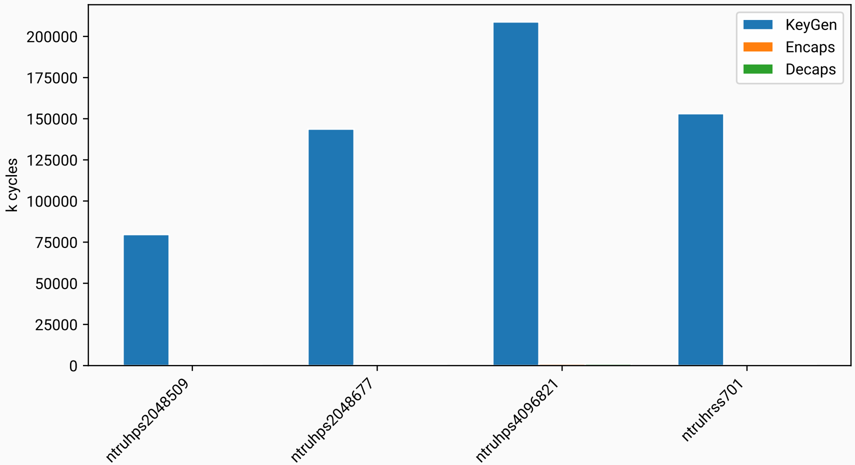
- Jan 2020: Faster implementation by Alkim, Bilgin, Cenk, and Gérard (ia.cr/2020/012)
- Jul 2020: New round 3 parameter sets

CRYSTALS-KYBER (ia.cr/2020/012)



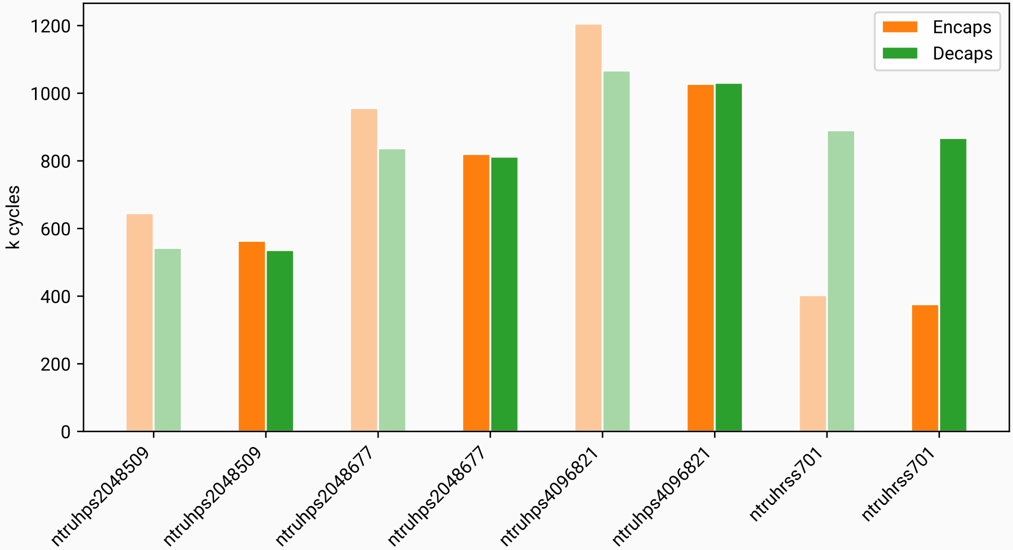
CRYSTALS-KYBER (Round 2 vs. Round 3 parameters)

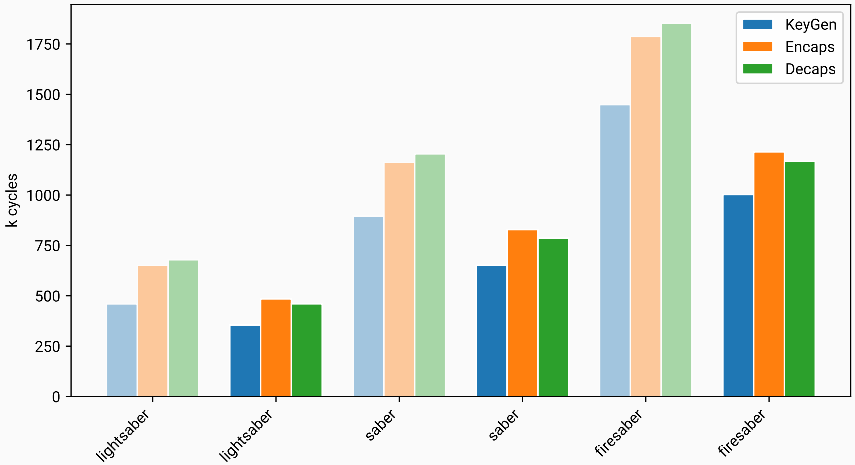




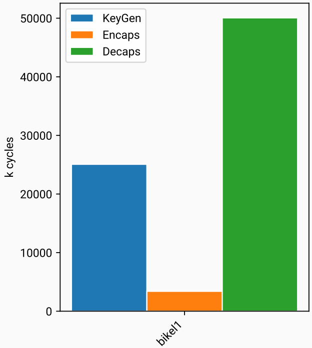
- Nov 2020: New implementation using NTTs by Chung, Hwang, Kannwischer, Seiler, Shih, Yang ([ia.cr/2020/1397](ia.cr/2020/1397))

NTRU (ia.cr/2020/1397)

