6 years of NIST PQC – looking back and ahead

Peter Schwabe

September 29, 2022
Part I – looking back
Part I – looking back

This talk is biased.
Part I – looking back

This talk is biased – after all, NIST PQC is a competition.
Part I – looking back

This talk is biased – after all, NIST PQC is a competition.

• Cryptographic engineering point of view
Part I – looking back

This talk is biased – after all, NIST PQC is a competition.

- Cryptographic engineering point of view
- Looking back at 6 years of NIST PQC:
  - What went well
  - What went not so well
Part I – looking back

This talk is biased – after all, NIST PQC is a competition.

- Cryptographic engineering point of view
- Looking back at 6 years of NIST PQC:
  - What went well (in work by other people)
  - What went not so well (in “our submissions”)
Our submissions

**CRYSTALS-Kyber**
- Roberto Avanzi
- Joppe Bos
- Jintai Ding
- Léo Ducas
- Eike Kiltz
- Tancrède Lepoint
- Vadim Lyubashevsky
- Peter Schwabe
- Gregor Seiler
- Damien Stehle

**CRYSTALS-Dilithium**
- Léo Ducas
- Eike Kiltz
- Tancrède Lepoint
- Vadim Lyubashevsky
- Peter Schwabe
- Gregor Seiler
- Damien Stehlé
- Shi Bai

**SPHINCS**
- Jean-Philippe Aumasson
- Daniel J. Bernstein
- Ward Beullens
- Christoph Dobruiung
- Maria Eichlseder
- Scott Fluhrer
- Stefan-Lukas Gazdag
- Andreas Hülsing
- Stefan Köbl
- Tanja Lange
- Martin M. Lauridsen
- Florian Mendel
- Ruben Niederhagen
- Christian Rechberger
- Joost Rijneveld
- Peter Schwabe
- Bas Westerbaan
“A complete written specification of the algorithms shall be included, consisting of all necessary mathematical operations, equations, tables, diagrams, and parameters that are needed to implement the algorithms. The document shall include design rationale and an explanation for all the important design decisions that are made”

—Dustin Moody, February 24, 2016 (PQCRYPTO 2016)
1. Designing – so many decisions!

- Choose concrete parameters for different security levels
- All randomness from `randombytes`
or just a seed?
- Fix sampling algorithms (e.g., constant-time sorting)
- Select symmetric primitives
- Concretize domain separation

...tradeoffs, tradeoffs, tradeoffs...
1. Designing – so many decisions!

“Oh, you mean numbers?!”

—Giulio Malavolta, September 2022
1. Designing – so many decisions!

“Oh, you mean numbers?!”

—Giulio Malavolta, September 2022

• Choose concrete parameters for different security levels
• All randomness from `randombytes` or just a seed?
• Fix sampling algorithms (e.g., constant-time sorting)
• Select symmetric primitives
• Concretize domain separation
1. Designing – so many decisions!

“Oh, you mean numbers?!”

—Giulio Malavolta, September 2022

• Choose concrete parameters for different security levels
• All randomness from randombytes or just a seed?
• Fix sampling algorithms (e.g., constant-time sorting)
• Select symmetric primitives
• Concretize domain separation
• Tradeoffs, tradeoffs, tradeoffs...
Dilithium – v3.0 vs. v3.1

- Message hash only 384 bits
- 192 bits of sec. against collisions
- Not sufficient for NIST level 5
SPHINCS+, round 3

- Security relies on DM – SPR (Distinct-function multi-target second preimage resistance) of underlying hash function
- Three different choices of hash function: SHA-256, SHAKE-256, Haraka
SPHINCS$^+$, round 3

- Security relies on DM – SPR (Distinct-function multi-target second preimage resistance) of underlying hash function
- Three different choices of hash function: SHA-256, SHAKE-256, Haraka
- Sydney Antonov, April 20, 2022: attack against DM-SPR of SHA-256
- Attack cost higher than NIST level 1, but lower than level 3 and 5
SPHINCS\(^+\), round 3

- Security relies on DM – SPR (Distinct-function multi-target second preimage resistance) of underlying hash function
- Three different choices of hash function: SHA-256, SHAKE-256, Haraka
- Sydney Antonov, April 20, 2022: attack against DM-SPR of SHA-256
- Attack cost higher than NIST level 1, but lower than level 3 and 5

“This is an interesting attack that does demonstrate that our real hash functions do not perfectly behave like random oracles”

—Andreas Hülsing, April 21, 2022
1. Will the schemes selected now be widely used?

