Kyber

Roberto Avanzi, Joppe Bos, Jintai Ding, Léo Ducas, Eike Kiltz, Tancrède Lepoint,
Vadim Lyubashevsky, John M. Schanck, **Peter Schwabe**, Gregor Seiler, Damien Stehlé

authors@pq-crystals.org

`https://pq-crystals.org/kyber`

March 26, 2023

Kyber summary

- MLWE-based IND-CCA2-secure KEM
    - IND-CPA secure LPR public-key encryption
    - Tweaked FO transform
- Only KEM selected by NIST for standardization after round 3

Kyber summary

- MLWE-based IND-CCA2-secure KEM
  - IND-CPA secure LPR public-key encryption
  - Tweaked FO transform
- Only KEM selected by NIST for standardization after round 3
- Very fast across different platforms
- Will be even faster with HW Keccak acceleration

Kyber summary

- MLWE-based IND-CCA2-secure KEM
  - IND-CPA secure LPR public-key encryption
  - Tweaked FO transform
- Only KEM selected by NIST for standardization after round 3
- Very fast across different platforms
- Will be even faster with HW Keccak acceleration
- Same optimized routines across all parameter sets
- Designed for efficient constant-time implementation
- Designed for efficient vectorization
- Designed for low memory consumption on embedded platforms

Decisions I: symmetric crypto (closed)

NIST decisions

- No change in domain separation
- No TurboShake for matrix generation
- Keccak-based only (no “90s version”)

Decisions II: FO transform (still open?)

Hashing prefix(pk)

- H(pk) into coins and shared key
- Cheaper and sufficient: Use prefix(pk) instead
- Little community feedback so far
- Probably stick to H(pk)?

Decisions II: FO transform (still open?)

Hashing prefix(pk)

- H(pk) into coins and shared key
- Cheaper and sufficient: Use prefix(pk) instead
- Little community feedback so far
- Probably stick to H(pk)?

Ciphertext hash

- Kyber hashes H(c) into shared key, also “double-hashing” of message

Decisions II: FO transform (still open?)

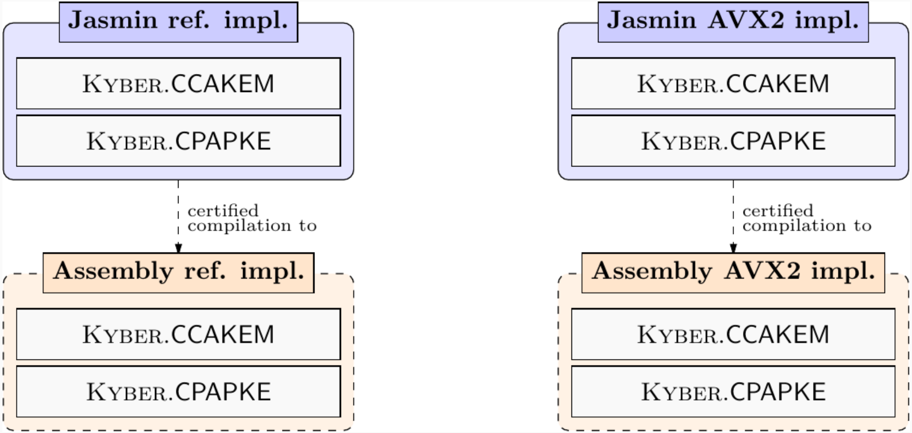
Hashing prefix(pk)

- H(pk) into coins and shared key
- Cheaper and sufficient: Use prefix(pk) instead
- Little community feedback so far
- Probably stick to H(pk)?

Ciphertext hash

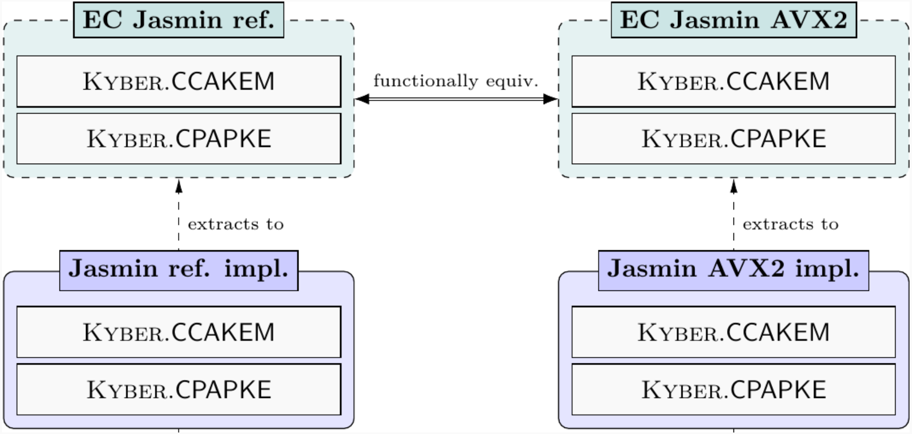
- Kyber hashes H(c) into shared key, also “double-hashing” of message
- Complicates QROM proofs
- Reductions less tight (additional collision bounds)
- Also: dropping this hash would speed up Encaps
- **Worth more discussion on pqc-forum!**

