Radboud University

From NewHope to Kyber

Peter Schwabe
peter@cryptojedi.org
https://cryptojedi.org

April 7, 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, . . .)

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**

Ring-Learning-with-errors (RLWE)

- Let $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$
- Let χ be an *error distribution* on $\mathcal{R}_q$
- Let $\mathbf{s} \in \mathcal{R}_q$ be secret
- Attacker is given pairs $(\mathbf{a}, \mathbf{as} + \mathbf{e})$ with
  - $\mathbf{a}$ uniformly random from $\mathcal{R}_q$
  - $\mathbf{e}$ sampled from χ
- Task for the attacker: find $\mathbf{s}$

Ring-Learning-with-errors (RLWE)

- Let $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$
- Let χ be an *error distribution* on $\mathcal{R}_q$
- Let $\mathbf{s} \in \mathcal{R}_q$ be secret
- Attacker is given pairs $(\mathbf{a}, \mathbf{as} + \mathbf{e})$ with
  - $\mathbf{a}$ uniformly random from $\mathcal{R}_q$
  - $\mathbf{e}$ sampled from χ
- Task for the attacker: find $\mathbf{s}$
- Common choice for χ: discrete Gaussian
- Common optimization for protocols: fix $\mathbf{a}$

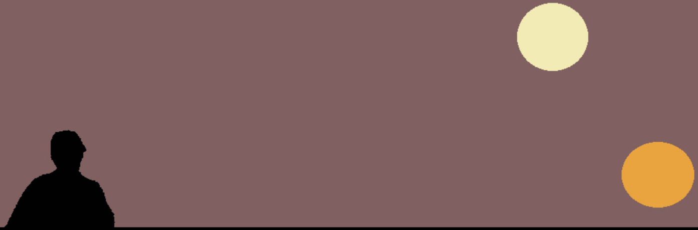
Alice (server)		Bob (client)
$\mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi$		$\mathbf{s}', \mathbf{e}' \xleftarrow{\$} \chi$
$\mathbf{b} \leftarrow \mathbf{a}\mathbf{s} + \mathbf{e}$	$\xrightarrow{\mathbf{b}}$	$\mathbf{u} \leftarrow \mathbf{a}\mathbf{s}' + \mathbf{e}'$
	$\xleftarrow{\mathbf{u}}$	

Alice has $\mathbf{t} = \mathbf{u}\mathbf{s} = \mathbf{a}\mathbf{s}\mathbf{s}' + \mathbf{e}'\mathbf{s}$

Bob has $\mathbf{t}' = \mathbf{b}\mathbf{s}' = \mathbf{a}\mathbf{s}\mathbf{s}' + \mathbf{e}\mathbf{s}'$

- Secret and noise polynomials $\mathbf{s}, \mathbf{s}', \mathbf{e}, \mathbf{e}'$ are small
- $\mathbf{t}$ and $\mathbf{t}'$ are *approximately* the same

POST-QUANTUM KEY EXCHANGE



A NEW HOPE

ERDEM ALKIM
LÉO DUCAS
THOMAS PÖPPELMANN
PETER SCHWABE

USENIX Security 2016: NewHope

- Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)
- Use reconcilation to go from approximate agreement to agreement
  - Originally proposed by Ding (2012)
  - Improvements by Peikert (2014)
  - More improvements in NewHope

USENIX Security 2016: NewHope

- Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)
- Use reconcilation to go from approximate agreement to agreement
  - Originally proposed by Ding (2012)
  - Improvements by Peikert (2014)
  - More improvements in NewHope
- NewHope-Simple (2016): Scrap complex reconciliation (pay 6.25% increase in ciphertext size)

USENIX Security 2016: NewHope

- Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)
- Use reconcilation to go from approximate agreement to agreement
  - Originally proposed by Ding (2012)
  - Improvements by Peikert (2014)
  - More improvements in NewHope
- NewHope-Simple (2016): Scrap complex reconciliation (pay 6.25% increase in ciphertext size)
- Very conservative parameters ($n = 1024, q = 12289$)
- Centered binomial noise ψ_k (HW(a)$-$HW(b) for k-bit a, b)
- Achieve ≈ 256 bits of post-quantum security according to very conservative analysis
- Higher security, shorter messages, and $> 10\times$ speedup