2. Will those schemes survive in the long run?
2. Proving

“Submitters are not required to provide a proof of security, although such proofs will be considered if they are available.”

—NIST PQC, Call for Proposals
Kyber round 1

- LPR scheme’s public key is $t = As + e$
- This is an (R/M)LWE sample and assumed to be uniform in the proof
Kyber round 1

• LPR scheme’s public key is $t = As + e$
• This is an (R/M)LWE sample and assumed to be uniform in the proof
• Kyber in round 1 compressed this (round off low bits)
• $t' = \text{Decompress(Compress}(t))$ is not uniform
Kyber round 1

- LPR scheme’s public key is \( t = As + e \)
- This is an (R/M)LWE sample and assumed to be uniform in the proof
- Kyber in round 1 compressed this (round off low bits)
- \( t' = \text{Decompress} (\text{Compress} (t)) \) is not uniform
- Reduction from MLWE in round-1 Kyber was invalid:

“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
SPHINCS$^+$ – original proof

- Reduce from second-preimage resistance
- Place challenge $x = H(y)$ inside hash chains
- Forgery produces preimage of $x$ with certain prob.
- Reduction hopes to obtain second preimage $y' \neq y$ with $x = H(y')$
SPHINCS$^+$ – original proof

- Reduce from second-preimage resistance
- Place challenge $x = H(y)$ inside hash chains
- Forgery produces preimage of $x$ with certain prob.
- Reduction hopes to obtain second preimage $y' \neq y$ with $x = H(y')$
- Problem: $\text{len}(x) = \text{len}(y)$
- Second preimage does not exist with high probability
SPHINCS$^+$ – original proof

- Reduce from second-preimage resistance
- Place challenge $x = H(y)$ inside hash chains
- Forgery produces preimage of $x$ with certain prob.
- Reduction hopes to obtain second preimage $y' \neq y$ with $x = H(y')$
- Problem: $\text{len}(x) = \text{len}(y)$
- Second preimage does not exist with high probability
- Forger can refuse to forge if there is a second preimage
2. Proving – so many failure modes

- Proof is wrong
2. Proving – so many failure modes

- Proof is wrong
  - Theorem is correct

- Theorem is also wrong
- Scheme is still (possibly) secure
- Scheme is efficiently broken
- Proof doesn't apply to the scheme
- Proof correct, but theorem "insufficient"

- Example: attack hides in non-tightness
- Proof (and possibly theorem) too vague
- Theorem and proof correct, but not very useful

- "A is secure if A is secure"
2. Proving – so many failure modes

• Proof is wrong
  • Theorem is correct
  • Theorem is also wrong
    • Scheme is still (possibly) secure
    • Scheme is efficiently broken
2. Proving – so many failure modes

• Proof is wrong
  • Theorem is correct
  • Theorem is also wrong
    • Scheme is still (possibly) secure
    • Scheme is efficiently broken

• Proof doesn’t apply to the scheme
2. Proving – so many failure modes

- Proof is wrong
  - Theorem is correct
  - Theorem is also wrong
    - Scheme is still (possibly) secure
    - Scheme is efficiently broken
- Proof doesn’t apply to the scheme
- Proof correct, but theorem “insufficient”
2. Proving – so many failure modes

- Proof is wrong
  - Theorem is correct
  - Theorem is also wrong
    - Scheme is still (possibly) secure
    - Scheme is efficiently broken
- Proof doesn't apply to the scheme
- Proof correct, but theorem “insufficient”
  - Example: attack hides in non-tightness
2. Proving – so many failure modes

