Faster polynomial multiplication on Cortex-M4 to speed up NIST candidates

Matthias Kannwischer, Joost Rijneveld and Peter Schwabe

Radboud University, Nijmegen, The Netherlands

2019-02-08
DiS Lunch Talk

The quantum threat

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

The quantum threat

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

The quantum threat

- 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

- 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

- 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)

So all is lost?

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

So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might even 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 even 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 even 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 even 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 $\mathbf{y} = \mathcal{MQ}(\mathbf{x})$

So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might even 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 even 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 $\mathcal{H}(x) \not\Rightarrow x$
  - ...

So all is lost?

- Symmetric crypto is fine!
  - Grover queries are expensive: AES-128 might even 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
- Standardize ‘portfolio’ of signatures and KEMs
  - See also: AES and SHA-3 competitions

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
- Standardize ‘portfolio’ of signatures and KEMs
  - See also: AES and SHA-3 competitions

- Nov ‘17: 82 submissions 8 involving RU

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
- Standardize ‘portfolio’ of signatures and KEMs
  - See also: AES and SHA-3 competitions

- Nov ‘17: 82 submissions 8 involving RU
- mid ‘18: $\approx$ 58 unbroken in Round 1 8 involving RU

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
- Standardize 'portfolio' of signatures and KEMs
  - See also: AES and SHA-3 competitions

Nov '17:	82 submissions	8 involving RU
mid '18:	$\approx$ 58 unbroken in Round 1	8 involving RU
Jan '19:	26 (17 + 9) in Round 2	

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
- Standardize 'portfolio' of signatures and KEMs
  - See also: AES and SHA-3 competitions

- Nov '17: 82 submissions 8 involving RU
- mid '18: $\approx$ 58 unbroken in Round 1 8 involving RU
- Jan '19: 26 (17 + 9) in Round 2 8 involving RU!

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
- Standardize 'portfolio' of signatures and KEMs
  - See also: AES and SHA-3 competitions

Nov '17:	82 submissions	8 involving RU
mid '18:	$\approx$ 58 unbroken in Round 1	8 involving RU
Jan '19:	26 (17 + 9) in Round 2	8 involving RU!

- Final selection: 2 - 4 years

NIST Post-Quantum not-a-competition

- National Institute of Standards and Technology
- Standardize 'portfolio' of signatures and KEMs
  - See also: AES and SHA-3 competitions

Nov '17:	82 submissions	8 involving RU
mid '18:	$\approx$ 58 unbroken in Round 1	8 involving RU
Jan '19:	26 (17 + 9) in Round 2	8 involving RU!

- Final selection: 2 - 4 years

- "Not a competition"
- "Performance will play a larger role in the 2nd round"

Post-quantum on small devices

"It's big and it's slow"

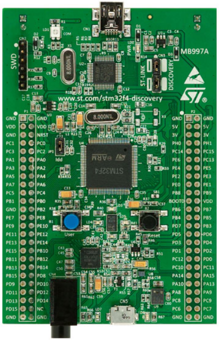
Post-quantum on small devices

"It's big and it's slow"
– everyone, always

Post-quantum on small devices

"It's big and it's slow"
– everyone, always

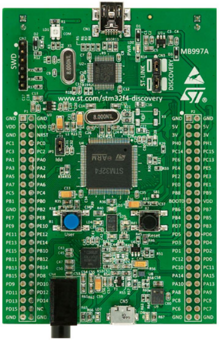
- STM32F4: ARM Cortex-M4
  - 32-bit, ARMv7-M
  - 192 KiB RAM, 168 MHz



Post-quantum on small devices

"It's big and it's slow"
– everyone, always

- STM32F4: ARM Cortex-M4
  - 32-bit, ARMv7-M
  - 192 KiB RAM, 168 MHz
  - used in Crypto Eng. course
- PQM4: test and optimize on the Cortex-M4
  - github.com/mupq/pqm4



STM32 F4

Learning with errors (LWE)

- Given uniform $\mathbf{A} \in \mathbb{Z}_q^{k \times \ell}$
- Given "noise distribution" χ
- Given samples $\mathbf{As} + \mathbf{e}$, with $\mathbf{e} \leftarrow \chi$

Learning with errors (LWE)

- Given uniform $\mathbf{A} \in \mathbb{Z}_q^{k \times \ell}$
- Given "noise distribution" χ
- Given samples $\mathbf{As} + \mathbf{e}$, with $\mathbf{e} \leftarrow \chi$

- Find $\mathbf{s}$

Learning with errors (LWE)

- Given uniform $\mathbf{A} \in \mathbb{Z}_q^{k \times \ell}$
- Given "noise distribution" χ
- Given samples $\mathbf{As} + \mathbf{e}$, with $\mathbf{e} \leftarrow \chi$
- Find $\mathbf{s}$
- Structured lattices: work in $\mathbb{Z}_q[x]/f$

