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

August 23, 2019

Reminder: the big picture

Kyber.CPAPKE: LPR encryption or "Noisy ElGamal"

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

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

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

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

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

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

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

Reminder: 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

- Use implicit rejection
- Hash public key into seed and shared key
- Hash ciphertext into shared key
- Use Keccak-based functions for all hashes and XOF

Reminder: Kyber in Round 1

- Use MLWE instead of LWE or RLWE

Reminder: Kyber in Round 1

- Use MLWE instead of LWE or RLWE
- Use $\mathcal{R} = \mathbb{Z}_q[X]/(X^{256} + 1)$ with $q = 7681$

Reminder: Kyber in Round 1

- Use MLWE instead of LWE or RLWE
- Use $\mathcal{R} = \mathbb{Z}_q[X]/(X^{256} + 1)$ with $q = 7681$
- Use centered binomial noise

Reminder: Kyber in Round 1

- Use MLWE instead of LWE or RLWE
- Use $\mathcal{R} = \mathbb{Z}_q[X]/(X^{256} + 1)$ with $q = 7681$
- Use centered binomial noise
- Generate **A** via XOF(ρ) ("NewHope style")

Reminder: Kyber in Round 1

- Use MLWE instead of LWE or RLWE
- Use $\mathcal{R} = \mathbb{Z}_q[X]/(X^{256} + 1)$ with $q = 7681$
- Use centered binomial noise
- Generate **A** via XOF(ρ) ("NewHope style")
- Compress ciphertexts (round off least-significant bits)

Reminder: Kyber in Round 1

- Use MLWE instead of LWE or RLWE
- Use $\mathcal{R} = \mathbb{Z}_q[X]/(X^{256} + 1)$ with $q = 7681$
- Use centered binomial noise
- Generate $\mathbf{A}$ via XOF(ρ) ("NewHope style")
- Compress ciphertexts (round off least-significant bits)
- Compress public keys

NIST comments

"We note that a potential issue is that the security proof does not directly apply to Kyber itself, but rather to a modified version of the scheme which does not compress the public key."

—NIST IR 8240

Main changes in round 2

1. Remove the public-key compression
   - Proof now applies to Kyber itself
   - However, bandwidth requirement increases

Main changes in round 2

1. Remove the public-key compression
    - Proof now applies to Kyber itself
    - However, bandwidth requirement increases
2. Reduce parameter q to 3329
    - Bandwidth requirement decreases
3. Update ciphertext-compression parameters

Main changes in round 2

Kyber sizes, round 1 vs. round 2

Kyber512 ($k = 2$, level 1)			
round 1, sizes in bytes		**round 2, sizes in bytes**	
pk:	736	**pk:**	800
ct:	800	**ct:**	736
Kyber768 ($k = 3$, level 3)			
round 1, sizes in bytes		**round 2, sizes in bytes**	
pk:	1088	**pk:**	1184
ct:	1152	**ct:**	1088
Kyber1024 ($k = 4$, level 5)			
round 1, sizes in bytes		**round 2, sizes in bytes**	
pk:	1440	**pk:**	1568
ct:	1504	**ct:**	1568

Main changes in round 2

1. Remove the public-key compression
    - Proof now applies to Kyber itself
    - However, bandwidth requirement increases
2. Reduce parameter q to 3329
    - Bandwidth requirement decreases
3. Update ciphertext-compression parameters
4. Update the specification of the NTT (inspired by NTTRU)
    - Even faster polynomial multiplication

Main changes in round 2

1. Remove the public-key compression
    - Proof now applies to Kyber itself
    - However, bandwidth requirement increases
2. Reduce parameter q to 3329
    - Bandwidth requirement decreases
3. Update ciphertext-compression parameters
4. Update the specification of the NTT (inspired by NTTRU)
    - Even faster polynomial multiplication
5. Reduce noise parameter to $\eta = 2$
    - Faster noise sampling

Main changes in round 2

1. Remove the public-key compression
    - Proof now applies to Kyber itself
    - However, bandwidth requirement increases
2. Reduce parameter q to 3329
    - Bandwidth requirement decreases
3. Update ciphertext-compression parameters
4. Update the specification of the NTT (inspired by NTTRU)
    - Even faster polynomial multiplication
5. Reduce noise parameter to $\eta = 2$
    - Faster noise sampling
6. Represent public key in NTT domain
    - Save several NTT computations

Kyber is fast

Kyber512 ($k = 2$, level 1)			
Sizes (in Bytes)		**Haswell Cycles (AVX2)**	
sk:	1632	**gen:**	29100
pk:	800	**enc:**	46196
ct:	736	**dec:**	39410