• Proof is wrong
  • Theorem is correct
  • Theorem is also wrong
    • Scheme is still (possibly) secure
    • Scheme is efficiently broken

• Proof doesn’t apply to the scheme

• Proof correct, but theorem “insufficient”
  • Example: attack hides in non-tightness

• Proof (and possibly theorem) too vague
2. Proving – so many failure modes

• Proof is wrong
  • Theorem is correct
  • Theorem is also wrong
    • Scheme is still (possibly) secure
    • Scheme is efficiently broken

• Proof doesn’t apply to the scheme
• Proof correct, but theorem “insufficient”
  • Example: attack hides in non-tightness
• Proof (and possibly theorem) too vague
• Theorem and proof correct, but not very useful

“A is secure if A is secure”
3. Implementing

“NISTPQC, despite being an important and timely project, has produced the largest regression ever in the quality of cryptographic software. This will not be easy to fix.”

—Daniel J. Bernstein, October 5, 2018
3. Implementing – Exhibit A

Dilithium commit on Dec. 28, 2017

- Bug in Dilithium sampler
- Two consecutive coefficients are equal
- Allows key recovery
- Reported by Peter Pessl on Dec. 27, 2017
PQClean

- Joint work with Matthias Kannwischer, Joost Rijneveld, John Schanck, Douglas Stebila, Goutam Tamvada, Thom Wiggers
- Test harness for PQC implementations
- Integrate reference implementations
  - Run through test harness
  - “clean up”
### 3. Implementing – Exhibit B

<table>
<thead>
<tr>
<th>Flaw</th>
<th>KEMs</th>
<th>Sigs</th>
<th>Flaw</th>
<th>KEMs</th>
<th>Sigs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory safety</td>
<td>3</td>
<td>4</td>
<td>Endianness assumptions</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>Signed integer overflow</td>
<td>3</td>
<td>1</td>
<td>Platform-specific behavior</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Alignment assumptions</td>
<td>4</td>
<td>4</td>
<td>Variable-Length Arrays</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Other Undefined Behavior</td>
<td>1</td>
<td>1</td>
<td>Compiler extensions</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Dead code</td>
<td>3</td>
<td>4</td>
<td>Integer sizes</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Global state</td>
<td>2</td>
<td>1</td>
<td>Non-constant time</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Licensing unclear</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
“In almost every scheme we identified “unclean” code, ranging from missing casts to memory safety problems and other forms of undefined behavior.”

—https://eprint.iacr.org/2022/337
Questions about the range analysis of iNTT for "Faster Kyber and Dilithium on the Cortex-M4" #226

JunhaoHuang opened this issue on Mar 3 · 4 comments

JunhaoHuang commented on Mar 3 · edited

Hi team, I am reading the Kyber code regarding the recent paper "Faster Kyber and Dilithium on the Cortex-M4", and I have a question about the matrix-vector product and Better Accumulation part regarding the \_stack version code.

I see that using the better accumulation technique in the \_speed version code, we can reduce each element of the output vector of matrix-vector product down to (-q,q). Since poly_invtt is normally used after the matrix-vector product, the range of the input vector of poly\_invtt lies in (-q,q) in the \_speed version code. The invtt function works in this situation.

What I wonder is that in the \_stack version code, the matacc function actually uses the previous double basemul accumulation function, and it should produce the result vector with element in (-kq, kq), k is the security parameter of Kyber. For Kyber1024, the range of each polynomial element that invtt takes should be (-4q,4q). However, the invtt function is the same as the \_speed version code. The first four layers of the light butterflies in invtt involve some additions and subtractions without multiplication. Therefore, For Kyber1024 in the \_stack version code, two layers of addition/subtraction might overflow the int16_t. I wonder how you deal with this problem in the \_stack code and why does it still work?
“...two layers of addition/subtraction might overflow the int16_t. I wonder how you deal with this problem in the f_stack code and why does it still work?”
“...two layers of addition/subtraction might overflow the int16_t. I wonder how you deal with this problem in the f_stack code and why does it still work?”

