Cortex-M4 optimizations for {R,M}LWE schemes

Erdem Alkım [1,2] **Yusuf Alper Bilgin** [3,4] Murat Cenk [4] François Gérard [5]

[1]Department of Computer Engineering, Ondokuz Mayıs University, Turkey

[2]Fraunhofer SIT, Darmstadt, Germany

[3]Aselsan Inc., Turkey

[4]Institute of Applied Mathematics, Middle East Technical University, Turkey

[5]Université libre de Bruxelles, Brussels, Belgium

y.alperbilgin@gmail.com

September, 2020

Overview

1. Introduction
2. Implementation Details
   - Optimizations for Speed
   - Optimizations for Stack Usage
   - Optimizations of Secret-key Size
3. Results
4. Conclusion

NIST Post-quantum Standardization Process

1^{st}, 2^{nd}, and 3^{rd} round finalists including alternate candidates

	Signatures			KEM/Encryption			Overall		
	1^{st}	2^{nd}	3^{rd}	1^{st}	2^{nd}	3^{rd}	1^{st}	2^{nd}	3^{rd}
Lattice-based	5	3	2	21	9	5	26	12	7
Code-based	2	0	0	17	7	3	19	7	3
Multi-variate	7	4	2	2	0	0	9	4	2
Symmetric-based	3	2	2				3	2	1
Other	2	0	0	5	1	1	7	1	1
Total	19	9	6	45	17	9	64	26	15

PQC Standardization Process: Third Round Candidate Announcement

Target {R,M}LWE Schemes

- Kyber
  - One of the third round finalists,
  - Based on MLWE problem,
  - Using 7-level NTT with $\mathbb{Z}_{3329}[X]/(X^{256}+1)$, and degree-2 schoolbook multiplications.

Target {R,M}LWE Schemes

- Kyber
  - One of the third round finalists,
  - Based on MLWE problem,
  - Using 7-level NTT with $\mathbb{Z}_{3329}[X]/(X^{256}+1)$, and degree-2 schoolbook multiplications.
- NewHope
  - Eliminated in the second round,
  - Based on RLWE problem,
  - Using 9-level or 10-level NTT with $\mathbb{Z}_{12289}[X]/(X^{512}+1)$ or $\mathbb{Z}_{12289}[X]/(X^{1024}+1)$.

Target {R,M}LWE Schemes

- KYBER
  - One of the third round finalists,
  - Based on MLWE problem,
  - Using 7-level NTT with $\mathbb{Z}_{3329}[X]/(X^{256}+1)$, and degree-2 schoolbook multiplications.
- NEWHOPE
  - Eliminated in the second round,
  - Based on RLWE problem,
  - Using 9-level or 10-level NTT with $\mathbb{Z}_{12289}[X]/(X^{512}+1)$ or $\mathbb{Z}_{12289}[X]/(X^{1024}+1)$.
- NEWHOPE-COMPACT [1]
  - Faster and smaller variant of NEWHOPE,
  - Based on RLWE problem,
  - Using 7-level NTT with $\mathbb{Z}_{3329}[X]/(X^{512}+1)$, $\mathbb{Z}_{3329}[X]/(X^{728}-X^{384}+1)$, $\mathbb{Z}_{3329}[X]/(X^{1024}+1)$, and degree 4, 6 or 8 schoolbook multiplications.

[1] E. Alkım, Y. A. Bilgin, M. Cenk, Compact and Simple RLWE Based Key Encapsulation Mechanism, Latincrypt2019

NewHope

Key Generation

Output: public key $pk = (\hat{b}', \rho)$
Output: secret key $sk = \hat{s}$

$seed \xleftarrow{\$} \{0, \cdots, 255\}^{32}$
$\rho, \sigma \leftarrow \text{SHAKE256}(64, seed)$
$\hat{a} \leftarrow \text{GenA}(\rho)$
$s \leftarrow \text{Sample}(\sigma, 0)$
$e \leftarrow \text{Sample}(\sigma, 1)$
$\hat{b} \leftarrow \hat{a} \circ \text{NTT}\ (s) + \text{NTT}\ (e)$
return $pk = (\hat{b}, \rho), sk = \hat{s}$

Decryption

Input: ciphertext $c = (\hat{u}, h)$
Input: secret key $sk = \hat{s}$
Output: message $\mu \in \{0, \cdots, 255\}^{32}$

$v' \leftarrow \text{Decompress}(h)$
return $\mu = \text{Decode}(v' - \text{NTT}^{-1}(\hat{u} \circ \hat{s}))$

