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, ..}

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

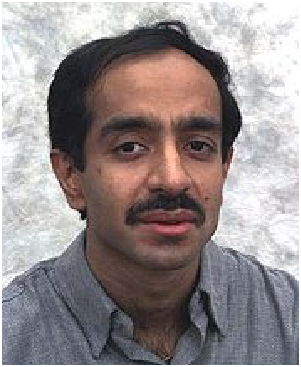
- RIOT Summit 2017: Post Quantum Crypto for the IoT, by Simona Samardjiska
- A large quantum computer can do..
  - Useful things: complex simulations that solve {global warming, world hunger, diseases, ..}
  - Destructive things: **break crypto**
- RSA is broken.
- ECC is broken.

The quantum threat

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

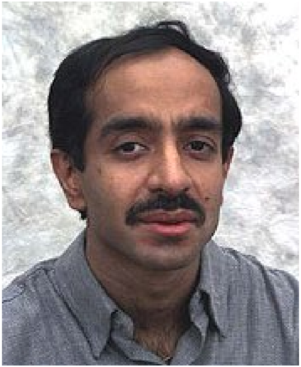
- A large quantum computer can do..
  - Useful things: complex simulations that solve {global warming, world hunger, diseases, ..}
  - Destructive things: **break crypto**

- RSA is broken.
- ECC is broken.
- Symmetric crypto is broken.. but easily fixed.



Grover: Search in $\mathcal{O}(\sqrt{n})$

Shor: Factorize in $poly(n)$



Grover: Search in $\mathcal{O}(\sqrt{n})$

Shor: Factorize in $poly(n)$

$\approx$ 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 $\mathbf{As} + \mathbf{e} \not\Rightarrow \mathbf{s}$

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 $\mathbf{As} + \mathbf{e} \not\Rightarrow \mathbf{s}$
  - Error-correcting codes $\mathbf{m}\widehat{\mathbf{G}} + \mathbf{z} \not\Rightarrow \mathbf{m}$

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	$\mathbf{As} + \mathbf{e} \not\Rightarrow \mathbf{s}$
Error-correcting codes	$\mathbf{m}\widehat{\mathbf{G}} + \mathbf{z} \not\Rightarrow \mathbf{m}$
Multivariate quadratics	$\mathbf{y} = \mathcal{M}\mathcal{Q}(\mathbf{x})$

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	$\mathbf{As} + \mathbf{e} \not\Rightarrow \mathbf{s}$
Error-correcting codes	$\mathbf{m}\widehat{\mathbf{G}} + \mathbf{z} \not\Rightarrow \mathbf{m}$
Multivariate quadratics	$\mathbf{y} = \mathcal{MQ}(\mathbf{x})$
Supersingular isogenies	$\phi : E_1 \to E_2$

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	$\mathbf{As} + \mathbf{e} \not\Rightarrow \mathbf{s}$
Error-correcting codes	$\mathbf{m}\widehat{\mathbf{G}} + \mathbf{z} \not\Rightarrow \mathbf{m}$
Multivariate quadratics	$\mathbf{y} = \mathcal{M}\mathcal{Q}(\mathbf{x})$
Supersingular isogenies	$\phi : E_1 \to E_2$
Hashes	$\mathcal{H}(x) \not\Rightarrow x$
...	

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	$\mathbf{As} + \mathbf{e} \not\Rightarrow \mathbf{s}$
Error-correcting codes	$\mathbf{m}\widehat{\mathbf{G}} + \mathbf{z} \not\Rightarrow \mathbf{m}$
Multivariate quadratics	$\mathbf{y} = \mathcal{MQ}(\mathbf{x})$
Supersingular isogenies	$\phi : E_1 \to E_2$
Hashes	$\mathcal{H}(x) \not\Rightarrow x$
. . .	
post-quantum RSA	*‘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
    $\Rightarrow$ 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

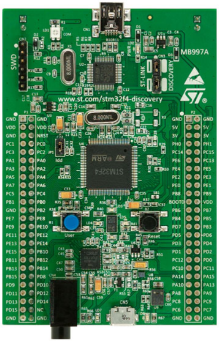
- Key sizes, ciphertext sizes, signature sizes ..?
- Speed ..?

> *"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



STM32 F4

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

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 $\boldsymbol{q = 2^k}$

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. $\mathcal{O}(n^2)$ 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

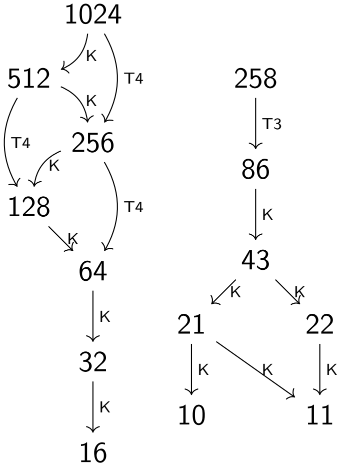
- Depends on degree n
- Break down into smaller n

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

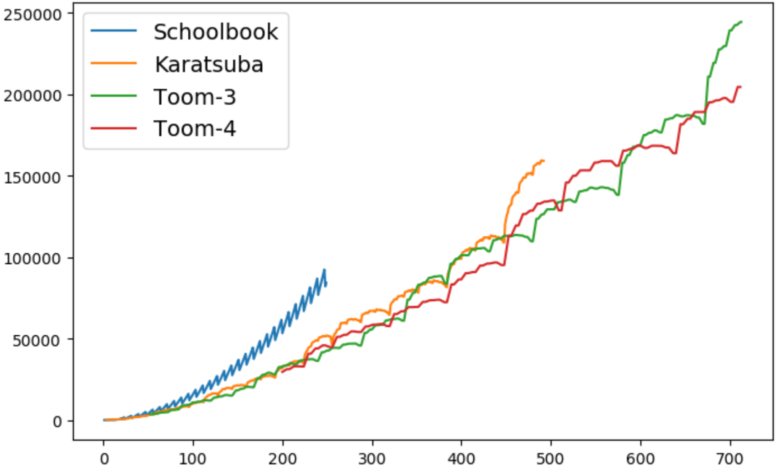
- Depends on degree *n*
- Break down into smaller *n*

- Schoolbook
  - i.e. $\mathcal{O}(n^2)$ 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 (≤ 1024)
- Python scripts generate ARMv7M assembly



Speed records

- Directly applies to several NIST submissions
  - Work in progress

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

[1] Karmakar, A., Mera, J. M. B., Roy, S. S., & Verbauwhede, I. (2018). Saber on ARM. IACR Transactions on Cryptographic Hardware and Embedded Systems, 243-266.

Interested?

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

All code available as public domain where possible.