From NewHope to Kyber

Peter Schwabe
peter@cryptojedi.org
https://cryptojedi.org
January 12, 2017
“In the past, people have said, maybe it’s 50 years away, it’s a dream, maybe it’ll happen sometime. I used to think it was 50. Now I’m thinking like it’s 15 or a little more. It’s within reach. It’s within our lifetime. It’s going to happen.”

—Mark Ketchen (IBM), Feb. 2012, about quantum computers
The end of crypto as we know it

Shor’s algorithm (1994)

- Factor integers in polynomial time
- Compute discrete logarithms in polynomial time
- Complete break of RSA, ElGamal, DSA, Diffie-Hellman
- Complete break of elliptic-curve variants (ECSDA, ECDH, ...)

Forward-secure post-quantum crypto

- Threatening today:
  - Attacker records encrypted messages now
  - Uses quantum computer in 1-2 decades to break encryption

"Perfect forward secrecy" (PFS) does not help

Countermeasure against key compromise

- Not a countermeasure against cryptographic break

Consequence:

Want post-quantum PFS crypto today
The end of crypto as we know it

Shor’s algorithm (1994)

- Factor integers in polynomial time
- Compute discrete logarithms in polynomial time
- Complete break of RSA, ElGamal, DSA, Diffie-Hellman
- Complete break of elliptic-curve variants (ECSDA, ECDH, . . . )

Forward-secure post-quantum crypto

- Threatening today:
  - Attacker records encrypted messages now
  - Uses quantum computer in 1-2 decades to break encryption
The end of crypto as we know it

Shor’s algorithm (1994)

- Factor integers in polynomial time
- Compute discrete logarithms in polynomial time
- Complete break of RSA, ElGamal, DSA, Diffie-Hellman
- Complete break of elliptic-curve variants (ECSDA, ECDH, ...)

Forward-secure post-quantum crypto

- Threatening today:
  - Attacker records encrypted messages now
  - Uses quantum computer in 1-2 decades to break encryption
- “Perfect forward secrecy” (PFS) does not help
  - Countermeasure against key compromise
  - Not a countermeasure against cryptographic break
The end of crypto as we know it

Shor’s algorithm (1994)

• Factor integers in polynomial time
• Compute discrete logarithms in polynomial time
• Complete break of RSA, ElGamal, DSA, Diffie-Hellman
• Complete break of elliptic-curve variants (ECSDA, ECDH, ...)

Forward-secure post-quantum crypto

• Threatening today:
  • Attacker records encrypted messages now
  • Uses quantum computer in 1-2 decades to break encryption
• “Perfect forward secrecy” (PFS) does not help
  • Countermeasure against key compromise
  • Not a countermeasure against cryptographic break
• Consequence: Want post-quantum PFS crypto today
• Let $R_q = \mathbb{Z}_q[X]/(X^n + 1)$
• Let $\chi$ be an error distribution on $R_q$
• Let $s \in R_q$ be secret
• Attacker is given pairs $(a, as + e)$ with
  • $a$ uniformly random from $R_q$
  • $e$ sampled from $\chi$
• Task for the attacker: find $s$
• Let $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$
• Let $\chi$ be an error distribution on $\mathcal{R}_q$
• Let $s \in \mathcal{R}_q$ be secret
• Attacker is given pairs $(a, as + e)$ with
  • $a$ uniformly random from $\mathcal{R}_q$
  • $e$ sampled from $\chi$
• Task for the attacker: find $s$
• Common choice for $\chi$: discrete Gaussian
• Common optimization for protocols: fix $a$
<table>
<thead>
<tr>
<th>Alice (server)</th>
<th>Bob (client)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s, e \leftarrow \chi )</td>
<td>( s', e' \leftarrow \chi )</td>
</tr>
<tr>
<td>( b \leftarrow as + e )</td>
<td>( b \rightarrow u )</td>
</tr>
<tr>
<td></td>
<td>( u \leftarrow as' + e' )</td>
</tr>
</tbody>
</table>