“...On your question on why it still works, I believe that this is an edge case that does not get triggered by the testing scripts.”
There is a bug in the inverse of NTT in Saber. But the bug is triggered with a very low probability that it is not triggered on testing.
“The idea is that participants put their algorithms into the ring, and then we all spend a few years beating on each other’s submissions.”

—Bruce Schneier, August 8, 2022
def recover_bit(ct, bit):
    assert bit < len(ct) // 4000
    ts = [struct.unpack('BB', ct[i:i+2]) for i in range(4000*bit, 4000*(bit+1), 2)]
    xs, ys = [a for a, b in ts if b == 1], [a for a, b in ts if b == 2]
    return sum(xs) / len(xs) >= sum(ys) / len(ys)

def decrypt(ct):
    res = sum(recover_bit(ct, b) << b for b in range(len(ct) // 4000))
    return int.to_bytes(res, len(ct) // 4000 // 8, 'little')
4. Attacking in 2022

ia.cr/2022/975
4. Attacking in 2022

Castryck, Decru: *An efficient key recovery attack on SIDH*
4. Attacking in 2022

Castryck, Decru: An efficient key recovery attack on SIDH

• SIDH was “A decade unscathed” (Craig Costello, ePrint 2021/543)
Castryck, Decru: *An efficient key recovery attack on SIDH*

- SIDH was “A decade unscathed” (Craig Costello, ePrint 2021/543)
- SIKE *lowered* parameters during NIST PQC (following Jaques, Schanck: *Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE* (ePrint 2019/103))
4. Attacking in 2022

Castryck, Decru: *An efficient key recovery attack on SIDH*

- SIDH was “A decade unscathed” (Craig Costello, ePrint 2021/543)
- SIKE *lowered* parameters during NIST PQC (following Jaques, Schanck: *Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE* (ePrint 2019/103))
- Competent, smart people tried to break it (e.g., Martindale, Panny: *How to not break SIDH* (ePrint 2019/558))
Castryck, Decru: *An efficient key recovery attack on SIDH*

- SIDH was “A decade unscathed” (Craig Costello, ePrint 2021/543)
- SIKE lowered parameters during NIST PQC (following Jaques, Schanck: *Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE* (ePrint 2019/103))
- Competent, smart people tried to break it (e.g., Martindale, Panny: *How to not break SIDH* (ePrint 2019/558))

Yet, **full break without any “warning”**
4. Attacking – even more attacks!
“I don’t know if you’re familiar with this website, twitter.com? If you like crypto drama, this is where you go. Except if you go to the pqc-forum, which is also... generally... it’s even better”.

—Bor de Kock, August 17, 2022
“I don’t know if you’re familiar with this website, twitter.com? If you like crypto drama, this is where you go. Except if you go to the pqc-forum, which is also... generally... it’s even better”.

—Bor de Kock, August 17, 2022

[https://www.youtube.com/watch?v=kXcYy8L0I9s](https://www.youtube.com/watch?v=kXcYy8L0I9s), starting at 20:22.
5. Communicating – pqc-forum
“Follow the “Rule of 1” and the “Rule of n”: When you speak, make 1 point and then let others speak, and when in a group of “n” people, speak “1/nth” of the time.”

—Aspiration Participants:Guidelines

Download all mails, run statistics
5. Communicating – pqc-forum

Download almost all mails, run statistics
Download almost all mails, run statistics

- pqc-forum had 666 threads ("conversations") on Sep. 14, 2022
- First mail by Dustin Moody from Aug. 1, 2016
- I have 2805 mails (first one from Nov. 2, 2016)
5. Communicating – pqc-forum

for i in mails/*;do
    FROM=$(grep ^From: $i | head -n 1 | sed "s/From:\ //" | sed "s/.*<\([^>]*\)>/\1/")
    if [ "$FROM" = "pqc-forum@list.nist.gov" ]; then
        FROM=$(grep ^X-Original-From: $i | head -n 1 | sed "s/X-Original-From:\ //" | sed "s/.*<\([^>]*\)>/\1/"
    fi
    echo $FROM
done | sort | uniq -c | sort -n

• 369 sender addresses
• Sometimes multiple addresses for one person
• 131 addresses sent just one mail
• 275 addresses sent at most 5 mails
5. Communicating – pqc-forum

The “Top 10”

1. address1 407
2. address2 146
3. address3 113
4. address4 106
5. address5 100
6. address6 81
7. address7 69
8. address8 68
9. address9 50
10. address10 47
10. address11 47

>50% of mails sent by only 15 people.