USENIX Security 2016: NewHope

- Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)
- Use reconcilation to go from approximate agreement to agreement
  - Originally proposed by Ding (2012)
  - Improvements by Peikert (2014)
  - More improvements in NewHope
- NewHope-Simple (2016): Scrap complex reconciliation (pay 6.25% increase in ciphertext size)
- Very conservative parameters ($n = 1024, q = 12289$)
- Centered binomial noise ψ_k (HW(a)$-$HW(b) for k-bit a, b)
- Achieve ≈ 256 bits of post-quantum security according to very conservative analysis
- Higher security, shorter messages, and $> 10\times$ speedup

USENIX Security 2016: NewHope

- Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)
- Use reconcilation to go from approximate agreement to agreement
    - Originally proposed by Ding (2012)
    - Improvements by Peikert (2014)
    - More improvements in NewHope
- NewHope-Simple (2016): Scrap complex reconciliation (pay 6.25% increase in ciphertext size)
- Very conservative parameters ($n = 1024, q = 12289$)
- Centered binomial noise ψ_k (HW(a)−HW(b) for k-bit a, b)
- Achieve ≈ 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 reconcilation to go from approximate agreement to agreement
    - Originally proposed by Ding (2012)
    - Improvements by Peikert (2014)
    - More improvements in NewHope
- NewHope-Simple (2016): Scrap complex reconciliation (pay 6.25% increase in ciphertext size)
- Very conservative parameters ($n = 1024, q = 12289$)
- Centered binomial noise ψ_k (HW(a)−HW(b) for k-bit a, b)
- Achieve ≈ 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

USENIX Security 2016: NewHope

- Improve IEEE S&P 2015 results by Bos, Costello, Naehrig, Stebila (BCNS)
- Use reconcilation to go from approximate agreement to agreement
  - Originally proposed by Ding (2012)
  - Improvements by Peikert (2014)
  - More improvements in NewHope
- NewHope-Simple (2016): Scrap complex reconciliation (pay 6.25% increase in ciphertext size)
- Very conservative parameters ($n = 1024, q = 12289$)
- Centered binomial noise ψ_k (HW(a)−HW(b) for k-bit a, b)
- Achieve ≈ 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

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."*

Conclusions for Google's experiment

"[. . .] we did not find any unexpected impediment to deploying something like NewHope. There were no reported problems caused by enabling it."

Conclusions for Google's experiment

"[. . .] 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.)"

Conclusions for Google's experiment

"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?

Are we done? Is the Internet safe again?

Disadvantages of NewHope

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

Are we done? Is the Internet safe again?

Disadvantages of NewHope

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

Are we done? Is the Internet safe again?

Disadvantages of NewHope

- Security analysis assumes that we have an LWE instance
- Structure of **R**LWE is ignored
- Somewhat large messages ($\approx$ 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)

Are we done? Is the Internet safe again?

Disadvantages of NewHope

- Security analysis assumes that we have an LWE instance
- Structure of **R**LWE is ignored
- Somewhat large messages ($\approx$ 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!

KYBER

THE KEM

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., 752×752, 752×8)
  - MLWE: matrices of smaller polynomials (e.g., $n = 256$) of small dimension (e.g., 3×3, 3×1)
- Less structure than RLWE, more efficient than LWE