Lattice-based KEMs – the basic idea

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

$$\begin{array}{llll} \text{Alice has} & \mathbf{v} & = \mathbf{us} & = \mathbf{ass}' + \mathbf{e}'\mathbf{s} \\ \text{Bob has} & \mathbf{v}' & = \mathbf{bs}' & = \mathbf{ass}' + \mathbf{es}' \end{array}$$

- Secret and noise $\mathbf{s}, \mathbf{s}', \mathbf{e}, \mathbf{e}'$ are small
- $\mathbf{v}$ and $\mathbf{v}'$ are *approximately* the same

Lattice-based KEMs submitted to NIST

- 22 of Round 1 submissions are lattice-based KEMs
  - 9 progressed to Round 2
- Large design space with many trade-offs:

Lattice-based KEMs submitted to NIST

- 22 of Round 1 submissions are lattice-based KEMs
  - 9 progressed to Round 2
- Large design space with many trade-offs:
  - LWE vs. LWR
  - LWE vs. Ring-LWE vs. Module-LWE

Lattice-based KEMs submitted to NIST

- 22 of Round 1 submissions are lattice-based KEMs
  - 9 progressed to Round 2
- Large design space with many trade-offs:
  - LWE vs. LWR
  - LWE vs. Ring-LWE vs. Module-LWE
  - Prime q vs. power-of-two q

Lattice-based KEMs submitted to NIST

- 22 of Round 1 submissions are lattice-based KEMs
  - 9 progressed to Round 2
- Large design space with many trade-offs:
  - LWE vs. LWR
  - LWE vs. Ring-LWE vs. Module-LWE
  - Prime q vs. power-of-two q
  - Prime n vs. power-of-two n

Lattice-based KEMs submitted to NIST

- 22 of Round 1 submissions are lattice-based KEMs
  - 9 progressed to Round 2
- Large design space with many trade-offs:
  - LWE vs. LWR
  - LWE vs. Ring-LWE vs. Module-LWE
  - Prime q vs. power-of-two q
  - Prime n vs. power-of-two n
  - NTRU vs. LWE/LWR ("quotient" vs. "product")

Lattice-based KEMs submitted to NIST

- 22 of Round 1 submissions are lattice-based KEMs
  - 9 progressed to Round 2
- Large design space with many trade-offs:
  - LWE vs. LWR
  - LWE vs. Ring-LWE vs. Module-LWE
  - Prime q vs. power-of-two q
  - Prime n vs. power-of-two n
  - NTRU vs. LWE/LWR ("quotient" vs. "product")
  - ...

Lattice-based KEMs submitted to NIST

- 22 of Round 1 submissions are lattice-based KEMs
  - 9 progressed to Round 2
- Large design space with many trade-offs:
  - LWE vs. LWR
  - LWE vs. Ring-LWE vs. Module-LWE
  - Prime q vs. power-of-two q
  - Prime n vs. power-of-two n
  - NTRU vs. LWE/LWR ("quotient" vs. "product")
  - ...

5 lattice-based KEMs

- RLizard, Saber, NTRU-HRSS, NTRUEncrypt, and Kindi
- Arithmetic in $\mathbb{Z}_{2^m}[x]/f$
  - $11 \leq m \leq 14$
  - $256 \leq n = \deg(f) \leq 1024$

5 lattice-based KEMs

- RLizard, Saber, NTRU-HRSS, NTRUEncrypt, and Kindi
- Arithmetic in $\mathbb{Z}_{2^m}[x]/f$
  - $11 \leq m \leq 14$
  - $256 \leq n = \deg(f) \leq 1024$

- Why these schemes?
  - Co-submitters of NTRU-HRSS
  - NTRU-HRSS could be faster than Round5
  - Only Saber has been optimized on Cortex-M4 (CHES 2018)

5 lattice-based KEMs

- RLizard, Saber, NTRU-HRSS, NTRUEncrypt, and Kindi
- Arithmetic in $\mathbb{Z}_{2^m}[x]/f$
  - $11 \leq m \leq 14$
  - $256 \leq n = \deg(f) \leq 1024$

- Why these schemes?
  - Co-submitters of NTRU-HRSS
  - NTRU-HRSS could be faster than Round5
  - Only Saber has been optimized on Cortex-M4 (CHES 2018)

- How to optimize those 5 KEMs?
  - Multiplication of polynomials with n coefficients over $\mathbb{Z}_{2^m}[x]$

