National Taiwan University

Case Studies on Implementing Number–Theoretic Transforms with Armv7-M, Armv7E-M, and Armv8-A

Vincent Hwang

May 26, 2022

Outline

Scope

	Cortex-M3	Cortex-M4	Cortex-A72
Dilithium			✓
Kyber			✓
NTRU		✓	
NTRU Prime		✓	
Saber	✓	✓	✓

Dilithium

	Cortex-M3	Cortex-M4	Cortex-A72
dilithium2	[GKS21]	[AHKS22]	✓
dilithium3	[GKS21]	[AHKS22]	✓
dilithium5	[GKS21]	[AHKS22]	✓

Kyber

	Cortex-M3	Cortex-M4	Cortex-A72
kyber512	[GKS21]	[AHKS22]	✓
kyber768	[GKS21]	[AHKS22]	✓
kyber1024	[GKS21]	[AHKS22]	✓

NTRU

	Cortex-M3	Cortex-M4	Cortex-A72
`ntruhps2048509`	-	[IKPC22]	[NG21]
`ntruhps2048677`	-	✓	[NG21]
`ntruhrss701`	-	✓	[NG21]
`ntruhps4096821`	-	✓	[NG21]

NTRU Prime

	Cortex-M3	Cortex-M4	Cortex-A72
`ntrup653`	-	[Che21]	-
`ntrup761`	-	[Che21]	-
`ntrup857`	-	✓	-
`ntrup953`	-	-	-
`ntrup1013`	-	✓	-
`ntrup1277`	-	✓	-

Saber

	Cortex-M3	Cortex-M4	Cortex-A72
`lightsaber`	✓	✓	✓
`saber`	✓	✓	✓
`firesaber`	✓	✓	✓

Cortex-M3 and Cortex-M4

Cortex-M3 and Cortex-M4

- Cortex-M3:
  - `nucleo-f207zg`
  - Armv7-M
  - `mul`, `mla`
  - Early-terminating: `smull`, `smlal`, `umull`, `umlal`
- Cortex-M4:
  - `stm32f4discovery`
  - Armv7E-M
  - Constant time: `smull`, `smlal`, `umull`, `umlal`
  - DSP extension
    - `smul{b, t}{b, t}`
    - `smla{b, t}{b, t}`
    - `smu{a, s}d{, x}`
    - `sml{a, s}d{, x}`

NTRU and NTRU Prime on Cortex-M4

NTRU and NTRU Prime on Cortex-M4

- NTRU
    1. [KRS19]: Toom–Cook, 2019
    2. [CHK$^+$21]: NTT, 2021
    3. [IKPC22]: Toeplitz matrix, 2022
    4. This thesis: NTT, 2022
- NTRU Prime
    1. Toom–Cook, 2019
    2. [ACC$^+$21]: NTT, 2021
    3. [Che21]: NTT, 2021
    4. This thesis: NTT, 2022

Target Operations

- “big by small” polynomial multiplications
- Small: $\mathbb{Z}_3$

NTRU, ring $\mathbb{Z}_q[x]/\langle x^n - 1\rangle$		
Parameter	q	n
`ntruhps2048509`	2048	509
`ntruhps2048677`	2048	677
`ntruhrss701`	8192	701
`ntruhps4096821`	4096	821
NTRU Prime, field $\mathbb{Z}_q[x]/\langle x^p - x - 1\rangle$		
Parameter	q	p
`ntrup653`	4621	653
`ntrup761`	4591	761
`ntrup857`	5167	857
`ntrup953`	6343	953
`ntrup1013`	7177	1013
`ntrup1277`	7879	1277

Overview of Approaches

- Compute the results in $\mathbb{Z}[x]$:
  - Compute in $\mathbb{Z}_{q'}[x] \Big/ \left\langle x^{n'} - 1 \right\rangle$ for suitable n', q'
  - 32-bit arithmetic
- Factor $n' = v q_0 q_1$ where $q_0 = 2^{k_0}, q_1 = 3^{k_1}$.
- Good–Thomas FFT
  - $k_1 = 0, v \perp q_0$
    - $\mathbb{Z}_{q'}[x] \Big/ \left\langle x^{n'} - 1 \right\rangle \cong \mathbb{Z}_{q'}[x] / \langle (x^{(0)})^{q_0} - 1 \rangle \otimes \mathbb{Z}_{q'}[x] / \langle (x^{(1)})^{v} - 1 \rangle$
  - $k_1 > 0$
    - Ensure $k_1 = 1, 2$ for compact code size
    - Let $\mathcal{R}' = \mathbb{Z}_{q'}[x] / \langle x^v - x^{(0)} x^{(1)} \rangle$
    - $\mathbb{Z}_{q'}[x] \Big/ \left\langle x^{n'} - 1 \right\rangle \cong \mathcal{R}'[x] / \langle (x^{(0)})^{q_0} - 1 \rangle \otimes \mathcal{R}'[x] / \langle (x^{(1)})^{q_1} - 1 \rangle$
    - Vector–radix FFT
    - Dedicated radix-$(2, 3)$ butterflies
  - Otherwise, choose v, q_0, q_1 again