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., 752×752, 752×8)
  - MLWE: matrices of smaller polynomials (e.g., $n = 256$) of small dimension (e.g., 3×3, 3×1)
- 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., 752×752, 752×8)
  - MLWE: matrices of smaller polynomials (e.g., $n = 256$) of small dimension (e.g., 3×3, 3×1)
- Less structure than RLWE, more efficient than LWE
- First construct IND-CPA-secure encryption of 256-bit messages
- Use Targhi-Unruh 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., 752×752, 752×8)
  - MLWE: matrices of smaller polynomials (e.g., $n = 256$) of small dimension (e.g., 3×3, 3×1)
- Less structure than RLWE, more efficient than LWE
- First construct IND-CPA-secure encryption of 256-bit messages
- Use Targhi-Unruh 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., 752×752, 752×8)
  - MLWE: matrices of smaller polynomials (e.g., $n = 256$) of small dimension (e.g., 3×3, 3×1)
- Less structure than RLWE, more efficient than LWE
- First construct IND-CPA-secure encryption of 256-bit messages
- Use Targhi-Unruh 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
  - Still depend on NTT (sampling of matrix $\mathbf{A}$)
  - 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., 752×752, 752×8)
  - MLWE: matrices of smaller polynomials (e.g., $n = 256$) of small dimension (e.g., 3×3, 3×1)
- Less structure than RLWE, more efficient than LWE
- First construct IND-CPA-secure encryption of 256-bit messages
- Use Targhi-Unruh 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
  - Still depend on NTT (sampling of matrix $\mathbf{A}$)
  - 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. Ψ_4

KeyGen():

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

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. Ψ_4

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

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

Decrypt(sk, $(\mathbf{u}, v)$):

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

Idea of the CCA transformation

Alice (Server)		Bob (Client)
<u>Gen():</u>		<u>Enc(seed, $\mathbf{b}$):</u>
pk, sk$\leftarrow$KeyGen()		$x \leftarrow \{0, \ldots, 255\}^{32}$
seed, $\mathbf{b}\leftarrow$pk	$\xrightarrow{\text{seed},\mathbf{b}}$	$x \leftarrow \mathrm{SHAKE}(x, 512)$
		k, coins, $q \leftarrow \mathrm{SHAKE}(x, 768)$
	$\xleftarrow{\mathbf{u}, v, q}$	$\mathbf{u}, v \leftarrow \mathrm{Encrypt}((\text{seed}, \mathbf{b}), x, \text{coins})$
<u>Dec($\mathbf{s}$, ($\mathbf{u}$, v)):</u>		
$x' \leftarrow \mathrm{Decrypt}(\mathbf{s}, (\mathbf{u}, v))$		
$k', coins', q' \leftarrow \mathrm{SHAKE}(x', 768)$		
$\mathbf{u}', v' \leftarrow \mathrm{Encrypt}((\text{seed}, \mathbf{b}), x', \text{coins}')$		
verify if $(\mathbf{u}', v', q') = (\mathbf{u}, v, q)$		

Idea of the CCA transformation

Alice (Server)		Bob (Client)
Gen():		Enc(seed, $\mathbf{b}$):
pk, sk$\leftarrow$KeyGen()		$x \leftarrow \{0, \ldots, 255\}^{32}$
seed, $\mathbf{b} \leftarrow$pk	$\xrightarrow{\text{seed}, \mathbf{b}}$	$x \leftarrow$SHAKE$(x, 512)$
		k, coins, $q \leftarrow$SHAKE$(x, 768)$
	$\xleftarrow{\mathbf{u}, v, q}$	$\mathbf{u}, v \leftarrow$Encrypt((seed, $\mathbf{b}$), x, coins)
Dec($\mathbf{s}$, ($\mathbf{u}$, v)):		
$x' \leftarrow$ Decrypt($\mathbf{s}$, ($\mathbf{u}$, v))		
k', $coins'$, $q' \leftarrow$SHAKE$(x', 768)$		
$\mathbf{u}', v' \leftarrow$Encrypt((seed, $\mathbf{b}$), x', coins$'$)		
verify if $(\mathbf{u}', v', q') = (\mathbf{u}, v, q)$		

Additionally:

- Hash the public key into x
  - Multi-target protection (for coins)
  - Turn into contributory KEM
- Hash the ciphertext into the final key

Kyber performance guesstimates

	NewHope	Kyber
public-key bytes	1 824	1 088
ciphertext bytes	2 048	1 152
Gen cycles	258 246	296 016
Enc cycles	384 994	395 948
Dec cycles	86 280	460 164

- Cycles are for C reference implementation on Haswell
- Optimized implementations for Kyber will follow

Stay tuned

http://pq-crystals.org/kyber