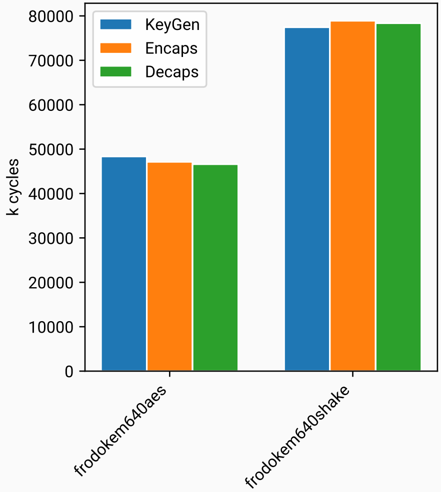




- Nov 2020: New implementation using NTTs by Chung, Hwang, Kannwischer, Seiler, Shih, Yang (ia.cr/2020/1397)

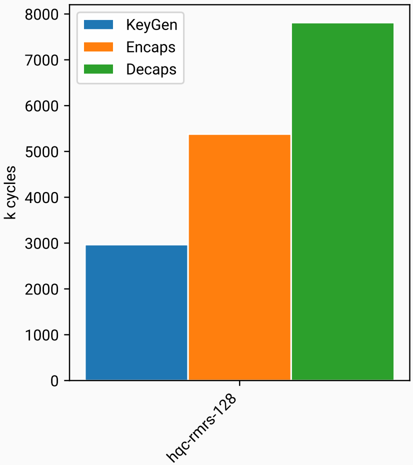


- New implementation by Chen and Chou (ia.cr/2021/493)

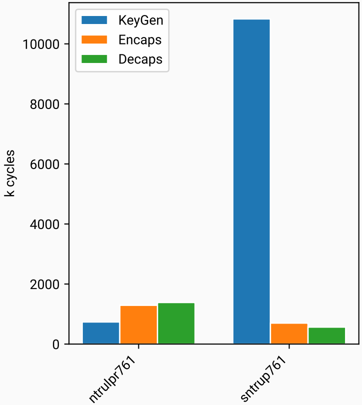


- No parameter and implementation changes

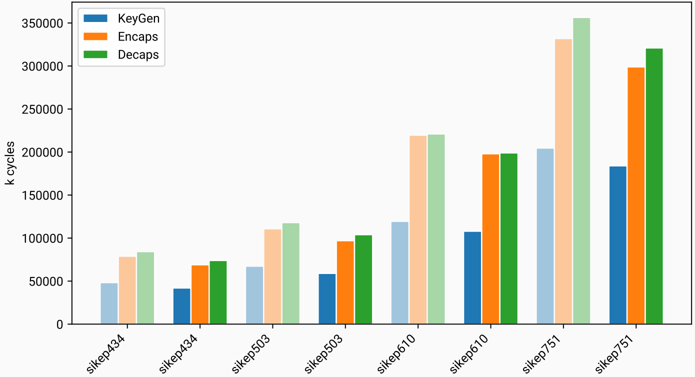
HQC



- Reference implementation from PQClean now working on the M4



- Jan 2019: Optimized implementation by Huang, Chen, and Yang (ia.cr/2019/100)
- Oct 2020: Faster implementation using NTTs by Alkim, Cheng, Chung, Evkan, Huang, Hwang, Li, Niederhagen, Shih, Wälde, and Yang (ia.cr/2020/1216)

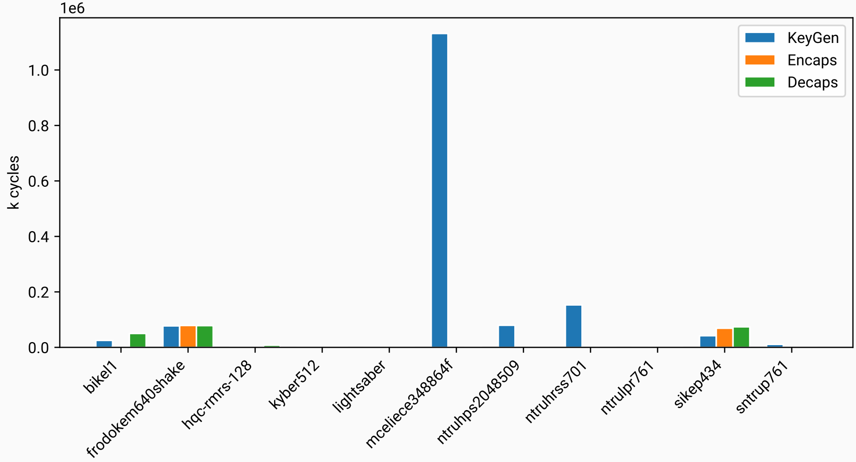


- Apr 2020: M4 implementation by Seo, Anastasova, Jalali, and Azarderakhsh (ia.cr/2020/410)
- Jan 2021: Faster implementation by Anastasova, Azarderakhsh, and Mozaffari Kermani (ia.cr/2021/115)
- Only former has code available and integrated in pqm4

