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 $\mathbf{A} \leftarrow \text{Parse}(\text{SHAKE128}(\rho))$
- Sampling $\mathbf{s}, \mathbf{e}, \mathbf{r}, \mathbf{e_1}, \mathbf{e_2} \leftarrow \text{CBD}_\eta(\text{SHAKE256}(\sigma, \cdot))$

Correct distribution
Random input bytes

Uniform sampling of $\mathbf{A}$

Binomial sampling of $\mathbf{s}, \mathbf{e}, \mathbf{r}, \mathbf{e_1}, \mathbf{e_2}$

EC Jasmin ref.

**KYBER.CCAKEM**

**KYBER.CPAPKE**

Correctly implements

EC Jasmin AVX2

**KYBER.CCAKEM**

**KYBER.CPAPKE**

Functionally equiv.
## Performance

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

- Spectre v1 protection: https://eprint.iacr.org/2022/1270
- Libjade: https://github.com/formosa-crypto/libjade