Approaches – Size-1536 Convolutions

`ntruhps2048677`, `ntruhrss701`, and `ntrulpr761/sntrup761`

- 32-bit
- Dedicated radix-$(2, 3)$ butterflies
- Size-1536 convolutions
- Cooley–Tukey, 2D Good–Thomas, and vector–radix FFTs
- Prior works
  - NTRU:
    1. Size-1536 convolution with 32-bit Good–Thomas, radix-2 splits only
    2. Toeplitz matrix, 16-bit, require $\frac{\mathbb{Z}_{2^k}[x]}{\langle x^n - 1 \rangle}$
  - NTRU Prime:
    1. Size-1536 convolution with 32-bit Good–Thomas, radix-2 splits only
    2. Size-1530 convolution with 16-bit Rader's, require $\mathbb{Z}_{4591}$, $153 | \mathbf{O}(4591)$
- Size-3 is worth implementing if combined with Good–Thomas FFT

Approaches – Size-1440 Convolutions

`ntruhps2048677`, `ntruhrss701`, and `ntrulpr653/sntrup653`

- 32-bit
- Dedicated radix-$(2, 3)$ butterflies
- Size-1440 convolutions
- Cooley–Tukey, 2D Good–Thomas, and vector–radix FFTs
- Prior works
  - NTRU:
    1. Size-1536 convolution with 32-bit Good–Thomas, radix-2 splits only
    2. Toeplitz matrix, require $\frac{\mathbb{Z}_{2^k}[x]}{\langle x^n - 1 \rangle}$
  - NTRU Prime:
    1. Size-1536 convolution with 32-bit Good–Thomas, radix-2 splits only
    2. Size-1320 convolution with 16-bit Rader's, require $\mathbb{Z}_{4621}$, $132 | \mathbf{O}(4621)$
- Size-1440 is clearly faster than size-1536

Approaches – Size-1728 Convolutions

`ntruhps4096821`, and `ntrulpr857/sntrup857`

- 32-bit
- Dedicated radix-$(2, 3)$ butterflies
- Size-1728 convolution
- Cooley–Tukey FFT, 2D Good–Thomas FFT, and vector–radix FFT
- Prior works
  - NTRU
    1. Size-1728 convolution with mixed-radix NTTs
    2. Toeplitz matrix, require $\frac{\mathbb{Z}_{2^k}[x]}{\langle x^n - 1 \rangle}$
  - NTRU Prime
    1. Size-1728 convolution with mixed-radix
    2. Size-1722 convolution with Rader's, require $\mathbb{Z}_{4621}$
- Good–Thomas with 27×64
  - Large code size
  - Solution: 9×64 is acceptable

Saber on Cortex-M4

Saber on Cortex-M3 and Cortex-M4

- Cortex-M4
    1. [KRS19]: Toom–Cook, 2019
    2. [IKPC20]: Toeplitz matrix, 2020
    3. [CHK+21]: NTT, 2021
    4. [ACC+22]: NTT, 2022 (selected)
    5. [BMK+22]: striding Toom–Cook, 2022
- Cortex-M3
    1. `pqm3`[1]: Toom–Cook, 2020
    2. [ACC+22]: NTT, 2022 (16-bit approach selected)

[1] `https://github.com/mupq/pqm3`

Target Operations

- "big by small" polynomial operations
- Small: $\left\{-\frac{\mu}{2}, \ldots, \frac{\mu}{2}\right\}$
- Matrix $A \in \left(\mathbb{Z}_{8192}[x] / \left\langle x^{256}+1\right\rangle\right)^{l \times l}$
- Vectors $b, s, s' \in \left(\mathbb{Z}_{8192}[x] / \left\langle x^{256}+1\right\rangle\right)^{l \times 1}$

	l	μ
`lightsaber`	2	10
`saber`	3	8
`firesaber`	4	6

Overview of Approaches

Cortex-M4

- Unmasked (“big by small”)
  - 32-bit, 16-bit
  - NTT: homomorphism
  - 4 strategies for time-memory tradeoffs
  - Speed-optimized
  - Stack-optimized: composite modulus
  - Prior works:
    - Karatsuba: stack-optimized
    - Toom–Cook: speed-optimized
    - Toeplitz matrix: speed-optimized
    - NTT: speed-optimized
- Masked (arithmetically, “big by big”)
  - 32-bit, 16-bit
  - NTT: homomorphism, 4 strategies, speed-optimized, stack-optimized
  - Prior work: Toom–Cook
  - NTT is faster since A is public