5 lattice-based KEMs

Merged: NTRU

- ~~RLizard~~, Saber, NTRU-HRSS, NTRUEncrypt, ~~and Kindi~~
- Arithmetic in $\mathbb{Z}_{2^m}[x]/f$
  - $11 \leq m \leq 14$
  - $256 \leq n = \deg(f) \leq 1024$

- Why these schemes?
  - Co-submitters of NTRU-HRSS
  - NTRU-HRSS could be faster than Round5
  - Only Saber has been optimized on Cortex-M4 (CHES 2018)

- How to optimize those 5 KEMs?
  - Multiplication of polynomials with n coefficients over $\mathbb{Z}_{2^m}[x]$

Schoolbook multiplication (e.g., $n = 6$)

$$a = a_5 x^5 + a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0$$
$$b = b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0$$

					$a_5 b_0$	$a_4 b_0$	$a_3 b_0$	$a_2 b_0$	$a_1 b_0$	$a_0 b_0$
				$a_5 b_1$	$a_4 b_1$	$a_3 b_1$	$a_2 b_1$	$a_1 b_1$	$a_0 b_1$	
			$a_5 b_2$	$a_4 b_2$	$a_3 b_2$	$a_2 b_2$	$a_1 b_2$	$a_0 b_2$		
		$a_5 b_3$	$a_4 b_3$	$a_3 b_3$	$a_2 b_3$	$a_1 b_3$	$a_0 b_3$			
	$a_5 b_4$	$a_4 b_4$	$a_3 b_4$	$a_2 b_4$	$a_1 b_4$	$a_0 b_4$				
$a_5 b_5$	$a_4 b_5$	$a_3 b_5$	$a_2 b_5$	$a_1 b_5$	$a_0 b_5$					

36 multiplications, 25 additions

Karatsuba's method

- Split inputs in half: $a = a_1 x^{n/2} + a_0, b = b_1 x^{n/2} + b_0$

Karatsuba's method

- Split inputs in half: $a = a_1 x^{n/2} + a_0, b = b_1 x^{n/2} + b_0$

$$a \cdot b = (a_1 x^{n/2} + a_0)(b_1 x^{n/2} + b_0)$$

Karatsuba's method

- Split inputs in half: $a = a_1 x^{n/2} + a_0, b = b_1 x^{n/2} + b_0$

$$\begin{aligned} a \cdot b &= (a_1 x^{n/2} + a_0)(b_1 x^{n/2} + b_0) \\ &= a_1 b_1 x^n + (a_1 b_0 + a_0 b_1) x^{n/2} + a_0 b_0 \end{aligned}$$

Karatsuba's method

- Split inputs in half: $a = a_1 x^{n/2} + a_0, b = b_1 x^{n/2} + b_0$

$$
\begin{aligned}
a \cdot b &= (a_1 x^{n/2} + a_0)(b_1 x^{n/2} + b_0) \\
&= a_1 b_1 x^n + (a_1 b_0 + a_0 b_1) x^{n/2} + a_0 b_0 \\
&= a_1 b_1 x^n + \\
&\qquad ((a_1 + a_0)(b_0 + b_1) - a_1 b_1 - a_0 b_0) x^{n/2} \\
&\qquad + a_0 b_0
\end{aligned}
$$

Karatsuba's method

- Split inputs in half: $a = a_1 x^{n/2} + a_0, b = b_1 x^{n/2} + b_0$

$$
\begin{aligned}
a \cdot b &= (a_1 x^{n/2} + a_0)(b_1 x^{n/2} + b_0) \\
&= a_1 b_1 x^n + (a_1 b_0 + a_0 b_1) x^{n/2} + a_0 b_0 \\
&= a_1 b_1 x^n + \\
&\qquad ((a_1 + a_0)(b_0 + b_1) - a_1 b_1 - a_0 b_0) x^{n/2} \\
&\qquad + a_0 b_0
\end{aligned}
$$

Karatsuba's method

- Split inputs in half: $a = a_1 x^{n/2} + a_0, b = b_1 x^{n/2} + b_0$

$$
\begin{aligned}
a \cdot b &= (a_1 x^{n/2} + a_0)(b_1 x^{n/2} + b_0) \\
&= a_1 b_1 x^n + (a_1 b_0 + a_0 b_1) x^{n/2} + a_0 b_0 \\
&= a_1 b_1 x^n + \\
&\qquad ((a_1 + a_0)(b_0 + b_1) - a_1 b_1 - a_0 b_0) x^{n/2} \\
&\qquad + a_0 b_0
\end{aligned}
$$

Karatsuba's method

- Split inputs in half: $a = a_1 x^{n/2} + a_0, b = b_1 x^{n/2} + b_0$

$$\begin{aligned} a \cdot b &= (a_1 x^{n/2} + a_0)(b_1 x^{n/2} + b_0) \\ &= a_1 b_1 x^n + (a_1 b_0 + a_0 b_1) x^{n/2} + a_0 b_0 \\ &= a_1 b_1 x^n + \\ &\qquad ((a_1 + a_0)(b_0 + b_1) - a_1 b_1 - a_0 b_0) x^{n/2} \\ &\qquad + a_0 b_0 \end{aligned}$$

