Implementing Post-Quantum Cryptography on the Cortex M4

Matthias Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen

Radboud University, Nijmegen, The Netherlands

2018-09-13
RIOT Summit 2018
The quantum threat

- RIOT Summit 2017: Post Quantum Crypto for the IoT, by Simona Samardjiska
The quantum threat

- RIOT Summit 2017: Post Quantum Crypto for the IoT, by Simona Samardjiska

- A large quantum computer can do..
  - Useful things: complex simulations that solve {global warming, world hunger, diseases, ..}

- RSA is broken.
- ECC is broken.
- Symmetric crypto is broken.. but easily fixed.
The quantum threat

► RIOT Summit 2017: Post Quantum Crypto for the IoT, by Simona Samardjiska

► A large quantum computer can do..
  ► Useful things: complex simulations that solve
    {global warming, world hunger, diseases, ..}
  ► Destructive things: **break crypto**
The quantum threat

- RIOT Summit 2017: Post Quantum Crypto for the IoT, by Simona Samardjiska

- A large quantum computer can do..
  - Useful things: complex simulations that solve 
    \{global warming, world hunger, diseases, ..\}
  - Destructive things: break crypto

- RSA is broken.
The quantum threat

- RIOT Summit 2017: Post Quantum Crypto for the IoT, by Simona Samardjiska

- A large quantum computer can do..
  - Useful things: complex simulations that solve 
    \{global warming, world hunger, diseases, ..\}
  - Destructive things: break crypto

- RSA is broken.
- ECC is broken.
The quantum threat

- RIOT Summit 2017: Post Quantum Crypto for the IoT, by Simona Samardjiska

- A large quantum computer can do..
  - Useful things: complex simulations that solve {global warming, world hunger, diseases, ..}
  - Destructive things: break crypto

- RSA is broken.
- ECC is broken.
- Symmetric crypto is broken.. but easily fixed.
**Grover:** Search in $\mathcal{O}(\sqrt{n})$

**Shor:** Factorize in $\text{poly}(n)$
Grover: Search in $O(\sqrt{n})$

Shor: Factorize in $\text{poly}(n)$

$\approx$ solve DLP
So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be ‘ok’
So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be ‘ok’

- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain’t one
So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be ‘ok’

- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain’t one

- Lattices

\[ \text{As} + e \not\Rightarrow s \]
So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be ‘ok’

- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain’t one

- Lattices
  - \( A_s + e \not\Rightarrow s \)

- Error-correcting codes
  - \( m\hat{G} + z \not\Rightarrow m \)
So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be ‘ok’

- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain’t one

- Lattices

- Error-correcting codes

- Multivariate quadratics

- Supersingular isogenies

- Hashes
So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be ‘ok’

- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain’t one

- Lattices
- Error-correcting codes
- Multivariate quadratics
- Supersingular isogenies

\[ \text{As} + e \not\approx s \]
\[ m\hat{G} + z \not\approx m \]
\[ y = MQ(x) \]
\[ \phi : E_1 \rightarrow E_2 \]
So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be ‘ok’

- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain’t one

- Lattices
- Error-correcting codes
- Multivariate quadratics
- Supersingular isogenies
- Hashes
- ...
So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might be ‘ok’

- Asymmetric crypto is fun!
  - 99 problems, but the DLP ain’t one

- Lattices
- Error-correcting codes
- Multivariate quadratics
- Supersingular isogenies
- Hashes
- ...  
- post-quantum RSA

\[ As + e \not\Rightarrow s \]
\[ mG + z \not\Rightarrow m \]
\[ y = MQ(x) \]
\[ \phi : E_1 \rightarrow E_2 \]
\[ \mathcal{H}(x) \not\Rightarrow x \]

‘What if we used 1 GiB keys?’
NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017
NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017

- 82 submissions
NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017
- 82 submissions
- \( \approx 58 \) still unbroken in Round 1
NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017