Time-Memory Tradeoffs

- 32-bit NTT: $\mathbb{Z}_{8192} \hookrightarrow \mathbb{Z}_{3329 \cdot 7681}$
- Increase of precision $\Longrightarrow$ increase of memory usage
- Matrix A, vectors b, s, s'
- Key generation: $A^T s$
- Encryption: $As', b^T s'$
- On-the-fly generation of A from `shake`
- Strategy A: $As' = \mathsf{NTT}^{-1}\left(\mathsf{NTT}(A) \cdot \underline{\mathsf{NTT}(s')}\right)$
- Strategy B: $A_{i,j} s'_j = \mathsf{NTT}^{-1}\left(\mathsf{NTT}(A_{i,j}) \cdot \underline{\mathsf{NTT}(s'_j)}\right)$
- Strategy C: $As' = \mathsf{NTT}^{-1}\left(\mathsf{NTT}(A) \cdot \mathsf{NTT}(s')\right)$
- Strategy D: $A_{i,j} s'_j = \mathsf{NTT}^{-1}\left(\mathsf{NTT}(A_{i,j}) \cdot \mathsf{NTT}(s'_j)\right)$
- Key generation: A, B, D
- Encryption: A, C, D

Stack Optimization

- 32-bit NTT: $\mathbb{Z}_{8192} \hookrightarrow \mathbb{Z}_{3329 \cdot 7681}$
- 16-bit NTTs: friendly for memory optimization
- Cortex-M4
  - Draft with $\mathbb{Z}_{3329}$ and $\mathbb{Z}_{7681}$ first
  - One 32-bit NTTand $(\mathrm{mod} 3329, \mathrm{mod} 7681)$ is much faster than two 16-bit NTTs
  - Replace $\mathbb{Z}_{3329}$ and $\mathbb{Z}_{7681}$ with $\mathbb{Z}_{3329 \cdot 7681}$ if both in memory, transform if needed
- Cortex-M3
  - Compute in $\mathbb{Z}_{3329}$ and $\mathbb{Z}_{7681}$
  - One variable-time 32-bit NTTand $(\mathrm{mod} 3329, \mathrm{mod} 7681)$ is *neglectably* faster than two constant-time 16-bit NTTs
  - Neglectably faster for NTT$(A_{i,j})$

Cortex-A72

Cortex-A72

- Raspberry pi 4
- Armv8.0-A
- 8 pipelines:
  - `F0`: logical, additions, subtractions, multiplications
  - `F1`: logical, additions, subtractions, shift operations
  - `I0` and `I1`: logical, addistions, subtractions
  - `M`: multiplications, divisions, shift operations
  - `B`: branches
  - `L`: load operations
  - `S`: store operations
- In-order frontend + out-of-order backend
  - Reduce the workload of `F0`
  - Constraints on decoding instructions
  - Instruction interleaving

Dilithium, Kyber, and Saber on Cortex-A72

Dilithium, Kyber, and Saber on Cortex-A72

1. [NG21]: Toom–Cook for NTRU and Saber; NTTs for Kyber and Saber
2. [BHK^{+}22]: NTTs for Dilithium, Kyber, and Saber

Target Operations

- Matrix $A \in (\mathbb{Z}_q[x]/\langle x^{256}+1\rangle)^{k\times l}$
- Vector $s, s' \in (\mathbb{Z}_q[x]/\langle x^{256}+1\rangle)^{l\times 1}$

	q	k	l	As	ss'
`dilithium2`	8380417	4	4	✓	
`dilithium3`	8380417	6	5	✓	
`dilithium5`	8380417	8	7	✓	
`kyber512`	3329	2	2	✓	✓
`kyber768`	3329	3	3	✓	✓
`kyber1024`	3329	4	4	✓	✓
`lightsaber`	8192	2	2	✓	✓
`saber`	8192	3	3	✓	✓
`firesaber`	8192	4	4	✓	✓

Overview of Optimizations

- Modular reductions and multiplications
  - Barrett multiplication
  - Correspondences between Barrett-type and Montgomery-type
  - Improve Barrett reduction, Montgomery reduction, and Montgomery multiplication.
- Instruction scheduling for Cooley–Tukey and Gentleman–Sande FFTs
- Asymmetric multiplicaton

Instruction Scheduling

- Radix-2 FFT for $R[x] \Big/ \Big\langle x^{2^k} - 1 \Big\rangle$: $\mathrm{rev}_{(2:k)}$
- Proposed policy
  - Schedule odd indices first
- Generalization
  - Height-based scheduling (compiler optimizations)
  - Closed form for FFT: $\mathrm{rev}^{\mathrm{rev}}_{(2:k)} = i \mapsto \mathrm{rev}_{(2:k)}(2^k - 1 - i)$
    - FFT: $\mathrm{rev}_{(2:k)}$
    - Dependencies: $i \mapsto 2^k - 1 - i$
    - $\mathrm{rev}_{(2:k)} \circ (i \mapsto 2^k - 1 - i) = (i \mapsto 2^k - 1 - i) \circ \mathrm{rev}_{(2:k)}$