Alice has \( t = us = ass' + e's \)
Bob has \( t' = bs' = ass' + es' \)

- Secret and noise polynomials \( s, s', e, e' \) are small
- \( t \) and \( t' \) are *approximately* the same
POST-QUANTUM KEY EXCHANGE

A NEW HOPE

erdem alkim
léo ducas
thomas pöppelmann
peter schwabe
• Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)

• Use reconciliation to go from approximate agreement to agreement
  • Originally proposed by Ding (2012)
  • Improvements by Peikert (2014)
  • More improvements in NewHope

• Very conservative parameters ($n = 1024$, $q = 12289$)

• Centered binomial noise $\psi_k(HW(a) - HW(b))$ for $k$-bit $a, b$

• Achieve $\approx 256$ bits of post-quantum security according to very conservative analysis

• Higher security, shorter messages, and $>10 \times$ speedup

• Choose a fresh parameter $a$ for every protocol run

• Encode polynomials in NTT domain

• Multiple implementations
• Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)
• Use reconciliation to go from approximate agreement to agreement
  • Originally proposed by Ding (2012)
  • Improvements by Peikert (2014)
  • More improvements in NewHope
• Very conservative parameters \((n = 1024, q = 12289)\)
• Centered binomial noise \(\psi_k (\text{HW}(a) - \text{HW}(b)\text{ for } k\text{-bit } a, b)\)
• Achieve \(\approx 256\) bits of post-quantum security according to very conservative analysis
• Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)

• Use reconciliation to go from approximate agreement to agreement
  • Originally proposed by Ding (2012)
  • Improvements by Peikert (2014)
  • More improvements in NewHope

• Very conservative parameters \((n = 1024, q = 12289)\)

• Centered binomial noise \(\psi_k (\text{HW}(a) - \text{HW}(b) \text{ for } k\text{-bit } a, b)\)

• Achieve \(\approx 256\) bits of post-quantum security according to very conservative analysis

• Higher security, shorter messages, and \(> 10\times\) speedup
• Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)
• Use reconciliation to go from approximate agreement to agreement
  • Originally proposed by Ding (2012)
  • Improvements by Peikert (2014)
  • More improvements in NewHope
• Very conservative parameters \((n = 1024, q = 12289)\)
• Centered binomial noise \(\psi_k (\text{HW}(a) - \text{HW}(b))\) for \(k\)-bit \(a, b\)
• Achieve \(\approx 256\) bits of post-quantum security according to very conservative analysis
• Higher security, shorter messages, and \(> 10\times\) speedup
• Choose a fresh parameter \(a\) for every protocol run
• Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)
• Use reconciliation to go from approximate agreement to agreement
  • Originally proposed by Ding (2012)
  • Improvements by Peikert (2014)
  • More improvements in NewHope
• Very conservative parameters \((n = 1024, q = 12289)\)
• Centered binomial noise \(\psi_k (\text{HW}(a)−\text{HW}(b) \text{ for } k\text{-bit } a, b)\)
• Achieve \(\approx 256\) bits of post-quantum security according to very conservative analysis
• Higher security, shorter messages, and \(> 10\times\) speedup
• Choose a fresh parameter \(a\) for every protocol run
• Encode polynomials in NTT domain
• Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)
• Use reconciliation to go from approximate agreement to agreement
  • Originally proposed by Ding (2012)
  • Improvements by Peikert (2014)
  • More improvements in NewHope
• Very conservative parameters ($n = 1024, q = 12289$)
• Centered binomial noise $\psi_k (\text{HW}(a) - \text{HW}(b)$ for $k$-bit $a, b$)
• Achieve $\approx 256$ bits of post-quantum security according to very conservative analysis
• Higher security, shorter messages, and $> 10 \times$ speedup
• Choose a fresh parameter $a$ for every protocol run
• Encode polynomials in NTT domain
• Multiple implementations
NewHope in the real world

- July 7, 2016, Google announces 2-year post-quantum experiment
- NewHope+X25519 (CECPQ1) in BoringSSL for Chrome Canary
- Used in access to select Google services