- 82 submissions
- \( \approx 58 \) still unbroken in Round 1

- PQC Standardization conference: April 11-13, 2018
  - 2nd conference: Aug/Sept 2019
- Final ‘portfolio:’ in 3 - 5 years
NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
  - See also: AES and SHA-3 competitions
- Deadline: November 30, 2017
- 82 submissions
- $\approx 58$ still unbroken in Round 1
- PQC Standardization conference: April 11-13, 2018
  - 2nd conference: Aug/Sept 2019
- Final ‘portfolio:’ in 3 - 5 years
- ‘Not a competition’
PQC on the IOT

▶ Algorithm flexibility?
  ▶ PQC is an active research field
PQC on the IOT

- Algorithm flexibility?
  - PQC is an active research field
    ⇒ things break
PQC on the IOT

- Algorithm flexibility?
  - PQC is an active research field
    - things break
    - not yet standardized
PQC on the IOT

- Algorithm flexibility?
  - PQC is an active research field
    ⇒ things break
    ⇒ not yet standardized

- Key sizes, ciphertext sizes, signature sizes ..?
- Speed ..?
PQC on the IOT

- Algorithm flexibility?
  - PQC is an active research field
    ⇒ things break
    ⇒ not yet standardized

- Key sizes, ciphertext sizes, signature sizes ..?
- Speed ..?

“It’s big and it’s slow”
PQC on the IOT

- Algorithm flexibility?
  - PQC is an active research field
    ⇒ things break
    ⇒ not yet standardized

- Key sizes, ciphertext sizes, signature sizes ..?

- Speed ..?

“It’s big and it’s slow”
– everyone, always
PQC on the IOT

▶ Algorithm flexibility?
  ▶ PQC is an active research field
    ⇒ things break
    ⇒ not yet standardized

▶ Key sizes, ciphertext sizes, signature sizes ..?
▶ Speed ..?

“It’s big and it’s slow”
− everyone, always

▶ This project: where do we stand? How do we improve?
PQM4 framework

- Deliverable of the EU H2020 PQCRYPTO project
  - ‘Small devices’

- Target platform: Cortex M4 (STM32 M4 discovery board)
  - STM32F407VG

- ‘PQC on M4’ framework
  - Testing
  - Benchmarking
PQM4 framework

- Build system: linkable static library per scheme
PQM4 framework

- Build system: linkable static library per scheme
- Internal correctness tests
  - Sign & verify, encapsulate & decapsulate, ..
PQM4 framework

- Build system: linkable static library per scheme
- Internal correctness tests
  - Sign & verify, encapsulate & decapsulate, ..
- Test vector comparison
  - WIP: NIST known answer test files

- Benchmarking cycle count and stack usage
- Easy integration of new schemes/implementations
- NIST level 3
- Accepting pull requests!

- FrodoKEM-640-cSHAKE, KINDI-256-3-4-2, Kyber-768,
  NewHope-1024-CCA-KEM, NTRU-HRSS-KEM-701, Saber, SIKE-p571,
  Streamlined NTRU Prime 4591761, Dilithium-III, qTesla-I, qTesla-III-size,
  qTesla-III-speed, SPHINCS+-SHAKE256-128s
PQM4 framework

- Build system: linkable static library per scheme
- Internal correctness tests
  - Sign & verify, encapsulate & decapsulate, ..
- Test vector comparison
  - WIP: NIST known answer test files
- Benchmarking cycle count and stack usage
PQM4 framework

- Build system: linkable static library per scheme
- Internal correctness tests
  - Sign & verify, encapsulate & decapsulate, ..
- Test vector comparison
  - WIP: NIST known answer test files
- Benchmarking cycle count and stack usage
- Easy integration of new schemes/implementations
  - NIST level 3
  - Accepting pull requests!
PQM4 framework

- Build system: linkable static library per scheme
- Internal correctness tests
  - Sign & verify, encapsulate & decapsulate, ..
- Test vector comparison
  - WIP: NIST known answer test files
- Benchmarking cycle count and stack usage
- Easy integration of new schemes/implementations
  - NIST level 3
  - Accepting pull requests!