Encryption

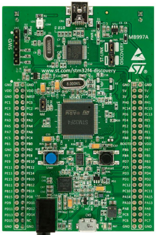
Input: public key $pk = (\hat{b}, \rho)$
Input: message μ encoded in $\mathcal{R}_q$
Input: seed $coin \in \{0, \cdots, 255\}^{32}$
Output: ciphertext $(\hat{u}', h)$

$\hat{a} \leftarrow \text{GenA}(\rho)$
$s' \leftarrow \text{Sample}(coin, 0)$
$e' \leftarrow \text{Sample}(coin, 1)$
$e'' \leftarrow \text{Sample}(coin, 2)$
$\hat{t} \leftarrow \text{NTT}\ (s')$
$\hat{u} \leftarrow \hat{a} \circ \hat{t} + \text{NTT}\ (e')$
$v' \leftarrow \text{NTT}^{-1}(\hat{b} \circ \hat{t}) + e'' + \mu$
return $c = (\hat{u}, \text{Compress}(v'))$

NewHope: Algorithm Specifications and Supporting Documentation

ARM Cortex-M4

- NIST recommended Cortex-M4 for PQC evaluation
- STM32F4DISCOVERY:
  - 32-bit, ARMv7E-M
  - Includes SIMD instructions
  - 1MB ROM, 192 KB RAM, 168 MHz
  - PQM4
  - 16 registers but only 14 avaliable



STMicroelectronics, STM32F4DISCOVERY

Previous optimizations of Kyber on Cortex-M4[1]

We also use them in our NewHope and NewHope-Compact implementations.

- Use signed representation
- Pack two coefficients into one register, utilize `uadd16` or `usub16` for parallel addition/subtraction
- All computations in Montgomery-domain
- Precompute twiddle factors - place them in Flash memory
- Enable link-time optimization (`flto`)

[1] L. Botros, M. Kannwisher, P. Schwabe, Memory-Efficient High-Speed Implementation of Kyber on Cortex-M4, Africacrypt2019

Montgomery Reduction

Proposed by Botros et. al.[1]
1: `smulbb` t, a, q^{-1}
2: `smulbb` t, t, q
3: `usub16` a, a, t

This work
1: `smulbb` $t, a, -q^{-1}$
2: `smlabb` a, t, q, a

- 3200 Montgomery reductions in
  $(\mathrm{NTT}^{-1}(\mathrm{NTT}\ (a) \circ \mathrm{NTT}\ (b)))$ where a and $b \in \mathbb{Z}_{3329}[X]/(X^{256}+1)$
- Double Montgomery reduction on a packed argument
  - ☐ 1 cycle faster than double Barrett reduction

[1] L. Botros, M. Kannwisher, P. Schwabe, Memory-Efficient High-Speed Implementation of Kyber on Cortex-M4, Africacrypt2019

More Aggressive Lazy Reduction

Lazy reductions after component-wise multiplication:

$$
\begin{aligned}
c[1] \leftarrow & (a[0] \cdot b[1]) \,\mathbf{mod}\, \boldsymbol{q} + (a[1] \cdot b[0]) \,\mathbf{mod}\, \boldsymbol{q} \\
c[1] \leftarrow & (a[0] \cdot b[1] \qquad\qquad + a[1] \cdot b[0]) \,\mathbf{mod}\, \boldsymbol{q}
\end{aligned}
$$

More Aggresive Lazy Reduction

Lazy reductions after component-wise multiplication:

$$c[1] \leftarrow (a[0] \cdot b[1]) \,\mathbf{mod}\, \boldsymbol{q} + (a[1] \cdot b[0]) \,\mathbf{mod}\, \boldsymbol{q}$$
$$c[1] \leftarrow (a[0] \cdot b[1] \qquad\qquad + a[1] \cdot b[0]) \,\mathbf{mod}\, \boldsymbol{q}$$

- We save:
  - 128 reductions for $\mathbb{Z}_{3329}[X]/(X^{256} + 1)$,
  - 1536 reductions for $\mathbb{Z}_{3329}[X]/(X^{512} + 1)$,
  - 3840 reductions for $\mathbb{Z}_{3457}[X]/(X^{768} - X^{384} + 1)$,
  - 7168 reductions for $\mathbb{Z}_{3329}[X]/(X^{1024} + 1)$,
- Skip the reductions after the multiplications in the first layer of NTT
  - Inputs are small, sampled from the centered binomial distribution.

Merging NTT Layers

- 8 registers out of 14 reserved for the coefficients
  - Perform 3 or 4 layers of the NTT at a time
  - 3+3+1 for KYBER
  - 4+3+2 or 4+3+3 for NEWHOPE
  - 3+4 for NEWHOPE-COMPACT

Stack Optimizations

NTT is already stack friendly (entirely in-place).

Stack Optimizations

NTT is already stack friendly (entirely in-place).