November 28, 2016: “At this point the experiment is concluded.”
NewHope in the real world

- July 7, 2016, Google announces 2-year post-quantum experiment
- NewHope+X25519 (CECPQ1) in BoringSSL for Chrome Canary
- Used in access to select Google services
- November 28, 2016: “At this point the experiment is concluded.”
“[…] we did not find any unexpected impediment to deploying something like NewHope. There were no reported problems caused by enabling it.”
“[…] if the need arose, it would be practical to quickly deploy NewHope in TLS 1.2. (TLS 1.3 makes things a little more complex and we did not test with CECPQ1 with it.)”
“Although the median connection latency only increased by a millisecond, the latency for the slowest 5% increased by 20ms and, for the slowest 1%, by 150ms. Since NewHope is computationally inexpensive, we’re assuming that this is caused entirely by the increased message sizes. Since connection latencies compound on the web (because subresource discovery is delayed), the data requirement of NewHope is moderately expensive for people on slower connections.”
Are we done? Is the Internet safe again?

- Security analysis assumes that we have an LWE instance
- Structure of RLWE is ignored
- Somewhat large messages (≈ 2KB each way)
- Maybe overly conservative security...
- “Only” does ephemeral key exchange
- Must not reuse keys/noise
- No CCA security
- Message format depends on multiplication algorithm

Back to the drawing board!
Are we done? Is the Internet safe again?

Disadvantages of NewHope

- Security analysis assumes that we have an LWE instance
- Structure of RLWE is ignored

Back to the drawing board!
Are we done? Is the Internet safe again?

Disadvantages of NewHope

- Security analysis assumes that we have an LWE instance
- Structure of RLWE is ignored
- Somewhat large messages ($\approx 2$KB each way)
- Maybe overly conservative security...?
Are we done? Is the Internet safe again?

Disadvantages of NewHope

- Security analysis assumes that we have an LWE instance
- Structure of RLWE is ignored
- Somewhat large messages ($\approx 2$KB each way)
- Maybe overly conservative security. . . ?
- “Only” does ephemeral key exchange
- Must not reuse keys/noise
- No CCA security
Disadvantages of NewHope

- Security analysis assumes that we have an LWE instance
- Structure of RLWE is ignored
- Somewhat large messages ($\approx 2$KB each way)
- Maybe overly conservative security...?
- “Only” does ephemeral key exchange
- Must not reuse keys/noise
- No CCA security
- Message format depends on multiplication algorithm
Are we done? Is the Internet safe again?

Disadvantages of NewHope

- Security analysis assumes that we have an LWE instance
- Structure of RLWE is ignored
- Somewhat large messages ($\approx 2$KB each way)
- Maybe overly conservative security...?
- “Only” does ephemeral key exchange
- Must not reuse keys/noise
- No CCA security
- Message format depends on multiplication algorithm

Back to the drawing board!
Shi Bai  Joppe Bos  Léo Ducas
Eike Kiltz  Tancrède Lepoint  Vadim Lyubashevsky
John M. Schanck  Peter Schwabe  Damien Stehlé
The design of Kyber (WiP)

- Use **Module-Lattices** and MLWE
  - RLWE: large polynomials (e.g., $n = 1024$)
  - LWE: matrices of integers with large dimension (e.g., $752 \times 752$, $752 \times 8$)
  - MLWE: matrices of smaller polynomials (e.g., $n = 256$) of small dimension (e.g., $3 \times 3$, $3 \times 1$)
- Less structure than RLWE, more efficient than LWE

Choose $d = 3$, $n = 256$, $q = 7681$ for very conservative security

Messages in "standard" format

No dependency on particular multiplication algorithm

Possibility for further compression of keys and ciphertext (WiP)

Easy to scale security by changing $d$
The design of Kyber (WiP)