High-assurance implementation



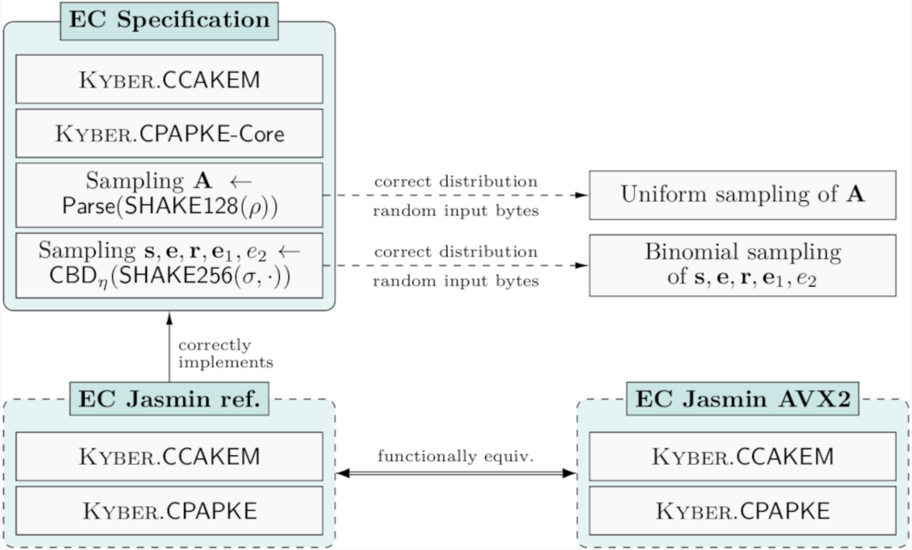
Joint work with José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Benjamin Grégoire, Vincent Laporte, Jean-Christophe Léchenet, Tiago Oliveira, Hugo Pacheco, Miguel Quaresma, Antoine Séré, and Pierre-Yves Strub.

High-assurance implementation



Joint work with José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Benjamin Grégoire, Vincent Laporte, Jean-Christophe Léchenet, Tiago Oliveira, Hugo Pacheco, Miguel Quaresma, Antoine Séré, and Pierre-Yves Strub.

High-assurance implementation


EC Specification
Kyber.CCAKEM
Kyber.CPAPKE-Core
Sampling $\mathbf{A} \leftarrow$ Parse(SHAKE128(ρ))
Sampling $\mathbf{s}, \mathbf{e}, \mathbf{r}, \mathbf{e}_1, e_2 \leftarrow$ CBD$_\eta$(SHAKE256($\sigma, \cdot$))
correct distribution
random input bytes
Uniform sampling of $\mathbf{A}$
correct distribution
random input bytes
Binomial sampling of $\mathbf{s}, \mathbf{e}, \mathbf{r}, \mathbf{e}_1, e_2$
correctly implements
EC Jasmin ref.
Kyber.CCAKEM
Kyber.CPAPKE
functionally equiv.
EC Jasmin AVX2
Kyber.CCAKEM
Kyber.CPAPKE

Performance

Implementation	operation	Skylake	Haswell	Comet Lake
C/asm AVX2	keygen	49572	47280	41682
	encaps	60018	62900	55956
	decaps	45854	47784	43906
Jasmin AVX2 (fully verified)	keygen	106578	96296	93244
	encaps	119308	111536	107474
	decaps	105336	98328	96564
Jasmin AVX2 (fully optimized)	keygen	50004	48800	45046
	encaps	65132	63988	59496
	decaps	50340	51444	48172

Spectre v1 protection

Joint work with Basavesh Ammanaghatta Shivakumar, Gilles Barthe, Benjamin Grégoire, Vincent Laporte, Tiago Oliveira, Swarn Priya, Peter Schwabe, and Lucas Tabary-Maujean.

- Security type system in jasmin
- Enforce no branching on secrets, no memory access at secret position
- Also enforce this **in speculative execution after misspeculated conditional branch**

Spectre v1 protection

Joint work with Basavesh Ammanaghatta Shivakumar, Gilles Barthe, Benjamin Grégoire, Vincent Laporte, Tiago Oliveira, Swarn Priya, Peter Schwabe, and Lucas Tabary-Maujean.

- Security type system in jasmin
- Enforce no branching on secrets, no memory access at secret position
- Also enforce this **in speculative execution after misspeculated conditional branch**
- Guide programmer to protect code
- Selective speculative load hardening (selSLH):
  - Misspeculation flag in register
  - Mask "transient" values with flag before leaking them

Spectre v1 protection

Joint work with Basavesh Ammanaghatta Shivakumar, Gilles Barthe, Benjamin Grégoire, Vincent Laporte, Tiago Oliveira, Swarn Priya, Peter Schwabe, and Lucas Tabary-Maujean.

- Security type system in jasmin
- Enforce no branching on secrets, no memory access at secret position
- Also enforce this **in speculative execution after misspeculated conditional branch**
- Guide programmer to protect code
- Selective speculative load hardening (selSLH):
  - Misspeculation flag in register
  - Mask “transient” values with flag before leaking them
- Overhead for Kyber768 (on Intel Comet Lake):
  - 0.28% for Keypair
  - 0.55% for Encaps
  - 0.75% for Decaps

More online

https://pq-crystals.org/kyber

- **High-assurance Kyber:** https://eprint.iacr.org/2023/215
- **Spectre v1 protection:** https://eprint.iacr.org/2022/1270
- **Libjade:** https://github.com/formosa-crypto/libjade