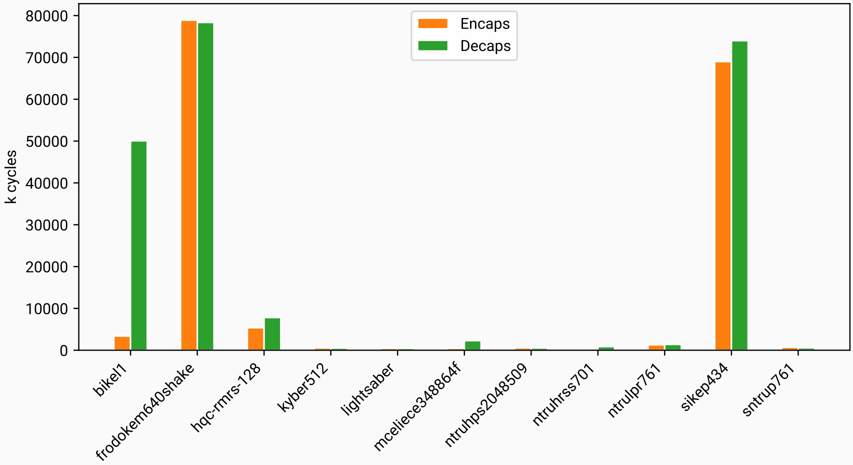
Summary: KEMs in Round 3

- Let's compare some of the results
  - NIST security level 1
    - For NTRUPrime: M4 implementations are only available on level 3
  - Only CCA variants for KEMs
  - SHAKE/SHA-3 parameter sets in case there are multiple ones
- For the full, up-to-date, and unbiased results go to `https://github.com/mupq/pqm4`

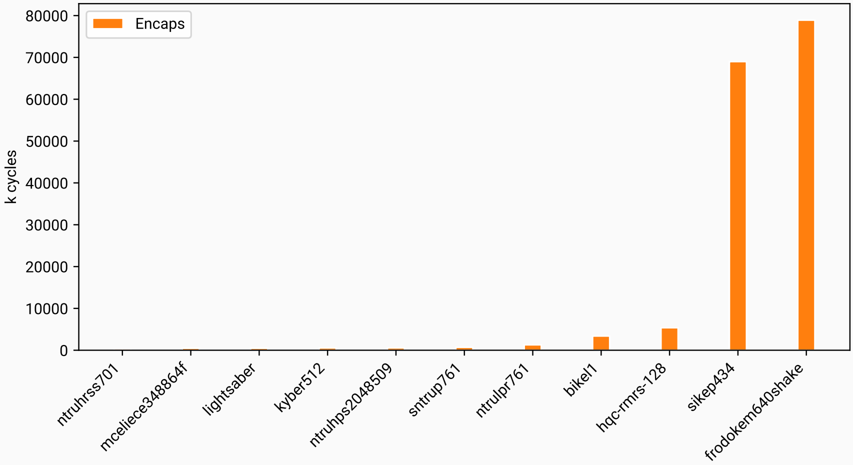
Summary: KEMs in Round 3



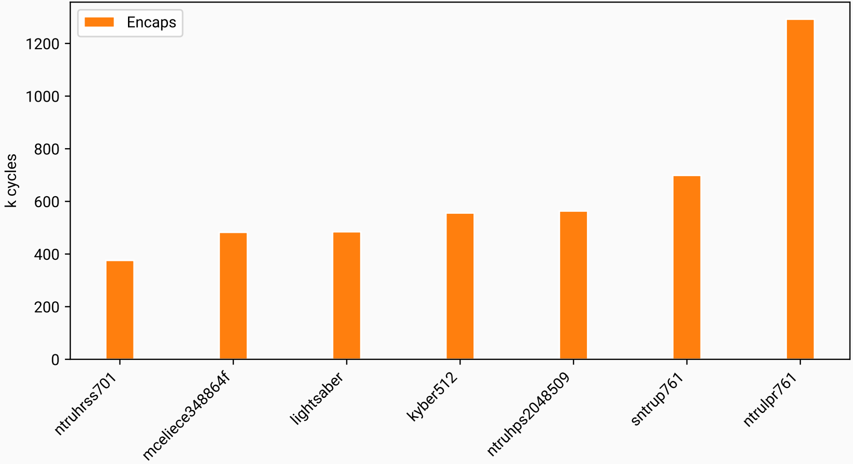
Summary: KEMs in Round 3



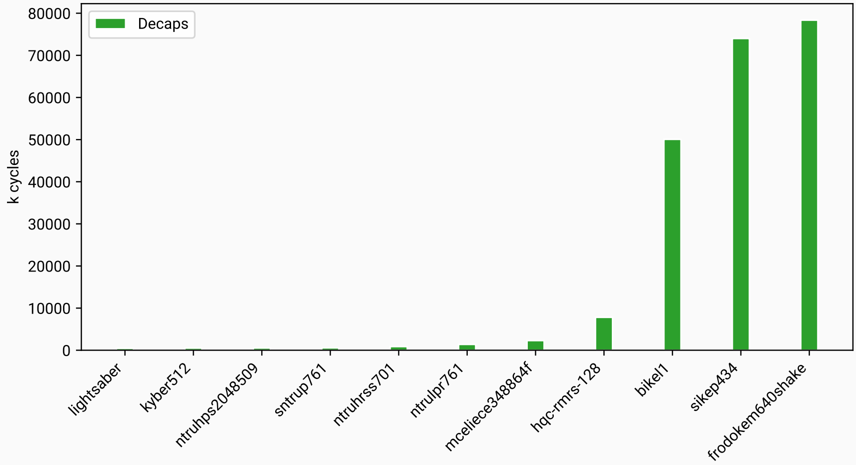
Summary: KEMs in Round 3 (Encapsulation)