Asymmetric Multiplication

- Polynomials $\boldsymbol{a}(x) = a_0 + a_1 x$, $\boldsymbol{b}(x) = b_0 + b_1 x$
- Compute $\boldsymbol{c}(x) = c_0 + c_1 x = \boldsymbol{a}(x)\boldsymbol{b}(x) \in R[x]/\langle x^2 - \psi \rangle$ essentially in $R[x]/\langle x^2 - 1 \rangle$ without requiring $\sqrt{\psi} \in R$
- $$\begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} a_0 b_0 + \psi(a_1 b_1) \\ a_0 b_1 + a_1 b_0 \end{pmatrix} = \begin{pmatrix} a_0 b_0 + a_1(\psi b_1) \\ a_0 b_1 + a_1 b_0 \end{pmatrix}$$
- $c_i = \sum_{j=0}^{i} a_j b_{i-j} + \sum_{j=i+1}^{n-1} a_j \left(\psi b_{n+i-j}\right)$
- First implemented in Ed25519 for $\mathbb{Z}_{2^{255}-19}$ as $R[x]/\langle x^5 - 19 \rangle$
- Matrix-to-vector multiplication
  - Explain how to save multiplications (this was not known previously)
  - Complete NTT: $\mathsf{NTT}^{-1}(\mathsf{NTT}(A) \cdot \underline{\mathsf{NTT}(s')})$
  - Incomplete NTT: $\mathsf{NTT}^{-1}(\texttt{asymmetric\_mul}(\mathsf{NTT}(A), \underline{\texttt{NTT\_heavy}(s')}))$
- Kyber: incomplete 7-layer radix-2 NTT for $R[x]/\langle x^{256} + 1 \rangle$
- Saber: choose incomplete 6-layer radix-2 NTT for $R[x]/\langle x^{256} + 1 \rangle$

More Optimizations and Future Works

Improving Näive Butterflies

- $(a_0, a_1) \mapsto (a_0 + \psi a_1, a_0 - \psi a_1)$
- $(a_0, a_1, a_2) \mapsto \begin{pmatrix} a_0 + a_1\psi + a_2\psi^2 \\ a_0 + a_1\psi\omega_3 + a_2\psi^2\omega_3^2 \\ a_0 + a_1\psi\omega_3^2 + a_2\psi^2\omega_3 \end{pmatrix}$
- $a_1\psi\omega_3^2 + a_2\psi^2\omega_3 = -\left(a_1\psi(1+\omega_3) + a_2\psi^2(1+\omega_3^2)\right)$
- $a_0 + a_1\psi\omega_3^2 + a_2\psi^2\omega_3 = a_0 - \left((a_1\psi + a_2\psi^2) + (a_1\psi\omega_3 + a_2\psi^2\omega_3^2)\right)$
- Compatible with vector–radix FFT
- For large prime r, pair with Rader's and Winograd's

CT–GS butterflies

- CT for $R[x]/\langle x^8 - 1\rangle$:
  $(a_0, \ldots, a_7) \mapsto (a'_0, \ldots, a'_7) \mapsto (a''_0, \ldots, a''_7) \mapsto (a'''_0, \ldots, a'''_7)$
- $(a'_5, a'_7) \mapsto (\omega_8 a''_5, \omega_8^3 a''_7) = (\omega_8 a'_5 + \omega_8^3 a'_7, \omega_8^3 a'_5 + \omega_8 a'_7)$
- Save one modular reduction
- Why "CT–GS": one can also derive it from GS for $R[x]/\langle x^8 - 1\rangle$
- Generalization to $R[x]/\langle x^8 - \psi^8\rangle$ where $\psi^8 \neq 1$?
- Generalizations to non-radix-2 and their relation to improved näive butterflies?