- Only need 3 half-size multiplications (instead of 4)
- Need some additional additions and subtractions

Karatsuba's method

- Split inputs in half: $a = a_1 x^{n/2} + a_0, b = b_1 x^{n/2} + b_0$

$$\begin{aligned} a \cdot b &= (a_1 x^{n/2} + a_0)(b_1 x^{n/2} + b_0) \\ &= a_1 b_1 x^n + (a_1 b_0 + a_0 b_1) x^{n/2} + a_0 b_0 \\ &= a_1 b_1 x^n + \\ &\qquad ((a_1 + a_0)(b_0 + b_1) - a_1 b_1 - a_0 b_0) x^{n/2} \\ &\qquad + a_0 b_0 \end{aligned}$$

- Only need 3 half-size multiplications (instead of 4)
- Need some additional additions and subtractions
- Can be applied recursively
  - At some threshold schoolbooks are more efficient

Toom-Cook's method

- Generalizes Karatsuba

Toom-Cook's method

- Generalizes Karatsuba
- Toom-3: split in 3 parts
  - 5 instead of 9 multiplications

Toom-Cook's method

- Generalizes Karatsuba
- Toom-3: split in 3 parts
  - 5 instead of 9 multiplications
- Toom-4: split in 4 parts
  - 7 instead of 16 multiplications

Toom-Cook's method

- Generalizes Karatsuba
- Toom-3: split in 3 parts
  - 5 instead of 9 multiplications
- Toom-4: split in 4 parts
  - 7 instead of 16 multiplications
- Toom-5: split in 5 parts

Toom-Cook's method

- Generalizes Karatsuba
- Toom-3: split in 3 parts
  - 5 instead of 9 multiplications
- Toom-4: split in 4 parts
  - 7 instead of 16 multiplications
- Toom-5: split in 5 parts
  - Loses too much precision!

- Toom-Cook uses divisions in $\mathbb{Z}$, not in $\mathbb{Z}_{16}$
- Losses add up!
  - Toom-3 (1 bit) + Toom-4 (3 bits) $\Rightarrow$ multiply in $\mathbb{Z}_{12}$

What's the best method?

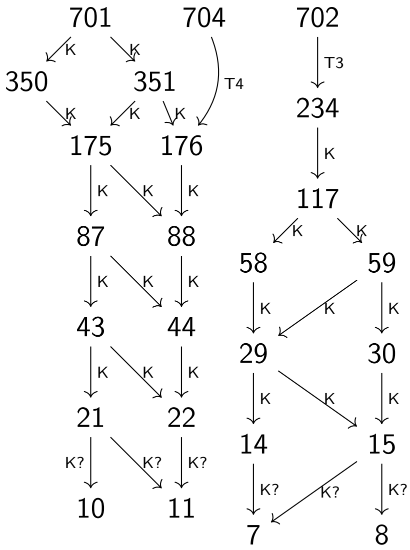
- Asymptotic: Toom-4 wins
- ... what about $n = 701$?

What's the best method?

- Asymptotic: Toom-4 wins
- ... what about $n = 701$?
- Our approach: Try all

What's the best method?

- Asymptotic: Toom-4 wins
- ... what about $n = 701$?
- Our approach: Try all
- We need
  - Fast Karatsuba for all n
  - Fast Toom-4 for all n
  - Fast Toom-3 for all n
  - Fast schoolbook for small n



Fast schoolbook multiplication

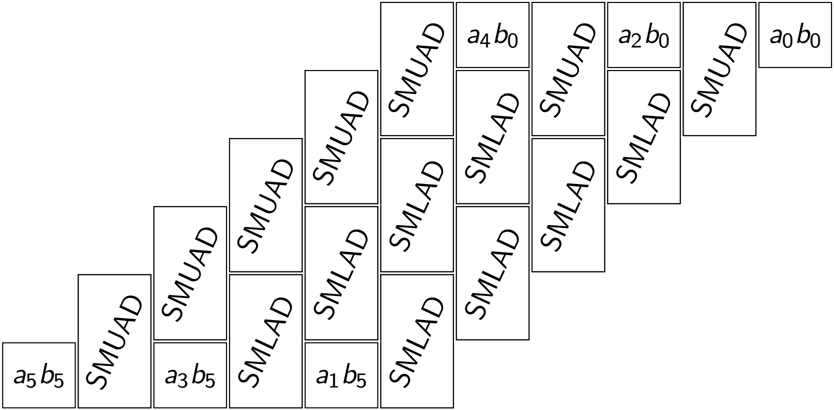
- ARMv7E-M supports SMUAD(X) and SMLAD(X)
- All in one clock cycle
- Perfect for polynomial multiplication