Kyber768 ($k = 3$, level 3)			
Sizes (in Bytes)		**Haswell Cycles (AVX2)**	
sk:	2400	**gen:**	57340
pk:	1184	**enc:**	78692
ct:	1088	**dec:**	68620

Kyber1024 ($k = 4$, level 5)			
Sizes (in Bytes)		**Haswell Cycles (AVX2)**	
sk:	3168	**gen:**	81244
pk:	1568	**enc:**	109584
ct:	1568	**dec:**	97280

Kyber is fast and small

Kyber512 ($k = 2$, level 1)			
Stack usage (in Bytes)		**Cortex-M4 Cycles**	
gen:	2952	**gen:**	513992
enc:	2552	**enc:**	652470
dec:	2560	**dec:**	620946

Kyber768 ($k = 3$, level 3)			
Stack usage (in Bytes)		**Cortex-M4 Cycles**	
gen:	3848	**gen:**	976205
enc:	3128	**enc:**	1146021
dec:	3072	**dec:**	1094314

Kyber1024 ($k = 4$, level 5)			
Stack usage (in Bytes)		**Cortex-M4 Cycles**	
gen:	4360	**gen:**	1574351
enc:	3584	**enc:**	1779192
dec:	3592	**dec:**	1708692

What are we benchmarking, really?

- More than 50% of the cycles are spent in Keccak
  - Many conservative choices in FO transform
  - Use SHAKE-128 to as XOF
  - Generally, Keccak is not very fast in software

What are we benchmarking, really?

- More than 50% of the cycles are spent in Keccak
  - Many conservative choices in FO transform
  - Use SHAKE-128 to as XOF
  - Generally, Keccak is not very fast in software
- Long-term solution: hardware-accelerated Keccak

What are we benchmarking, really?

- More than 50% of the cycles are spent in Keccak
    - Many conservative choices in FO transform
    - Use SHAKE-128 to as XOF
    - Generally, Keccak is not very fast in software
- Long-term solution: hardware-accelerated Keccak
- Short-term problem:
    - Benchmarks of lattice-based KEMs are really benchmarks of symmetric crypto
    - Risk to make wrong decision about *lattice* design from "symmetrically tainted" benchmarks

What are we benchmarking, really?

- More than 50% of the cycles are spent in Keccak
  - Many conservative choices in FO transform
  - Use SHAKE-128 to as XOF
  - Generally, Keccak is not very fast in software
- Long-term solution: hardware-accelerated Keccak
- Short-term problem:
  - Benchmarks of lattice-based KEMs are really benchmarks of symmetric crypto
  - Risk to make wrong decision about *lattice* design from “symmetrically tainted” benchmarks
- Maybe just a small problem, because lattice-based KEMs are all fast enough

What are we benchmarking, really?

- More than 50% of the cycles are spent in Keccak
  - Many conservative choices in FO transform
  - Use SHAKE-128 to as XOF
  - Generally, Keccak is not very fast in software
- Long-term solution: hardware-accelerated Keccak
- Short-term problem:
  - Benchmarks of lattice-based KEMs are really benchmarks of symmetric crypto
  - Risk to make wrong decision about *lattice* design from "symmetrically tainted" benchmarks
- Maybe just a small problem, because lattice-based KEMs are all fast enough
- Better to decide based on
  - size/bandwidth
  - RAM/ROM footprint and gate count in HW
  - simplicity
  - how conservative designs are
  - cost of SCA protection

Kyber-90s

https://www.bbc.co.uk/bbcthree/article/91603cc1-f159-4c89-9462-443a078945ca

90s crypto (AES, SHA-2) is accelerated in HW!

Kyber-90s performance (Haswell cycles)

Kyber512 ($k = 2$, level 1)			
Kyber cycles		**Kyber-90s cycles**	
gen:	29100	**gen:**	15792
enc:	46196	**enc:**	26612
dec:	39410	**dec:**	22248

Kyber768 ($k = 3$, level 3)			
Kyber cycles		**Kyber-90s cycles**	
gen:	57340	**gen:**	25632
enc:	78692	**enc:**	39976
dec:	68620	**dec:**	33744

Kyber1024 ($k = 4$, level 5)			
Kyber cycles		**Kyber-90s cycles**	
gen:	81244	**gen:**	38164
enc:	109584	**enc:**	57280
dec:	97280	**dec:**	50360

Kyber online

https://pq-crystals.org/kyber