Vectorization for Non-Radix-2 NTTs

- Future work
- Good–Thomas FFT
  - Require $n = vq_0q_1$ where $q_0 \perp q_1$
  - Let $\mathcal{R}' = R[x] \Big/ \left\langle x^v - x^{(0)}x^{(1)} \right\rangle$
  - $R[x]/\langle x^n - 1\rangle \cong \mathcal{R}'[x^{(0)}] \Big/ \left\langle \left(x^{(0)}\right)^{q_0} - 1 \right\rangle \otimes \mathcal{R}'[x^{(1)}] \Big/ \left\langle \left(x^{(1)}\right)^{q_1} - 1 \right\rangle$
- Truncated Schönhage
  - Stick to the original coefficient ring
  - Let $\mathcal{R}' = R[x]/\langle x^v - y\rangle \hookrightarrow \mathcal{R}'' = R[x]/\left\langle x^{2v} + 1 \right\rangle$, $\omega = x^{\frac{4v}{2^{\lceil \log_2 q_0 q_1 \rceil}}}$
  - $R[x]/\langle \prod_{i=0}^{q_0q_1-1}(x^v - \omega^{\mathrm{rev}(2:i)})\rangle \hookrightarrow \mathcal{R}''[y]/\langle \prod_{i=0}^{q_0q_1-1}(y - \omega^{\mathrm{rev}(2:i)})\rangle \cong \prod_{i=0}^{q_0q_1-1} \mathcal{R}''[y]/\langle y - \omega^{\mathrm{rev}(2:i)}\rangle$
- What if $\exists \omega_{q_0} \in R$?
  - Good–Thomas, Schönhage, and vectorization-friendly?

Results

NTRU and NTRU Prime on Cortex-M4

Table: Comparisons of `polymul` in NTRU on Cortex-M4.

NTRU (q, n)	Convolution	This work	[CHK+21]	[IKPC22]
$(677, 2048)$	Size-677	$-/-$	$-/-$	144k/$-$
	Size-1440	140k/143k	$-/-$	$-/-$
	Size-1536	147k/149k	156k/$-$	$-/-$
$(701, 8192)$	Size-701	$-/-$	$-/-$	144k/$-$
	Size-1440	141k/143k	$-/-$	$-/-$
	Size-1536	148k/150k	156k/$-$	$-/-$
$(821, 4096)$	Size-821	$-/-$	$-/-$	193k/$-$
	Size-1728	178k/182k	199k/$-$	$-/-$

polymul

Table: Comparisons of `polymul` in NTRU Prime on Cortex-M4.

NTRU Prime (p, q)	Convolution	This work	[ACC$^+$21]	[Che21]
$(653, 4621)$	Size-1320	$-/-$	$-/-$	120k$/-$
	Size-1440	142k$/147$k	$-/-$	$-/-$
$(761, 4591)$	Size-1530	$-/-$	152k$/-$	142k$/-$
	Size-1536	151k$/153$k	159k$/-$	$-/-$
	Size-1620	$-/-$	185k$/-$	$-/-$
$(857, 5167)$	Size-1722	$-/-$	$-/-$	203k$/-$
	Size-1728	182k$/186$k	$-/-$	$-/-$
$(1013, 7177)$	Size-2048	224k$/227$k	$-/-$	$-/-$
$(1277, 7879)$	Size-2560	285k$/290$k	$-/-$	$-/-$

NTRU on Cortex-M4

Table: Performance of NTRU on Cortex-M4.

Parameter		[CHK+21]	[IKPC22]	[Li21]	This thesis
ntruhps2048677	**K**	143 725k	142 378k	4 625k	3 906k
	E	821k	816k	820k	523k
	D	818k	729k	812k	714k
ntruhrss701	**K**	153 403k	153 479k	4 233k	3 816k
	E	377k	369k	376k	359k
	D	871k	787k	868k	774k
ntruhps4096821	**K**	207 495k	212 377k	6 116k	5 208k
	E	1 027k	1 026k	1 027k	651k
	D	1 030k	914k	1 031k	902k

NTRU Prime on Cortex-M4 I

Table: Performance of NTRU Prime on Cortex-M4.

Parameter		[ACC$^+$21]	[Che21]	This thesis
ntrulpr653	**K**	-	678k	667k
	E	-	1 158k	1 127k
	D	-	1 233k	1 226k
sntrup653	**K**	-	6 715k	6 673k
	E	-	632k	619k
	D	-	487k	522k
ntrulpr761	**K**	731k	727k	710k
	E	1 102k	1 312k	1 266k
	D	1 200k	1 394k	1 365k
sntrup761	**K**	10 778k	7 951k	7 937k
	E	694k	684k	666k
	D	572k	538k	563k

NTRU Prime on Cortex-M4 II

ntrulpr857	K	-	-	882k
	E	-	-	1 460k
	D	-	-	1 588k
sntrup857	K	-	-	10 189k
	E	-	-	809k
	D	-	-	679k
ntrulpr1013	K	-	-	1 059k
	E	-	-	1 742k
	D	-	-	1 899k
sntrup1013	K	-	-	13 841k
	E	-	-	981k
	D	-	-	827k
ntrulpr1277	K	-	-	1 360k
	E	-	-	2 207k
	D	-	-	2 401k

NTRU Prime on Cortex-M4 III

sntrup1277	**K**	-	-	22 756k
	E	-	-	1 253k
	D	-	-	1 058k

NTRU Prime on Cortex-M4 IV

Saber on Cortex-M4

polymul

Table: The operation counts and performance of `polymul` in Saber on Cortex-M4.