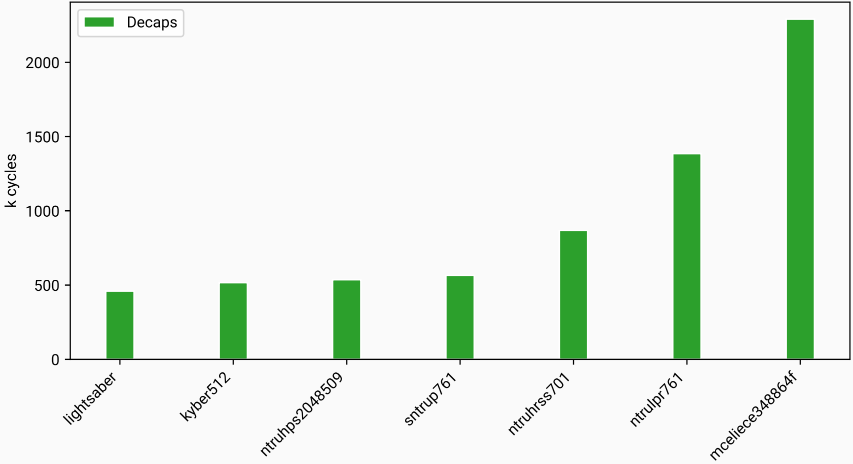
Summary: KEMs in Round 3 (Encapsulation)



Summary: KEMs in Round 3 (Decapsulation)

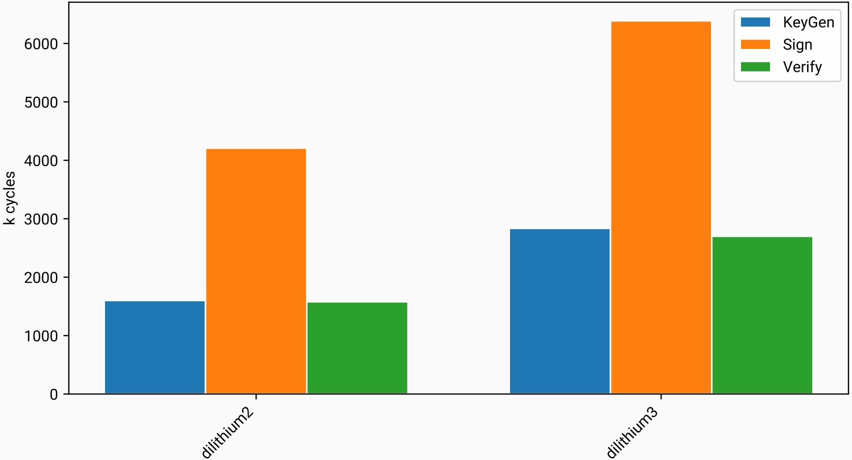


Summary: KEMs in Round 3 (Decapsulation)



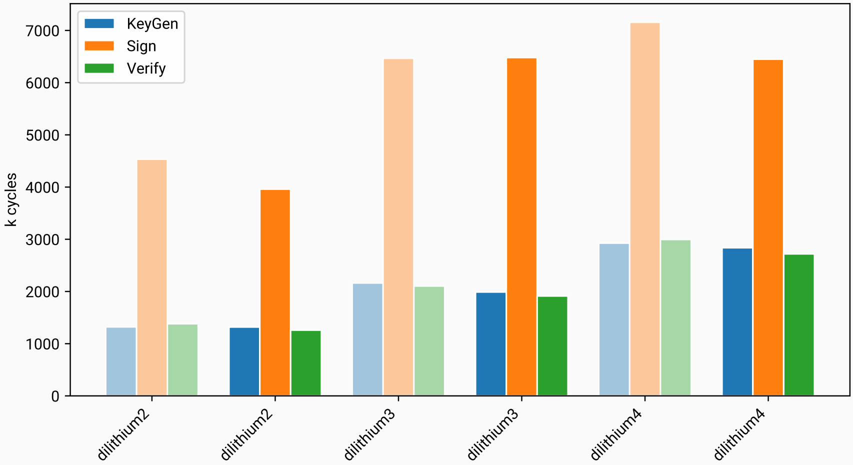
Implementation updates: Signatures

	pqm4	M4 implementation	new since Aug 2019	implementation
DILITHIUM	✓	✓	✓	ia.cr/2020/1278
Falcon	✓	✓	✗	ia.cr/2019/893
Rainbow	✗	✓	✓	ia.cr/2021/532
GeMSS	✗	✗	✗	–
Picnic	soon	✓	✓	unpublished
SPHINCS+	✓	✗	✗	ref

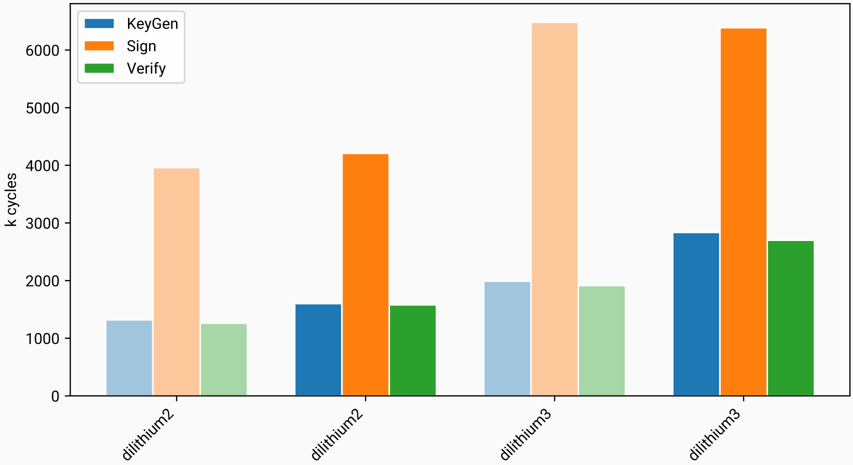


- New implementation by Greconici, Kannwischer, and Sprenkels (ia.cr/2020/1278)
- Significant parameter changes in Round 3