instruction	semantics
`smuad Ra, Rb, Rc`	$\mathtt{Ra} \leftarrow \mathtt{Rb}_\mathtt{L} \cdot \mathtt{Rc}_\mathtt{L} + \mathtt{Rb}_\mathtt{H} \cdot \mathtt{Rc}_\mathtt{H}$
`smuadx Ra, Rb, Rc`	$\mathtt{Ra} \leftarrow \mathtt{Rb}_\mathtt{L} \cdot \mathtt{Rc}_\mathtt{H} + \mathtt{Rb}_\mathtt{H} \cdot \mathtt{Rc}_\mathtt{L}$
`smlad Ra, Rb, Rc, Rd`	$\mathtt{Ra} \leftarrow \mathtt{Rb}_\mathtt{L} \cdot \mathtt{Rc}_\mathtt{L} + \mathtt{Rb}_\mathtt{H} \cdot \mathtt{Rc}_\mathtt{H} + \mathtt{Rd}$
`smladx Ra, Rb, Rc, Rd`	$\mathtt{Ra} \leftarrow \mathtt{Rb}_\mathtt{L} \cdot \mathtt{Rc}_\mathtt{H} + \mathtt{Rb}_\mathtt{H} \cdot \mathtt{Rc}_\mathtt{L} + \mathtt{Rd}$

Fast schoolbook multiplication, $n = 6$

					$a_5 b_0$	$a_4 b_0$	$a_3 b_0$	$a_2 b_0$	$a_1 b_0$	$a_0 b_0$
				$a_5 b_1$	$a_4 b_1$	$a_3 b_1$	$a_2 b_1$	$a_1 b_1$	$a_0 b_1$	
			$a_5 b_2$	$a_4 b_2$	$a_3 b_2$	$a_2 b_2$	$a_1 b_2$	$a_0 b_2$		
		$a_5 b_3$	$a_4 b_3$	$a_3 b_3$	$a_2 b_3$	$a_1 b_3$	$a_0 b_3$			
	$a_5 b_4$	$a_4 b_4$	$a_3 b_4$	$a_2 b_4$	$a_1 b_4$	$a_0 b_4$				
$a_5 b_5$	$a_4 b_5$	$a_3 b_5$	$a_2 b_5$	$a_1 b_5$	$a_0 b_5$					

Fast schoolbook multiplication, $n = 6$



Fast schoolbook multiplication: less repacking

					$a_5 b_0$	$a_4 b_0$	$a_3 b_0$	$a_2 b_0$	$a_1 b_0$	$a_0 b_0$
				$a_5 b_1$	$a_4 b_1$	$a_3 b_1$	$a_2 b_1$	$a_1 b_1$	$a_0 b_1$	
			$a_5 b_2$	$a_4 b_2$	$a_3 b_2$	$a_2 b_2$	$a_1 b_2$	$a_0 b_2$		
		$a_5 b_3$	$a_4 b_3$	$a_3 b_3$	$a_2 b_3$	$a_1 b_3$	$a_0 b_3$			
	$a_5 b_4$	$a_4 b_4$	$a_3 b_4$	$a_2 b_4$	$a_1 b_4$	$a_0 b_4$				
$a_5 b_5$	$a_4 b_5$	$a_3 b_5$	$a_2 b_5$	$a_1 b_5$	$a_0 b_5$					

Fast schoolbook multiplication: less repacking

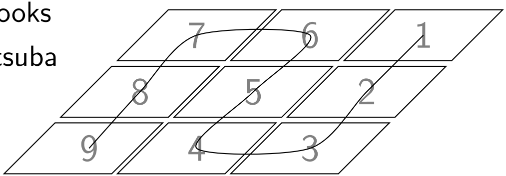
					$a_5 b_0$	$a_4 b_0$	$a_3 b_0$	$a_2 b_0$	$a_1 b_0$	$a_0 b_0$
				$a_5 b_1$	$a_4 b_1$	$a_3 b_1$	$a_2 b_1$	$a_1 b_1$	$a_0 b_1$	
			$a_5 b_2$	$a_4 b_2$	$a_3 b_2$	$a_2 b_2$	$a_1 b_2$	$a_0 b_2$		
		$a_5 b_3$	$a_4 b_3$	$a_3 b_3$	$a_2 b_3$	$a_1 b_3$	$a_0 b_3$			
	$a_5 b_4$	$a_4 b_4$	$a_3 b_4$	$a_2 b_4$	$a_1 b_4$	$a_0 b_4$				
$a_5 b_5$	$a_4 b_5$	$a_3 b_5$	$a_2 b_5$	$a_1 b_5$	$a_0 b_5$					