Operation	Performance		Operation Count	
	24 MHz	168 MHz	speed-opt	stack-opt
32-bit NTT	5 855	6 108	2	1
16-bit NTT$(\mathrm{mod}\,7681)$	4 918	5 171	0	1
16-bit NTT$(\mathrm{mod}\,3329)$	4 470	4 705	0	1
32-bit `base_mul`	4 186	4 304	1	0
32-bit to 16-bit	1 181	1 263	0	1
16-bit `base_mul`	2 966	3 049	0	1
32×16-bit `base_mul`			0	1
CRT	2 438	2 515	0	1
32-bit NTT^{-1}	7 315	7 647	1	1
Overall performance				
24 MHz			23 077	32 557
168 MHz			23 958	33 842

MatrixVectorMul

Table: Performance of `MatrixVectorMul` in Saber on Cortex-M4.

	lightsaber	saber	firesaber
`MatrixVectorMul` (Enc, A)	67 624	133 587	221 006
	70 172	138 386	228 741
`MatrixVectorMul` (Enc, B)	81 844	176 277	306 387
	85 109	183 335	318 609
`MatrixVectorMul` (Enc, C)	79 225	168 417	290 681
	82 177	174 556	301 200
`MatrixVectorMul` (Enc, D)	131 373	296 370	527 770
	136 671	308 516	549 413

InnerProd

Table: Performance of InnerProd in Saber on Cortex-M4.

	lightsaber	saber	firesaber
InnerProd (Enc, A)	28 009	38 736	49 456
	29 046	40 156	51 165
InnerProd (Dec, A)	39 621	56 172	72 708
	41 119	58 311	75 404
InnerProd (Dec, B)	46 728	70 397	94 049
	48 591	73 288	97 855
InnerProd (Dec, C)	39 630	56 170	72 706
	41 132	58 301	75 354
InnerProd (Dec, D)	65 698	98 847	131 993
	68 389	102 920	137 372

Saber on Cortex-M4 I

Table: Performance of Saber on Cortex-M4.

		firesaber	
Implementation		Cycle	Stack
This thesis (speed, A)	**K**	**989k**	7 668
	E	**1 199k**	8 340
	D	**1 144k**	8 348
[CHK+21]	**K**	1 008k	37 116
	E	1 255k	40 484
	D	1 227k	41 964
This thesis (stack, D)	**K**	1 326k	**4 300**
	E	1 624k	**3 316**
	D	1 605k	**3 324**

Saber on Cortex-M4 II

This thesis (stack, D)	**K**	1 326k	**4 300**
	E	1 624k	**3 316**
	D	1 605k	**3 324**
[IKPC20]	**K**	1 319k	20 144
	E	1 621k	22 992
	D	1 649k	24 472
[BMK+22]	**K**	1 350k	4 280
	E	1 654k	4 792
	D	1 674k	5 304
[MKV20] (speed)	**K**	1 340k	26 448
	E	1 642k	29 228
	D	1 679k	30 768
[MKV20] (stack)	**K**	2 046k	5 116
	E	2 538k	3 668
	D	2 740k	3 684

Saber on Cortex-M4 III

Saber on Cortex-M3

`polymul`

Table: Performance of `polymul` in Saber on Cortex-M3.

	30 MHz	120 MHz
`polymul`	68 776	69 334
NTT($\mathrm{mod}\,7681$)	8 689	8 762
NTT($\mathrm{mod}\,3329$)	8 033	8 112
`base_mul`	5 987	6 036
NTT^{-1}	9 554	9 683
CRT	4 639	4 652

Table: Performance of MatrixVectorMul in Saber on Cortex-M3.

	lightsaber	saber	firesaber
MatrixVectorMul (Enc, A)	198 705	390 340	635 269
	200 205	393 051	639 462
MatrixVectorMul (Enc, B)	245 164	529 525	904 660
	247 198	534 017	912 688
MatrixVectorMul (Enc, C)	231 951	490 113	832 063
	233 609	493 390	837 424
MatrixVectorMul (Enc, D)	278 404	629 242	1 097 587
	280 644	634 312	1 106 774

Table: Performance of `InnerProd` in Saber on Cortex-M3.

	lightsaber	saber	firesaber
InnerProd (Enc, A)	82 722	113 283	143 856
	83 362	114 087	144 810
InnerProd (Dec, A)	115 966	163 177	210 355
	116 802	164 269	211 701
InnerProd (Dec, B)	139 223	209 686	280 127
	140 360	211 428	282 419
InnerProd (Dec, C)	115 979	163 142	210 352
	116 823	164 231	211 692
InnerProd (Dec, D)	139 207	209 641	280 071
	140 324	211 322	282 335