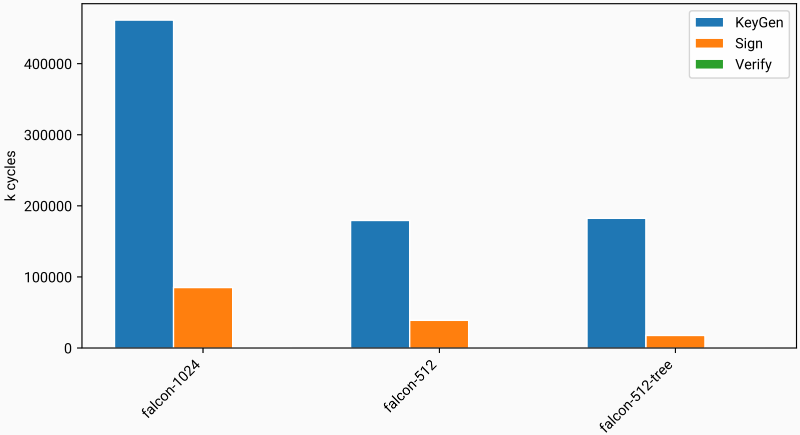
DILITHIUM (ia.cr/2020/1278)



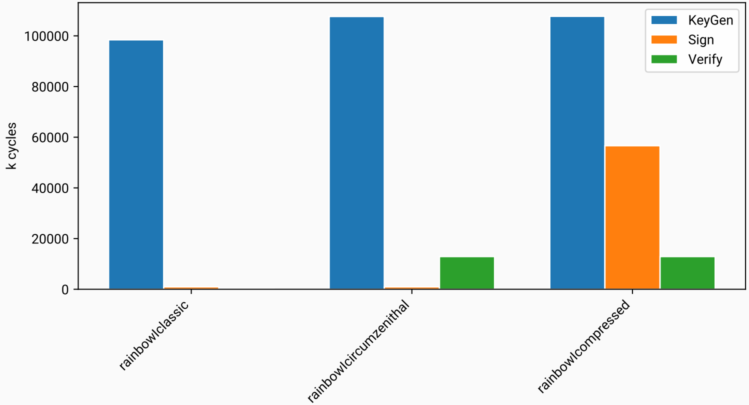
DILITHIUM (Round 2 vs. Round 3 parameters)



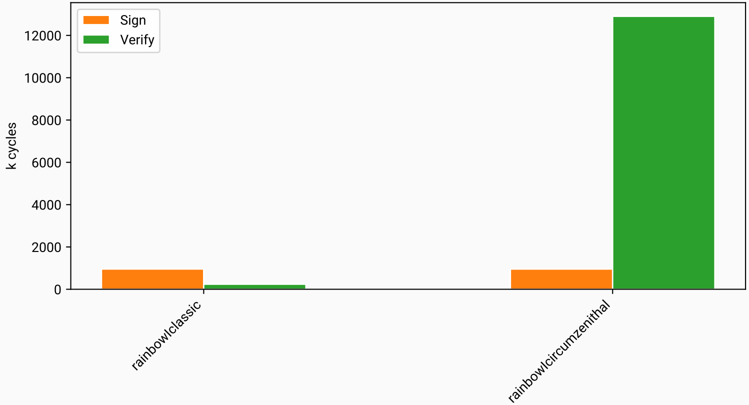
Falcon



- No changes in Falcon
- Fastest implementation is still by Pornin (ia.cr/2019/893)

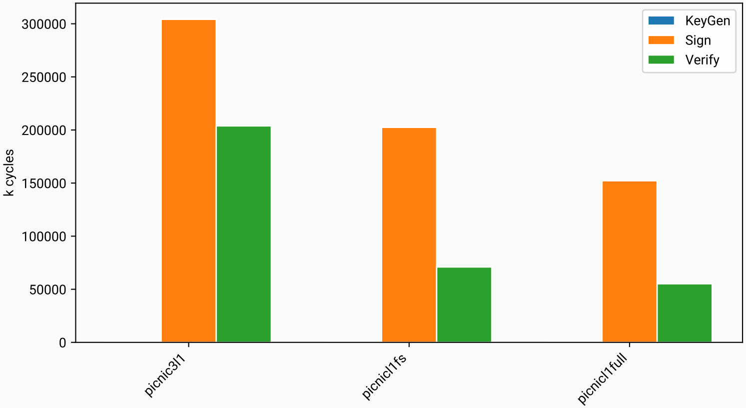


- Keys too large for our target platform
- New implementation by Chou, Kannwischer, and Yang targeting the EFM32GG11B (ia.cr/2021/532)



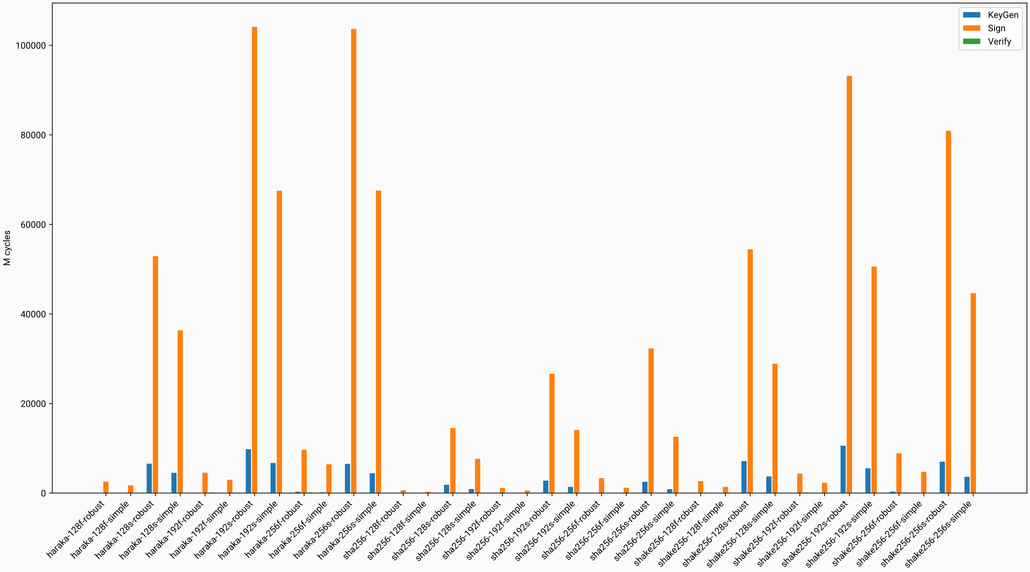
- Keys too large for our target platform
- New implementation by Chou, Kannwischer, and Yang targeting the EFM32GG11B (ia.cr/2021/532)

