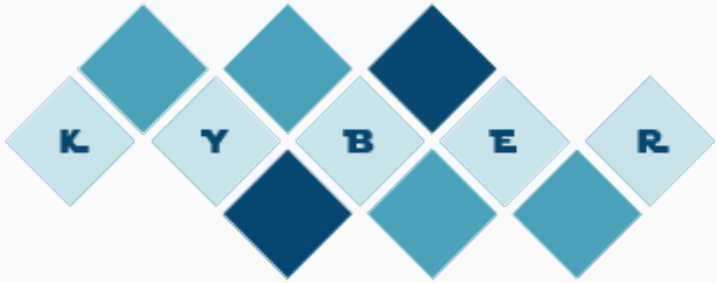

CRYSTALS–Kyber

Roberto Avanzi, Joppe Bos, 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`

April 12, 2018

The big picture

Kyber.CPAPKE: LPR encryption or "Noisy ElGamal"

$\mathbf{s}, \mathbf{e} \leftarrow \chi$

$sk = \mathbf{s}, pk = \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$

$\mathbf{r}, \mathbf{e}_1, e_2 \leftarrow \chi$

$\mathbf{u} \leftarrow \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$

$v \leftarrow \mathbf{t}^T\mathbf{r} + e_2 + \mathrm{Enc}(m)$

$c = (\mathbf{u}, v)$

$m = Dec(v - \mathbf{s}^T\mathbf{u})$

The big picture

Kyber.CPAPKE: LPR encryption or "Noisy ElGamal"

$\mathbf{s}, \mathbf{e} \leftarrow \chi$

$sk = \mathbf{s}, pk = \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$

$\mathbf{r}, \mathbf{e}_1, e_2 \leftarrow \chi$

$\mathbf{u} \leftarrow \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$

$v \leftarrow \mathbf{t}^T\mathbf{r} + e_2 + \mathrm{Enc}(m)$

$c = (\mathbf{u}, v)$

$m = Dec(v - \mathbf{s}^T\mathbf{u})$

Kyber.CCAKEM: CCA-secure KEM via tweaked FO transform

- Enforce "honest" encapsulation
- Generate all randomnes in encryption via PRG, encrypt seed
- Recover seed during decapsulation
- Reencrypt and compare ciphertexts

Kyber.CPAPKE – design decisions I

- Use MLWE instead of LWE or RLWE
  - Performance similar to RLWE
  - Very easy to scale security and performance
  - Remove some of the cyclic structure of RLWE
- Use $\mathcal{R} = \mathbb{Z}_q[X]/(X^{256} + 1)$ with $q = 7681$
  - Fast, simple, in-place negacyclic NTT for multiplication
  - Most widely studied and best understood structure

Kyber.CPAPKE – design decisions I

- Use MLWE instead of LWE or RLWE
  - Performance similar to RLWE
  - Very easy to scale security and performance
  - Remove some of the cyclic structure of RLWE
- Use $\mathcal{R} = \mathbb{Z}_q[X]/(X^{256} + 1)$ with $q = 7681$
  - Fast, simple, in-place negacyclic NTT for multiplication
  - Most widely studied and best understood structure
- Use centered binomial noise
  - Efficient to sample without timing leakage

Kyber.CPAPKE – design decisions I

- Use MLWE instead of LWE or RLWE
  - Performance similar to RLWE
  - Very easy to scale security and performance
  - Remove some of the cyclic structure of RLWE
- Use $\mathcal{R} = \mathbb{Z}_q[X]/(X^{256} + 1)$ with $q = 7681$
  - Fast, simple, in-place negacyclic NTT for multiplication
  - Most widely studied and best understood structure
- Use centered binomial noise
  - Efficient to sample without timing leakage
- Generate $\mathbf{A}$ via XOF(ρ) ("NewHope style")
  - Avoid "nothing-up-my-sleeves" discussions
  - Avoid all-for-the-price-of-one attacks
  - Sample $\mathbf{A}$ in NTT domain: save k^2 NTTs

Kyber.CPAPKE – design decisions II

- Compress ciphertexts (round off least-significant bits)
  - Reduce bandwidth requirements
  - Introduce extra “LWR” noise

Kyber.CPAPKE – design decisions II

- Compress ciphertexts (round off least-significant bits)
  - Reduce bandwidth requirements
  - Introduce extra "LWR" noise
- Compress public keys
  - Reduce bandwidth requirements
  - Adds MLWR-style assumption instead of pure reduction from MLWE (thanks to D'Anvers for pointing this out)
  - No actual attacks or security problems
  - Could fix proof by re-randomizing after decompression

Kyber.CPAPKE – design decisions II

- Compress ciphertexts (round off least-significant bits)
  - Reduce bandwidth requirements
  - Introduce extra "LWR" noise
- Compress public keys
  - Reduce bandwidth requirements
  - Adds MLWR-style assumption instead of pure reduction from MLWE (thanks to D'Anvers for pointing this out)
  - No actual attacks or security problems
  - Could fix proof by re-randomizing after decompression
- Allow decapsulation failures
  - Failure probability $< 2^{-140}$
  - Avoiding failures would cost security (or performance)

FO transform – tweaks

- Hash public key into seed and shared key
  - Multitarget protection against precomputation attacks
  - Obtain contributory KEM

FO transform – tweaks

- Hash public key into seed and shared key
  - Multitarget protection against precomputation attacks
  - Obtain contributory KEM
- Hash ciphertext into shared key
  - Shared key depends on full KEM transcript
  - More robust when building, e.g., AKE from Kyber

FO transform – tweaks

- Hash public key into seed and shared key
  - Multitarget protection against precomputation attacks
  - Obtain contributory KEM
- Hash ciphertext into shared key
  - Shared key depends on full KEM transcript
  - More robust when building, e.g., AKE from Kyber
- Use Keccak-based functions for all hashes and XOF

Parameter sets and performance

Kyber512 ($k = 2$, level 1)			
Sizes (in Bytes)		**Haswell Cycles (AVX2)**	
sk:	1632	**gen:**	55 160
pk:	736	**enc:**	75 680
ct:	800	**dec:**	74 428

Kyber768 ($k = 3$, level 3)			
Sizes (in Bytes)		**Haswell Cycles (AVX2)**	
sk:	2400	**gen:**	85 472
pk:	1088	**enc:**	112 660
ct:	1152	**dec:**	108 904

Kyber1024 ($k = 4$, level 5)			
Sizes (in Bytes)		**Haswell Cycles (AVX2)**	
sk:	3168	**gen:**	121 056
pk:	1440	**enc:**	157 964
ct:	1504	**dec:**	154 952

Kyber online

https://pq-crystals.org/kyber