Saber on Cortex-M3

Table: Performance of Saber on Cortex-M3.

		firesaber	
Implementation		Cycle	Stack
[ACC+22] (speed, A)	**K**	**1 503k**	7 804
	E	**1 817k**	8 484
	D	**1 885k**	8 484
[ACC+22] (stack, D)	**K**	2 029k	**4 436**
	E	2 492k	**3 460**
	D	2 559k	**3 460**
pqm3 (Toom–Cook)	**K**	2 171k	20 116
	E	2 688k	22 964
	D	2 933k	24 444

Dilithium, Kyber, and Saber on Cortex-A72

NTT-related

$(\cdot)$ is `dim` $\times$ `base_mul`.

Table: Performance of `NTT`, `NTT_heavy`, `base_mul`, and NTT^{-1} in `kyber768`, `saber`, and `dilithium3` on Cortex-A72.

	NTT	NTT_heavy	$(\cdot)$	NTT^{-1}	CRT
kyber768 [BHK+22]	1 200	1 434	952	1 338	–
kyber768 [NG21]	1 473	–	3 040	1 661	–
saber 32-bit [BHK+22]	1 529	2 031	2 689	1 896	–
saber 16-bit [NG21]	1 991	–	1 500	1 893	813
dilithium3 [BHK+22]	2 241	–	1 378	2 821	–
dilithium3 (ref)	9 302	–	11 625	11 633	–

`MatrixVectorMul` and `InnerProd` in Kyber

Table: Performance of `MatrixVectorMul` and `InnerProd` in Kyber on Cortex-A72.

		`MV`	`IP`(Enc)	`IP` (Dec)
`kyber512`	[BHK+22]	6 849	2 000	4 844
	[NG21]	10 700	–	7 100
`kyber768`	[BHK+22]	11 077	2 242	6 518
	[NG21]	19 300	–	9 900
`kyber1024`	[BHK+22]	16 338	2 758	8 487
	[NG21]	–	–	–

Table: Performance of `MatrixVectorMul` and `InnerProd` in Dilithium on Cortex-A72.

		`MV`	`IP`(Enc)	`IP` (Dec)
`dilithium2`	[BHK+22]	26 268	–	–
	(ref)	135 182	–	–
`dilithium3`	[BHK+22]	38 107	–	–
	(ref)	215 503	–	–
`dilithium5`	[BHK+22]	54 759	–	–
	(ref)	334 865	–	–

`MatrixVectorMul` and `InnerProd` in Saber

Table: Performance of `MatrixVectorMul` and `InnerProd` in Saber on Cortex-A72.

		MV	IP(Enc)	IP (Dec)
lightsaber	[BHK+22]	18 149	7 038	11 113
	[NG21] (NTT)	37 000	–	22 500
	[NG21] (TC)	40 200	–	18 100
saber	[BHK+22]	35 730	9 284	15 452
	[NG21] (NTT)	71 300	–	31 500
	[NG21] (TC)	81 000	–	25 000
firesaber	[BHK+22]	56 109	11 783	20 112
	[NG21] (NTT)	–	–	–
	[NG21] (TC)	–	–	–

Dilithium, Kyber, and Saber I

Table: Performance of Dilithium, Kyber, and Saber on Cortex-A72. We benchmark the fastest implementations by [NG21] in SUPERCOP.

		K	E	D
`kyber512`	[BHK+22]	62 459	80 710	76 443
	[NG21]	67 903	88 906	87 563
`kyber768`	[BHK+22]	99 201	127 453	120 665
	[NG21]	110 784	141 312	138 984
`kyber1024`	[BHK+22]	156 694	192 280	184 161
	[NG21]	176 809	215 665	214 076

Dilithium, Kyber, and Saber II

lightsaber	[BHK+22]	64 181	87 272	92 813
	[NG21]	83 960	118 583	136 203
saber	[BHK+22]	109 192	140 103	147 925
	[NG21]	158 757	206 337	226 304
firesaber	[BHK+22]	175 104	211 382	222 317
	[NG21]	245 249	304 128	330 750
		K	**S**	**V**
dilithium2	[BHK+22]	269 724	649 230	272 824
	Ref	410 312	1 353 753	449 633
dilithium3	[BHK+22]	515 776	1 089 387	447 460
	Ref	743 166	2 308 598	728 866
dilithium5	[BHK+22]	782 752	1 436 988	764 886
	Ref	1 151 504	2 903 604	1 198 723

Dilithium, Kyber, and Saber III

Thanks for listening

Reference I

[ACC+21] Erdem Alkim, Dean Yun-Li Cheng, Chi-Ming Marvin Chung, Hülya Evkan, Leo Wei-Lun Huang, Vincent Hwang, Ching-Lin Trista Li, Ruben Niederhagen, Cheng-Jhih Shih, Julian Wälde, and Bo-Yin Yang, *Polynomial Multiplication in NTRU Prime Comparison of Optimization Strategies on Cortex-M4*, IACR Transactions on Cryptographic Hardware and Embedded Systems **2021** (2021), no. 1, 217–238, `https://tches.iacr.org/index.php/TCHES/article/view/8733`.