- Keys too large for our target platform
- No implementations available for other M4 platforms

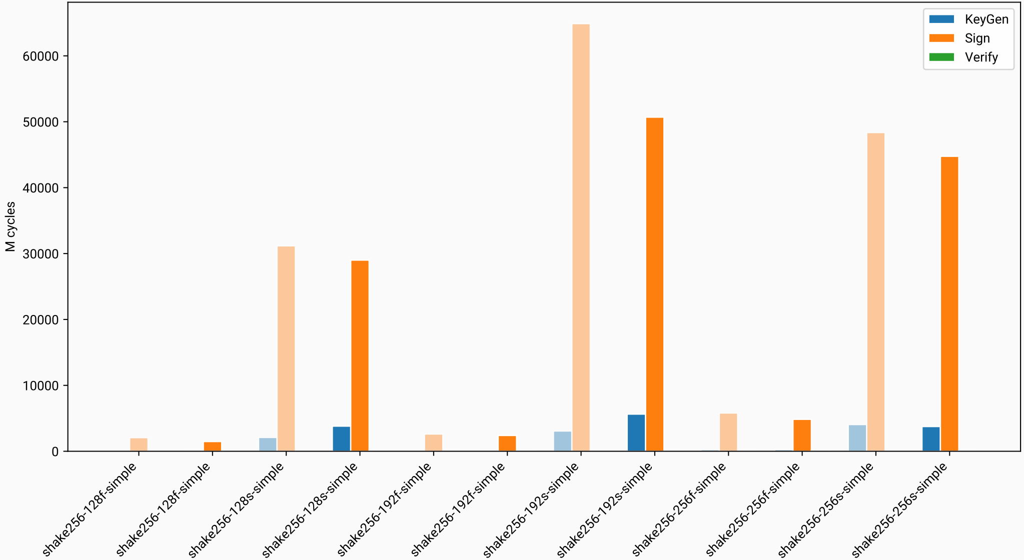


- New M4 implementation by Aranha, Kales, Ramacher, and Zaverucha
- Not published yet, but soon to be merged into pqm4

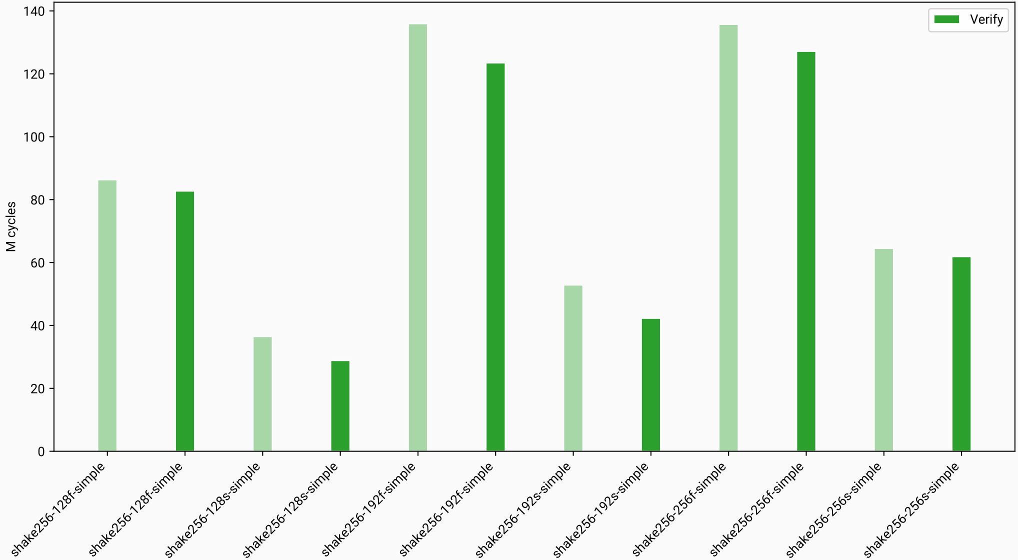
SPHINCS+


KeyGen
Sign
Verify
M cycles
0
20000
40000
60000
80000
100000
haraka-128f-robust
haraka-128f-simple
haraka-128s-robust
haraka-128s-simple
haraka-192f-robust
haraka-192f-simple
haraka-192s-robust
haraka-192s-simple
haraka-256f-robust
haraka-256f-simple
haraka-256s-robust
haraka-256s-simple
sha256-128f-robust
sha256-128f-simple
sha256-128s-robust
sha256-128s-simple
sha256-192f-robust
sha256-192f-simple
sha256-192s-robust
sha256-192s-simple
sha256-256f-robust
sha256-256f-simple
sha256-256s-robust
sha256-256s-simple
shake256-128f-robust
shake256-128f-simple
shake256-128s-robust
shake256-128s-simple
shake256-192f-robust
shake256-192f-simple
shake256-192s-robust
shake256-192s-simple
shake256-256f-robust
shake256-256f-simple
shake256-256s-robust
shake256-256s-simple