>30% of all words by non-NIST authors are from one address.
The “Top 10”

1. address1  
2. dustin.moody@nist.gov  
3. address3  
4. address4  
5. daniel.apon@nist.gov  
6. jacob.alperin-sheriff@nist.gov  
7. address7  
8. ray.perlner@nist.gov  
9. address9  
10. address10  
10. address11

50% of mails sent by only 15 people.  
30% of all words by non-NIST authors are from one address.
The “Top 10”

<table>
<thead>
<tr>
<th>Rank</th>
<th>Email Address</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>address1</td>
<td>407</td>
</tr>
<tr>
<td>2</td>
<td><a href="mailto:dustin.moody@nist.gov">dustin.moody@nist.gov</a></td>
<td>146</td>
</tr>
<tr>
<td>3</td>
<td>address3</td>
<td>113</td>
</tr>
<tr>
<td>4</td>
<td>address4</td>
<td>106</td>
</tr>
<tr>
<td>5</td>
<td><a href="mailto:daniel.apon@nist.gov">daniel.apon@nist.gov</a></td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td><a href="mailto:jacob.alperin-sheriff@nist.gov">jacob.alperin-sheriff@nist.gov</a></td>
<td>81</td>
</tr>
<tr>
<td>7</td>
<td>address7</td>
<td>69</td>
</tr>
<tr>
<td>8</td>
<td><a href="mailto:ray.perlner@nist.gov">ray.perlner@nist.gov</a></td>
<td>68</td>
</tr>
<tr>
<td>9</td>
<td>address9</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td><a href="mailto:peter@cryptojedi.org">peter@cryptojedi.org</a></td>
<td>47</td>
</tr>
<tr>
<td>10</td>
<td>address11</td>
<td>47</td>
</tr>
</tbody>
</table>

>50% of mails sent by only 15 people.

>30% of all words by non-NIST authors are from one address.
The “Top 10”

1. address1 407
2. dustin.moody@nist.gov 146
3. address3 113
4. address4 106
5. daniel.apon@nist.gov 100
6. jacob.alperin-sheriff@nist.gov 81
7. address7 69
8. ray.perliner@nist.gov 68
9. address9 50
10. peter@cryptojedi.org 47
10. address11 47

>50% of mails sent by only 15 people.
The “Top 10”

<table>
<thead>
<tr>
<th>Rank</th>
<th>Email Address</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>address1</td>
<td>407</td>
</tr>
<tr>
<td>2</td>
<td><a href="mailto:dustin.moody@nist.gov">dustin.moody@nist.gov</a></td>
<td>146</td>
</tr>
<tr>
<td>3</td>
<td>address3</td>
<td>113</td>
</tr>
<tr>
<td>4</td>
<td>address4</td>
<td>106</td>
</tr>
<tr>
<td>5</td>
<td><a href="mailto:daniel.apon@nist.gov">daniel.apon@nist.gov</a></td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td><a href="mailto:jacob.alperin-sheriff@nist.gov">jacob.alperin-sheriff@nist.gov</a></td>
<td>81</td>
</tr>
<tr>
<td>7</td>
<td>address7</td>
<td>69</td>
</tr>
<tr>
<td>8</td>
<td>ray <a href="mailto:perlner@nist.gov">perlner@nist.gov</a></td>
<td>68</td>
</tr>
<tr>
<td>9</td>
<td>address9</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td><a href="mailto:peter@cryptojedi.org">peter@cryptojedi.org</a></td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>address11</td>
<td>47</td>
</tr>
</tbody>
</table>

>50% of mails sent by only 15 people.