Fast schoolbook multiplication: less repacking

					$a_5 b_0$	$a_4 b_0$	$a_3 b_0$	$a_2 b_0$	$a_1 b_0$	$a_0 b_0$
				$a_5 b_1$	$a_4 b_1$	$a_3 b_1$	$a_2 b_1$	$a_1 b_1$	$a_0 b_1$	
			$a_5 b_2$	$a_4 b_2$	$a_3 b_2$	$a_2 b_2$	$a_1 b_2$	$a_0 b_2$		
		$a_5 b_3$	$a_4 b_3$	$a_3 b_3$	$a_2 b_3$	$a_1 b_3$	$a_0 b_3$			
	$a_5 b_4$	$a_4 b_4$	$a_3 b_4$	$a_2 b_4$	$a_1 b_4$	$a_0 b_4$				
$a_5 b_5$	$a_4 b_5$	$a_3 b_5$	$a_2 b_5$	$a_1 b_5$	$a_0 b_5$					

Exploring the design space

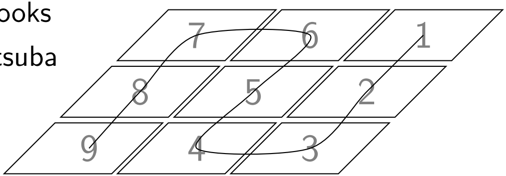
- Hand-optimize small schoolbooks
- Compose larger schoolbooks
- Recursive Toom / Karatsuba



Exploring the design space

- Hand-optimize small schoolbooks
- Compose larger schoolbooks
- Recursive Toom / Karatsuba

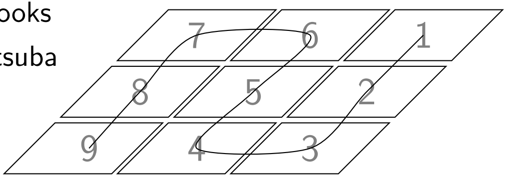
- Python that writes asm
  - Naming things



Exploring the design space

- Hand-optimize small schoolbooks
- Compose larger schoolbooks
- Recursive Toom / Karatsuba

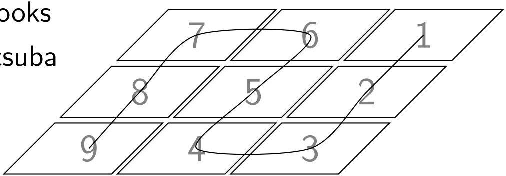
- Python that writes asm
  - Naming things
  - Calling functions



Exploring the design space

- Hand-optimize small schoolbooks
- Compose larger schoolbooks
- Recursive Toom / Karatsuba

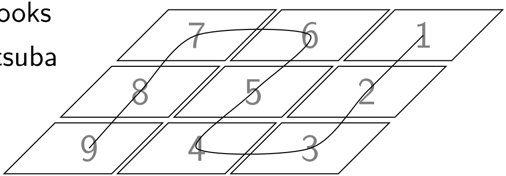
- Python that writes asm
  - Naming things
  - Calling functions
  - (Un)rolling loops



Exploring the design space

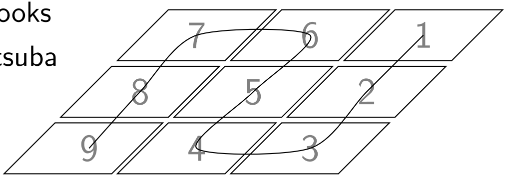
- Hand-optimize small schoolbooks
- Compose larger schoolbooks
- Recursive Toom / Karatsuba

- Python that writes asm
  - Naming things
  - Calling functions
  - (Un)rolling loops
  - Poor man's register allocation



Exploring the design space

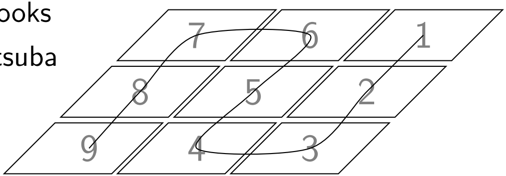
- Hand-optimize small schoolbooks
- Compose larger schoolbooks
- Recursive Toom / Karatsuba



- Python that writes asm
  - Naming things
  - Calling functions
  - (Un)rolling loops
  - Poor man's register allocation
  - Post-processing asm

Exploring the design space

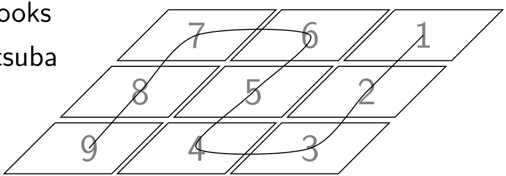
- Hand-optimize small schoolbooks
- Compose larger schoolbooks
- Recursive Toom / Karatsuba