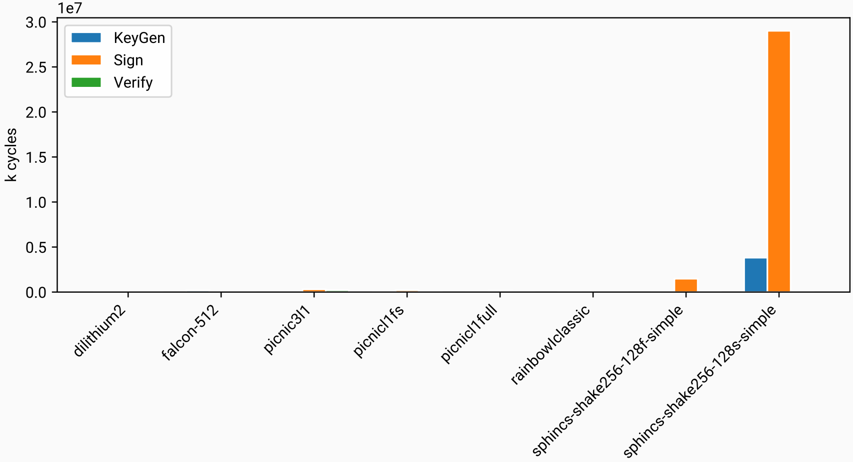
SPHINCS+ (Round 2 vs. Round 3)



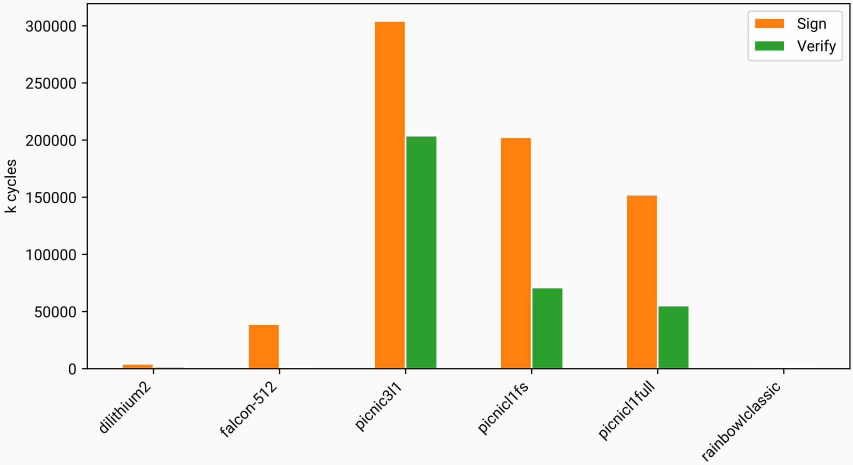
SPHINCS+ Verification (Round 2 vs. Round 3)