FrodoKEM-640-cSHAKE, KINDI-256-3-4-2, Kyber-768, NewHope-1024-CCA-KEM, NTRU-HRSS-KEM-701, Saber, SIKE-p571, Streamlined NTRU Prime 4591761, Dilithium-III, qTesla-I, qTesla-III-size, qTesla-III-speed, SPHINCS+-SHAKE256-128s
Adding new schemes

1. Copy into subdirectory of crypto_kem/ or crypto_sign/
   - e.g. crypto_kem/newhope1024cca/m4 or crypto_sign/dilithium/ref
Adding new schemes

1. Copy into subdirectory of crypto_kem/ or crypto_sign/
   ▶ e.g. crypto_kem/newhope1024cca/m4 or crypto_sign/dilithium/ref

2. Write a Makefile to build libpqm4.a
   ▶ Flexible template included
Adding new schemes

1. Copy into subdirectory of crypto_kem/ or crypto_sign/
   - e.g. crypto_kem/newhope1024cca/m4 or crypto_sign/dilithium/ref

2. Write a Makefile to build libpqm4.a
   - Flexible template included

3. Optionally, for pure C: add libpqhost.a host target
Adding new schemes

1. Copy into subdirectory of crypto_kem/ or crypto_sign/
   ▶ e.g. crypto_kem/newhope1024cca/m4 or crypto_sign/dilithium/ref

2. Write a Makefile to build libpqm4.a
   ▶ Flexible template included

3. Optionally, for pure C: add libpqhost.a host target

4. Optionally, replace SHA3 calls
   ▶ PQM4 contains highly optimized SHA3 & variants
Adding new schemes

1. Copy into subdirectory of `crypto_kem/` or `crypto_sign/`
   ▶ e.g. `crypto_kem/newhope1024cca/m4` or `crypto_sign/dilithium/ref`

2. Write a Makefile to build `libpqm4.a`
   ▶ Flexible template included

3. Optionally, for pure C: add `libpqhost.a` host target

4. Optionally, replace SHA3 calls
   ▶ PQM4 contains highly optimized SHA3 & variants
PQM4 on RIOT OS

- Ongoing work by Sara Stadler & Jonas Wloka (Uni. Bremen)
  https://github.com/jowlo/pqm4
PQM4 on RIOT OS

- Ongoing work by Sara Stadler & Jonas Wloka (Uni. Bremen)
  https://github.com/jowlo/pqm4
- Adapted build system
- Uses RIOT’s hardware interfacing functions
PQM4 on RIOT OS

- Ongoing work by Sara Stadler & Jonas Wloka (Uni. Bremen)
  https://github.com/jowlo/pqm4

- Adapted build system
- Uses RIOT’s hardware interfacing functions
- Report some success on M0 and M3 targets
PQM4 on RIOT OS

- Ongoing work by Sara Stadler & Jonas Wloka (Uni. Bremen)
  https://github.com/jowlo/pqm4

- Adapted build system
- Uses RIOT’s hardware interfacing functions
- Report some success on M0 and M3 targets
- Crypto schemes are not ready for production use
Optimized implementations: lattice-based schemes

- Lattice-based schemes are popular: 28 NIST submissions
Optimized implementations: lattice-based schemes

- Lattice-based schemes are popular: 28 NIST submissions

- Ideal lattices: arithmetic in a polynomial ring
  - Very fast (beats ECC!)
  - Acceptable sizes (1-2KiB ciphertexts/keys)
Optimized implementations: lattice-based schemes

- Lattice-based schemes are popular: 28 NIST submissions
- Ideal lattices: arithmetic in a polynomial ring
  - Very fast (beats ECC!)
  - Acceptable sizes (1-2KiB ciphertexts/keys)
- Fast polynomial multiplication, coefficients modulo q
Optimized implementations: lattice-based schemes