- Python that writes asm
  - Naming things
  - Calling functions
  - (Un)rolling loops
  - Poor man's register allocation
  - Post-processing asm
  - .. Naming things

Exploring the design space

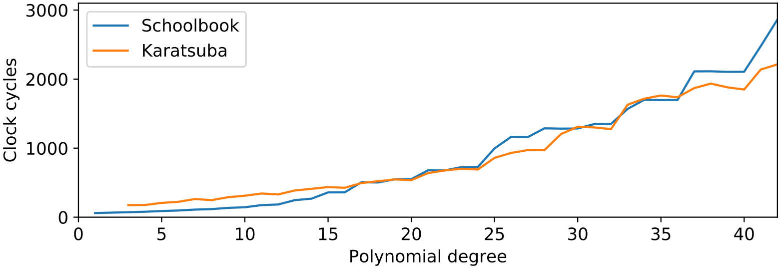
- Hand-optimize small schoolbooks
- Compose larger schoolbooks
- Recursive Toom / Karatsuba



- Python that writes asm
  - Naming things
  - Calling functions
  - (Un)rolling loops
  - Poor man's register allocation
  - Post-processing asm
  - .. Naming things

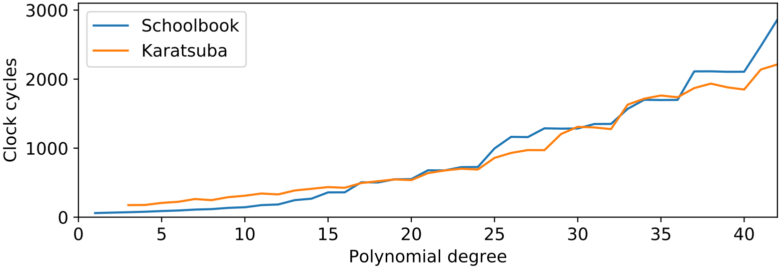
- Automated compiling + benchmarking

Schoolbook vs. Karatsuba



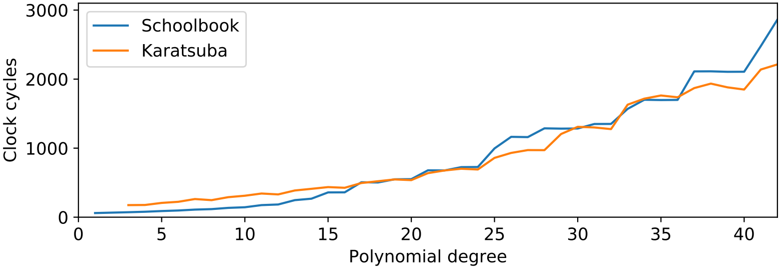
- Schoolbook is faster for $n \leq 16$

Schoolbook vs. Karatsuba



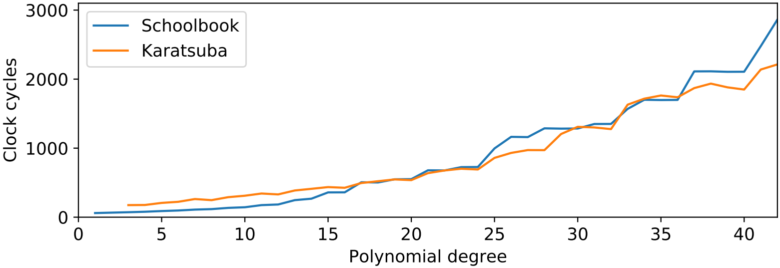
- Schoolbook is faster for $n \leq 16$
- Karatsuba is faster for $n > 36$

Schoolbook vs. Karatsuba



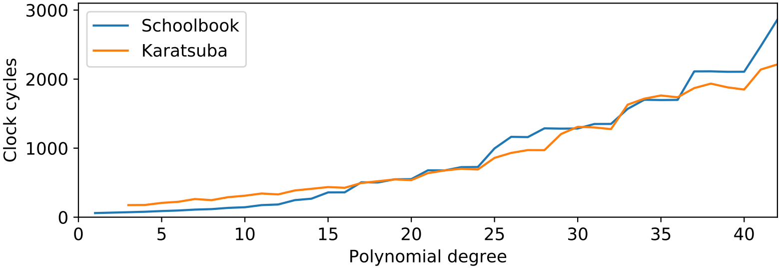
- Schoolbook is faster for $n \leq 16$
- Karatsuba is faster for $n > 36$
- We are mainly interested in $n = \{10, 11, 12, 16\}$