- Use **Module-Lattices** and MLWE
  - RLWE: large polynomials (e.g., $n = 1024$)
  - LWE: matrices of integers with large dimension (e.g., $752 \times 752$, $752 \times 8$)
  - MLWE: matrices of smaller polynomials (e.g., $n = 256$) of small dimension (e.g., $3 \times 3$, $3 \times 1$)
- Less structure than RLWE, more efficient than LWE
- First construct IND-CPA-secure encryption of 256-bit messages
The design of Kyber (WiP)

- Use **Module-Lattices** and MLWE
  - RLWE: large polynomials (e.g., $n = 1024$)
  - LWE: matrices of integers with large dimension (e.g., $752 \times 752$, $752 \times 8$)
  - MLWE: matrices of smaller polynomials (e.g., $n = 256$) of small dimension (e.g., $3 \times 3$, $3 \times 1$)
- Less structure than RLWE, more efficient than LWE
- First construct IND-CPA-secure encryption of 256-bit messages
- Use Fujisake-Okamoto CCA transform to build **CCA-secure KEM**
  - Can be used just like NewHope (but can cache keys!)
  - Can also be used for KEM-DEM to encrypt messages
  - Can be used in authenticated key exchange (without signatures)
The design of Kyber (WiP)

- Use **Module-Lattices** and MLWE
  - RLWE: large polynomials (e.g., $n = 1024$)
  - LWE: matrices of integers with large dimension (e.g., $752 \times 752$, $752 \times 8$)
  - MLWE: matrices of smaller polynomials (e.g., $n = 256$) of small dimension (e.g., $3 \times 3$, $3 \times 1$)
- Less structure than RLWE, more efficient than LWE
- First construct IND-CPA-secure encryption of 256-bit messages
- Use Fujisake-Okamoto CCA transform to build **CCA-secure KEM**
  - Can be used just like NewHope (but can cache keys!)
  - Can also be used for KEM-DEM to encrypt messages
  - Can be used in authenticated key exchange (without signatures)
- Choose $d = 3$, $n = 256$, $q = 7681$ for very conservative security
The design of Kyber (WiP)

- Use **Module-Lattices** and MLWE
  - RLWE: large polynomials (e.g., \( n = 1024 \))
  - LWE: matrices of integers with large dimension (e.g., \( 752 \times 752, 752 \times 8 \))
  - MLWE: matrices of smaller polynomials (e.g., \( n = 256 \)) of small dimension (e.g., \( 3 \times 3, 3 \times 1 \))
- Less structure than RLWE, more efficient than LWE
- First construct IND-CPA-secure encryption of 256-bit messages
- Use Fujisake-Okamoto CCA transform to build **CCA-secure KEM**
  - Can be used just like NewHope (but can cache keys!)
  - Can also be used for KEM-DEM to encrypt messages
  - Can be used in authenticated key exchange (without signatures)
- Choose \( d = 3, n = 256, q = 7681 \) for very conservative security
- Messages in “standard” format
  - No dependency on particular multiplication algorithm
  - Possibility for further compression of keys and ciphertext (WiP)
The design of Kyber (WiP)

- Use **Module-Lattices** and MLWE
  - RLWE: large polynomials (e.g., \( n = 1024 \))
  - LWE: matrices of integers with large dimension (e.g., \( 752 \times 752 \), \( 752 \times 8 \))
  - MLWE: matrices of smaller polynomials (e.g., \( n = 256 \)) of small dimension (e.g., \( 3 \times 3 \), \( 3 \times 1 \))
- Less structure than RLWE, more efficient than LWE
- First construct IND-CPA-secure encryption of 256-bit messages
- Use Fujisake-Okamoto CCA transform to build **CCA-secure KEM**
  - Can be used just like NewHope (but can cache keys!)
  - Can also be used for KEM-DEM to encrypt messages
  - Can be used in authenticated key exchange (without signatures)
- Choose \( d = 3, n = 256, q = 7681 \) for very conservative security
- Messages in “standard” format
  - No dependency on particular multiplication algorithm
  - Possibility for further compression of keys and ciphertext (WiP)
- Easy to scale security by changing \( d \)
Kyber’s encryption scheme