[ACC+22] Amin Abdulrahman, Jiun-Peng Chen, Yu-Jia Chen, Vincent Hwang, Matthias J. Kannwischer, and Bo-Yin Yang, *Multi-moduli NTTs for Saber on Cortex-M3 and Cortex-M4*, IACR Transactions on Cryptographic Hardware and Embedded Systems **2022** (2022), no. 1, 127–151, `https://tches.iacr.org/index.php/TCHES/article/view/9292`.

Reference II

[AHKS22] Amin Abdulrahman, Vincent Hwang, Matthias J. Kannwischer, and Dann Sprenkels, *Faster Kyber and Dilithium on the Cortex-M4*, To appear at ACNS 2022, available as `https://eprint.iacr.org/2022/112`.

[BHK+22] Hanno Becker, Vincent Hwang, Matthias J. Kannwischer, Bo-Yin Yang, and Shang-Yi Yang, *Neon NTT: Faster Dilithium, Kyber, and Saber on Cortex-A72 and Apple M1*, IACR Transactions on Cryptographic Hardware and Embedded Systems **2022** (2022), no. 1, 221–244, `https://tches.iacr.org/index.php/TCHES/article/view/9295`.

Reference III

[BMK$^+$22] Hanno Becker, Jose Maria Bermudo Mera, Angshuman Karmakar, Joseph Yiu, and Ingrid Verbauwhedeg, *Polynomial multiplication on embedded vector architectures*, IACR Transactions on Cryptographic Hardware and Embedded Systems **2022** (2022), no. 1, 482–505, https://tches.iacr.org/index.php/TCHES/article/view/9305.

[Che21] Yun-Li Cheng, *Number Theoretic Transform for Polynomial Multiplication in Lattice-based Cryptography on ARM Processors*, Master's thesis, 2021, https://github.com/dean3154/ntrup_m4.

Reference IV

[CHK+21] Chi-Ming Marvin Chung, Vincent Hwang, Matthias J. Kannwischer, Gregor Seiler, Cheng-Jhih Shih, and Bo-Yin Yang, *NTT Multiplication for NTT-unfriendly Rings New Speed Records for Saber and NTRU on Cortex-M4 and AVX2*, IACR Transactions on Cryptographic Hardware and Embedded Systems **2021** (2021), no. 2, 159–188, `https://tches.iacr.org/index.php/TCHES/article/view/8791`.

[GKS21] Denisa O. C. Greconici, Matthias J. Kannwischer, and Daan Sprenkels, *Compact Dilithium Implementations on Cortex-M3 and Cortex-M4*, IACR Transactions on Cryptographic Hardware and Embedded Systems **2021** (2021), no. 1, 1–24, `https://tches.iacr.org/index.php/TCHES/article/view/8725`.

Reference V

[IKPC20] Írem Keskinkurt Paksoy and Murat Cenk, *TMVP-based Multiplication for Polynomial Quotient Rings and Application to Saber on ARM Cortex-M4*, Cryptology ePrint Archive (2020), `https://eprint.iacr.org/2020/1302`.

[IKPC22] ______, *Faster NTRU on ARM Cortex-M4 with TMVP-based multiplication*, `https://ia.cr/2022/300`.

[KRS19] Matthias J Kannwischer, Joost Rijneveld, and Peter Schwabe, *Faster Multiplication in $\mathbb{Z}_{2^m}[x]$ on Cortex-M4 to Speed up NIST PQC Candidates*, International Conference on Applied Cryptography and Network Security, Springer, 2019, pp. 281–301.

[Li21] Ching-Lin Li, *Implementation of Polynomial Modular Inversion in Lattice-based cryptography on ARM*, Master's thesis, 2021, `https://github.com/trista5658321/polyinv-m4`.

Reference VI

[MKV20] Jose Maria Bermudo Mera, Angshuman Karmakar, and Ingrid Verbauwhede, *Time-memory trade-off in Toom-Cook multiplication: an application to module-latticebased cryptography*, IACR Transactions on Cryptographic Hardware and Embedded Systems **2020** (2020), no. 2, 222–244, `https://tches.iacr.org/index.php/TCHES/article/view/8550`.

[NG21] Duc Tri Nguyen and Kris Gaj, *Optimized Software Implementations of CRYSTALS-Kyber, NTRU, and Saber Using NEON-Based Special Instructions of ARMv8*, 2021, Third PQC Standardization Conference.