Schoolbook vs. Karatsuba



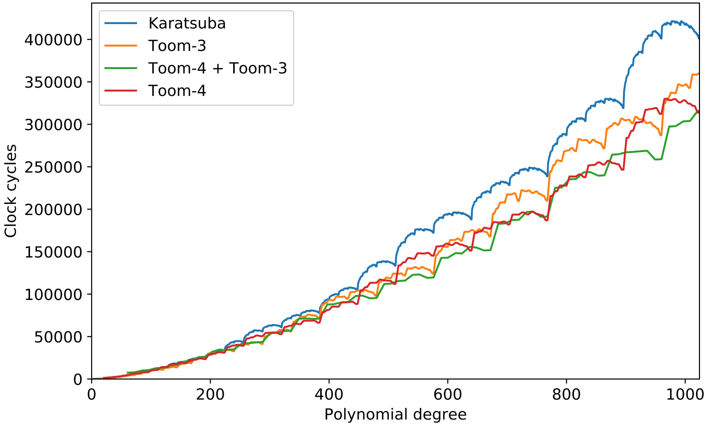
- Schoolbook is faster for $n \leq 16$
- Karatsuba is faster for $n > 36$
- We are mainly interested in $n = \{10, 11, 12, 16\}$
  - or multiples $\{20, 22, 24, 32\}$

Schoolbook vs. Karatsuba



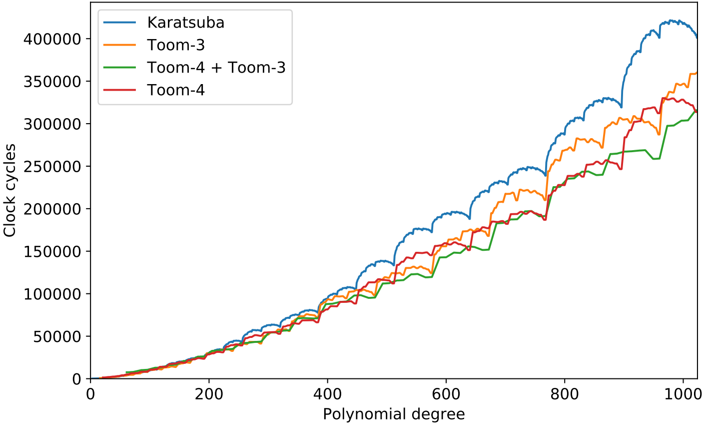
- Schoolbook is faster for $n \leq 16$
- Karatsuba is faster for $n > 36$
- We are mainly interested in $n = \{10, 11, 12, 16\}$
  - or multiples $\{20, 22, 24, 32\}$
- For $\{20, 22, 24, 32\}$ Karatsuba is faster

Karatsuba vs. Toom-4 vs. Toom-3



- Toom, multiple layers of Karatsuba
- Should be monotonic
  - Some schoolbooks are just not that optimized

Karatsuba vs. Toom-4 vs. Toom-3



- We are mainly interested in $n = \{256, 701, 743, 1024\}$
  - Ensure those 'make sense'

Speed records

scheme	params	impl	key gen	encaps	decaps
KINDI	$n = 256$ $q = 2^{14}$	ref	21 794k	28 176k	37 129k
		ours	**1 010k**	**1 365k**	**1 563k**
NTRU-HRSS	$n = 701$ $q = 2^{13}$	ref	205 156k	5 166k	15 067k
		ours	**161 790k**	**432k**	**863k**
NTRU-KEM	$n = 743$ $q = 2^{11}$	ref	59 815k	7 540k	14 229k
		ours	**5 663k**	**1 655k**	**1 904k**
SABER	$n = 256$ $q = 2^{13}$	ref	6 530k	8 684k	10 581k
		[1]	1 147k	1 444k	1 543k
		ours	**949k**	**1 232k**	**1 260k**
RLizard	$n = 1024$ $q = 2^{11}$	ref	26 423k	32 156k	53 181k
		ours	**537k**	**1 358k**	**1 740k**

[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.

Results & Conclusions

- Runtime dominated by polynomial multiplication
  - After optimizing: SHA2/SHA3/SHAKE is significant ($\approx$ 50%)
  - Optimized implementations exist

Results & Conclusions

- Runtime dominated by polynomial multiplication
  - After optimizing: SHA2/SHA3/SHAKE is significant ($\approx$ 50%)
  - Optimized implementations exist
- Fastest PQC implementations on the Cortex-M4
  - More than 2x outperform R5ND_1PKEb and R5ND_3PKEb
- Scripts easily apply to parameter changes in Round 2

Paper: https://eprint.iacr.org/2018/1018 (in submission)

Software: https://github.com/mupq/polymul-z2mx-m4

PQM4: https://github.com/mupq/pqm4

All code available as public domain where possible