$q = 7681$, $n = 256$, $d = 3$

We work with matrices of polynomials in $\mathbb{Z}_{7681}[x]/(x^{256} + 1)$ of dim. $d = 3$ and a distribution of poly with binomial coeffs. $\Psi_4$

KeyGen():

- $\text{seed} \leftarrow \{0, \ldots, 255\}^{32}$
- $A = \begin{pmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} \\
    a_{31} & a_{32} & a_{33}
\end{pmatrix} \leftarrow \text{SHAKE}(\text{seed})$
- $s, e \leftarrow \Psi_4^d$
- $b = A \cdot s + e$
- Define $pk = (\text{seed}, b)$ and $sk = s$
Kyber’s encryption scheme

$q = 7681, n = 256, d = 3$

We work with matrices of polynomials in $\mathbb{Z}_{7681}[x]/(x^{256} + 1)$ of dim. $d = 3$ and a distribution of poly with binomial coeffs. $\Psi_4$

Encrypt($pk, m \in \{0, 1\}^{256}$, coins):

- seed, $b \leftarrow pk$
- $A = SHAKE(seed)$
- $s' \leftarrow \Psi_4^d(coins, 1)$
- $e' \leftarrow \Psi_4^d(coins, 2)$
- $e'' \leftarrow \Psi_4(coins, 3)$
- $u = (s')^t \cdot A + e'$
- $v = \langle b, s' \rangle + e'' + \lfloor q/2 \rfloor \cdot \sum_i m_i x^i$
- Output $(u, v)$

Decrypt($sk, (u, v)$):

- $w = v - \langle u, s \rangle$
- for $i \in \{0, \ldots, 255\}$,
  - $m_i \leftarrow \begin{cases} 1 & \text{if } w_i \in \left(\frac{q}{4}, \frac{3 \cdot q}{4}\right) \\ 0 & \text{otherwise} \end{cases}$
- Output $(m_0, \ldots, m_{255})$
### Idea of the CCA transformation

<table>
<thead>
<tr>
<th>Alice (Server)</th>
<th>Bob (Client)</th>
</tr>
</thead>
</table>
| **Gen()**:
  \(pk, sk\leftarrow \text{KeyGen()}\) |
  \(\text{Enc}(\text{seed}, b)\):
  \(x\leftarrow \{0, \ldots, 255\}^{32}\) |
| \(\text{seed}, b\leftarrow \text{pk}\) | \(x\leftarrow \text{SHA3-256}(x)\)
  \(k, \text{coins}\leftarrow \text{SHA3-512}(x)\) |
| \(\text{Dec}(s, (u, v))\):
  \(x'\leftarrow \text{Decrypt}(s, (u, v))\) |
  \(u, v\leftarrow \text{Encrypt}((\text{seed}, b), x, \text{coins})\) |
| \(k', \text{coins}'\leftarrow \text{SHA3-512}(x')\) |
| \(u', v'\leftarrow \text{Encrypt}((\text{seed}, b), x', \text{coins}')\) |
| **verify if** \((u', v') = (u, v)\) |

### Additionally:

- Hash the public key into the coins
- Hash the ciphertext into the final key
Kyber performance guesstimates

<table>
<thead>
<tr>
<th></th>
<th>NewHope</th>
<th>Kyber</th>
</tr>
</thead>
<tbody>
<tr>
<td>public-key bytes</td>
<td>1824</td>
<td>1280</td>
</tr>
<tr>
<td>ciphertext bytes</td>
<td>2048</td>
<td>1344</td>
</tr>
<tr>
<td>Gen cycles</td>
<td>258 246</td>
<td>296 544</td>
</tr>
<tr>
<td>Enc cycles</td>
<td>384 994</td>
<td>401 960</td>
</tr>
<tr>
<td>Dec cycles</td>
<td>86 280</td>
<td>469 872</td>
</tr>
</tbody>
</table>

- Cycles are for C reference implementation on Haswell
- Optimized implementations for Kyber will follow
- Kyber sizes are probably going to improve
http://pq-crystals.org/kyber