- Lattice-based schemes are popular: 28 NIST submissions

- Ideal lattices: arithmetic in a polynomial ring
  - Very fast (beats ECC!)
  - Acceptable sizes (1-2KiB ciphertexts/keys)

- Fast polynomial multiplication, coefficients modulo \( q \)
  - Varying degree \( n \)
  - Prime \( q \) or \( q = 2^n \)
Optimized implementations: lattice-based schemes

- Lattice-based schemes are popular: 28 NIST submissions

- Ideal lattices: arithmetic in a polynomial ring
  - Very fast (beats ECC!)
  - Acceptable sizes (1-2KiB ciphertexts/keys)

- Fast polynomial multiplication, coefficients modulo $q$
  - Varying degree $n$
  - Prime $q$ or $q = 2^k$
How to multiply a polynomial

- Depends on degree $n$
How to multiply a polynomial

- Depends on degree $n$
- Break down into smaller $n$
How to multiply a polynomial

- Depends on degree $n$
- Break down into smaller $n$
- Schoolbook
  - i.e. $O(n^2)$ multiplications
How to multiply a polynomial

- Depends on degree $n$
- Break down into smaller $n$

- Schoolbook
  - i.e. $\mathcal{O}(n^2)$ multiplications

- Karatsuba
  - trade a $\frac{1}{2}n$-mult for additions
How to multiply a polynomial

- Depends on degree $n$
- Break down into smaller $n$

- Schoolbook
  - i.e. $O(n^2)$ multiplications

- Karatsuba
  - trade a $\frac{1}{2}n$-mult for additions

- Toom-3 / Toom-4
  - split into 3 or 4 parts
How to multiply a polynomial

- Depends on degree \( n \)
- Break down into smaller \( n \)
- Schoolbook
  - i.e. \( O(n^2) \) multiplications
- Karatsuba
  - trade a \( \frac{1}{2} n \)-mult for additions
- Toom-3 / Toom-4
  - split into 3 or 4 parts
Preliminary results

- Arbitrary degree $n \leq 1024$
- Python scripts generate ARMv7M assembly
## Speed records

- Directly applies to several NIST submissions
- Work in progress

<table>
<thead>
<tr>
<th>scheme</th>
<th>params</th>
<th>impl</th>
<th>key gen</th>
<th>encaps</th>
<th>decaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>KINDI</td>
<td>$n = 256$ $q = 2^{14}$</td>
<td>ref</td>
<td>22,942k</td>
<td>29,656k</td>
<td>37,817k</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ours</td>
<td>1,101k</td>
<td>1,494k</td>
<td>1,726k</td>
</tr>
<tr>
<td>NTRU-HRSS</td>
<td>$n = 701$ $q = 2^{13}$</td>
<td>ref</td>
<td>204,854k</td>
<td>5,166k</td>
<td>15,067k</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ours</td>
<td>164,090k</td>
<td>451k</td>
<td>917k</td>
</tr>
<tr>
<td>NTRU-KEM</td>
<td>$n = 743$ $q = 2^{11}$</td>
<td>ref</td>
<td>53,326k</td>
<td>7,144k</td>
<td>12,782k</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ours</td>
<td>5,445k</td>
<td>1,825k</td>
<td>2,145k</td>
</tr>
<tr>
<td>SABER</td>
<td>$n = 256$ $q = 2^{13}$</td>
<td>ref</td>
<td>7,123k</td>
<td>9,471k</td>
<td>12,304k</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1]</td>
<td>1,147k</td>
<td>1,444k</td>
<td>1,543k</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ours</td>
<td>982k</td>
<td>1,277k</td>
<td>1,323k</td>
</tr>
<tr>
<td>RLizard</td>
<td>$n = 1024$ $q = 2^{11}$</td>
<td>ref</td>
<td>26,428k</td>
<td>32,211k</td>
<td>57,344k</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ours</td>
<td>626k</td>
<td>1,513k</td>
<td>1,986k</td>
</tr>
</tbody>
</table>

Interested?
Find us at https://github.com/mupq/pqm4

All code available as public domain where possible.