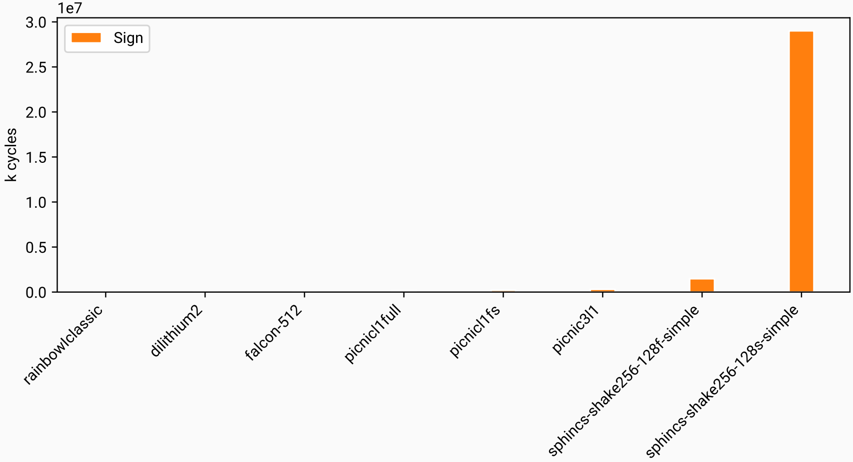
Summary: Signatures in Round 3



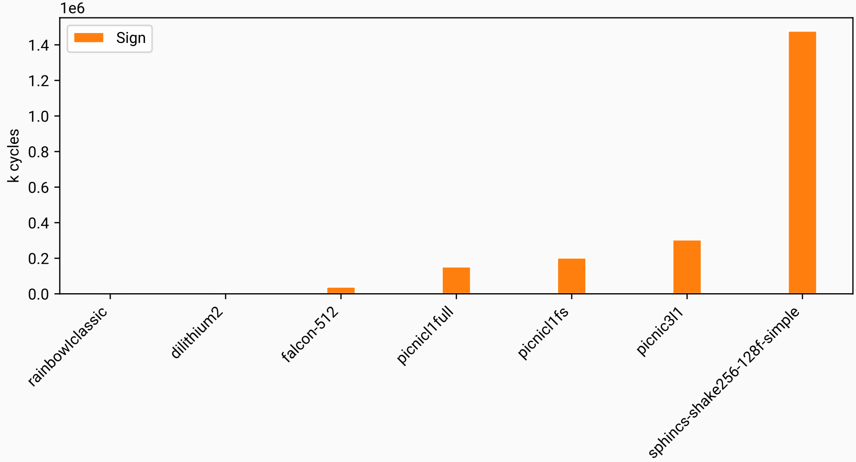
Summary: Signatures in Round 3



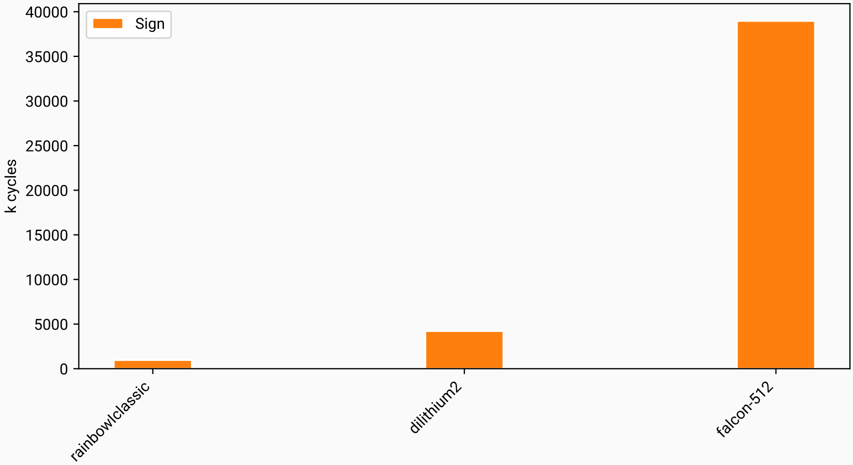
Summary: Signatures in Round 3 (Sign)



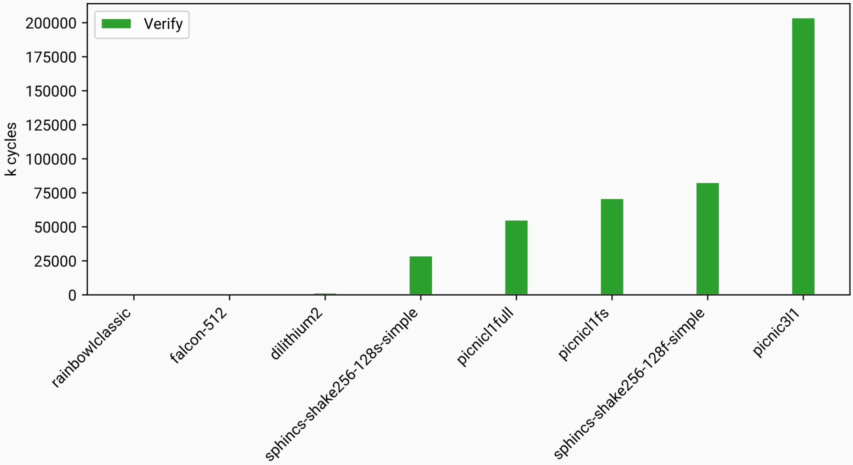
Summary: Signatures in Round 3 (Sign)



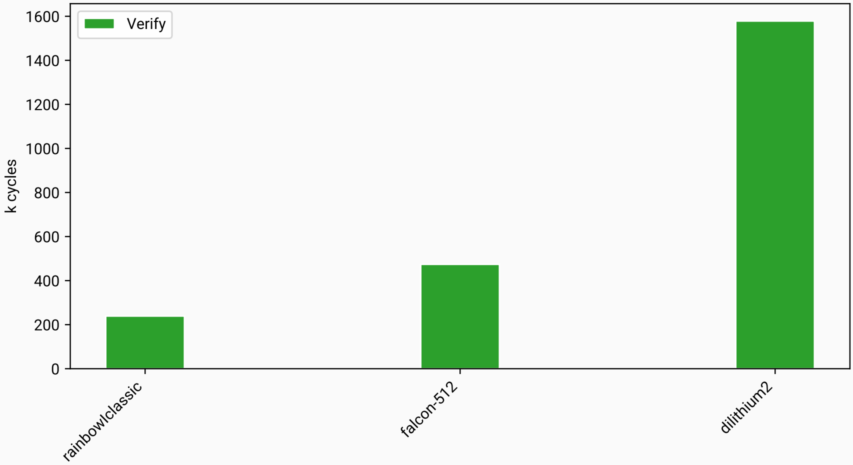
Summary: Signatures in Round 3 (Sign)



Summary: Signatures in Round 3 (Verify)



Summary: Signatures in Round 3 (Verify)



Conclusion

- Significant progress in embedded PQC implementations in the last 2 years
- Still optimization work left; some candidates have received very little attention
- Memory consumption often considered now; code size usually ignored

- Fastest KEM (Encaps): **NTRU-HRSS-701**
- Fastest KEM (Decaps): **Lightsaber**
- Fastest KEM (Encaps+Decaps): **Lightsaber**
- Fastest Signature: **Rainbow**; followed by Dilithium (signing) or Falcon (verification)

What's next?

- Implementations for the M4 can only get faster (and smaller)
- Cortex-M3
  - More limited instruction set
  - Long multiplications have data-dependent cycle count
    $\rightarrow$ compiling reference code often leads to vulnerable implementations
  - `https://github.com/mupq/pqm3`
- Countermeasures against side-channel attacks and fault attacks
  - Only very few PQC schemes have (good) masked implementations
  - Ideally: Collection of masked implementations; unified framework for performing leakage evaluation and t-tests (new build system supports ChipWhisperer!)
- Vectorized ARM implementations
- RISC-V

Thank you very much for your attention

`matthias@kannwischer.eu`

`https://github.com/mupq/pqm4`