>30% of all words by non-NIST authors are from one address.
Part II – looking ahead

- More designing, proving, implementing, attacking, communicating in rounds 4, 5, 6, . . .
- Additional scrutiny of selected algorithms
- Standardization and deployment of selected algorithms
Part II – looking ahead

• More designing, **proving, implementing**, attacking, communicating in rounds 4, 5, 6,…
• **Additional scrutiny** of selected algorithms
• Standardization and **deployment** of selected algorithms
Formosa Crypto

- Effort to formally verify crypto
- Currently three main projects:
  - EasyCrypt proof assistant
  - jasmin programming language
  - libjade (PQ-)crypto library
- Core community of \( \approx 30-40 \) people
- Discussion forum with \( >100 \) people
The toolchain and workflow

- EasyCrypt Model (.ec files)
- EasyCrypt
- Jasmin code (.jazz, .jinc)
- Jasmin Compiler
- assembly

interactive proofs for all kinds of properties

extracts to

certifiably-compiles to

automatic safety checker
• Syntax is very C like
• Compilation is much more predictable:
  • Generally: 1 line in jasmin → 1 line in asm
  • A few exceptions, but highly predictable
  • Compiler does not schedule code
  • Compiler does not spill registers, syntactically correct code may fail to compile!
• Syntax is very C like
• Compilation is much more predictable:
  • Generally: 1 line in jasmin → 1 line in asm
  • A few exceptions, but highly predictable
  • Compiler does not schedule code
  • Compiler does not spill registers, syntactically correct code may fail to compile!
• Compiler is formally proven to preserve semantics
• Compiler is formally proven to preserve constant-time property
• Separate compiler run to ensure memory safety (statically!)
(Speculative) constant-time

Guaranteed constant-time code

• Information-flow type system, distinguish high (secret) and low (public) data
• Prevent branching and memory indexing on secret data
• Compilation is proven to preserve this property!
(Speculative) constant-time

Guaranteed constant-time code

- Information-flow type system, distinguish high (secret) and low (public) data
- Prevent branching and memory indexing on secret data
- Compilation is proven to preserve this property!

Guaranteed Spectre v1 protection

- Extend type system: “transient” (public, but may be secret during misspeculation)
- Keep predicate to track misspeculation
- Mask transient data with predicate
- Approach is Selective Speculative Load Hardening (selSLH)
- Performance overhead for crypto: <1%

Formally verified Kyber (WIP)
Formally verified Kyber (WIP)
Formally verified Kyber (WIP)
Formally verified Kyber (WIP)

- Connect to EasyCrypt IND-CPA proof of HMLWE.CPAPKE-Core EC model
- Connect to EasyCrypt IND-CCA proof of HMLWE.CCAKEM EC model

Joint work with José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Benjamin Grégoire, Andreas Hülsing, Vincent Laporte, Jean-Christophe Léchenet, Tiago Oliveira, Hugo Pacheco, Miguel Quaresma, Antoine Séré, and Pierre-Yves Strub
Had NIST required computer-verified software and proofs,
Had NIST required computer-verified software and proofs,

- we would have had way fewer bugs in PQC software;
- we would have much higher confidence in all proofs;
Had NIST required computer-verified software and proofs,

- we would have had way fewer bugs in PQC software;
- we would have much higher confidence in all proofs;
- attacks could be much more focused;
Had NIST required computer-verified software and proofs,

- we would have had way fewer bugs in PQC software;
- we would have much higher confidence in all proofs;
- attacks could be much more focused;
- we could heavily reduce “noise” in discussions;
Had NIST required computer-verified software and proofs,

• we would have had way fewer bugs in PQC software;
• we would have much higher confidence in all proofs;
• attacks could be much more focused;
• we could heavily reduce “noise” in discussions;

... and we would probably not have had a single submission.
Interested? Get involved!

https://formosa-crypto.org

https://formosa-crypto.zulipchat.com/