Previous optimizations for KYBER on Cortex-M4[1]:

- Inline comparision in CCA decapsulation,
- On-the-fly generation of matrix **A** in matrix-vector multiplication.

In this work, these are also implemented for NEWHOPE and NEWHOPE-COMPACT.

[1] L. Botros, M. Kannwisher, P. Schwabe, Memory-Efficient High-Speed Implementation of Kyber on Cortex-M4, Africacrypt2019

Stack Optimizations: KeyGen

On-the-fly error addition:

Instead of computing

$$\hat{b} \leftarrow \hat{a} \circ \mathsf{NTT}\,(s) + \mathsf{NTT}\,(e),$$

we compute

$$\hat{b} \leftarrow \mathsf{NTT}\,(\mathsf{NTT}^{-1}(\hat{a} \circ \mathsf{NTT}\,(s)) + e)$$

Stack Optimizations: KeyGen

On-the-fly error addition:

Instead of computing

$$\hat{b} \leftarrow \hat{a} \circ \mathrm{NTT}\ (s) + \mathrm{NTT}\ (e),$$

we compute

$$\hat{b} \leftarrow \mathrm{NTT}\ (\mathrm{NTT}^{-1}(\hat{a} \circ \mathrm{NTT}\ (s)) + e)$$

At the cost of 1 NTT^{-1}, the stack usage is decreased $\approx$1 polynomial.

Secret-key Size Optimization

- Store secret-key in NTT domain

Secret-key Size Optimization

- Store secret-key in NTT domain
- Store only 32 byte seed, re-run KeyGen during Decaps

Secret-key Size Optimization

- Store secret-key in NTT domain
- Store only 32 byte seed, re-run KeyGen during Decaps
- Store secret-key in normal domain

Secret-key Size Optimization

- Store secret-key in NTT domain
- Store only 32 byte seed, re-run KeyGen during Decaps
- Store secret-key in normal domain
- Store 32 byte secret-key seed

Secret-key Size Optimization

- Store secret-key in NTT domain
- Store only 32 byte seed, re-run KeyGen during Decaps
- Store secret-key in normal domain
- Store 32 byte secret-key seed ✓

Secret-key Size Optimization

- Store secret-key in NTT domain
- Store only 32 byte seed, re-run KeyGen during Decaps
- Store secret-key in normal domain
- Store 32 byte secret-key seed ✓

	Secret-key size	Decapsulation time
Kyber	-96%	+7%
NewHope	-96%	+18%
NewHope-Compact	-96%	+9%

Cycle Count Comparision

Scheme		Previous work	This work Speed	This work Stack Opt.	This work Short sk
NewHope	512	G: 588 683 [a] E: 918 558 [a] D: 904 800 [a]	G: 561 161 E: 865 243 D: 820 130	G: 578 850 E: 865 856 D: 820 742	G: 555 625 E: 865 018 D: 968 961
	1024	G: 1 161 112 [a] E: 1 777 918 [a] D: 1 760 470 [a]	G: 1 117 398 E: 1 687 272 D: 1 612 960	G: 1 157 331 E: 1 689 375 D: 1 614 804	G: 1 106 350 E: 1 686 964 D: 1 918 747
NH-Cmpct	512	-	G: 335 991 E: 531 453 D: 484 416	G: 349 692 E: 532 423 D: 484 945	G: 330 826 E: 531 282 D: 526 805
	768	-	G: 501 885 E: 782 315 D: 717 250	G: 524 181 E: 784 117 D: 718 950	G: 494 364 E: 782 471 D: 786 664
	1024	-	G: 658 581 E: 1 022 903 D: 940 023	G: 686 225 E: 1 025 503 D: 941 076	G: 648 206 E: 1 022 773 D: 1 030 809
Kyber	512	G: 514 291 [b] E: 652 769 [b] D: 621 245 [b]	G: 452 919 E: 586 380 D: 542 576	G: 461 693 E: 586 754 D: 543 332	G: 446 876 E: 586 403 D: 579 594
	768	G: 976 757 [b] E: 1 146 556 [b] D: 1 094 849 [b]	G: 860 227 E: 1 031 603 D: 967 124	G: 872 140 E: 1 030 764 D: 966 848	G: 850 228 E: 1 030 679 D: 1 021 439
	1024	G: 1 575 052 [b] E: 1 779 848 [b] D: 1 709 348 [b]	G: 1 394 148 E: 1 603 776 D: 1 522 900	G: 1 410 591 E: 1 603 988 D: 1 523 175	G: 1 381 514 E: 1 603 772 D: 1 595 530

[a] PQM4, commit be0c421aaecaad4443071bfcf62ad397d4f40832

[b] L. Botros, M. Kannwisher, P. Schwabe, Memory-Efficient High-Speed Implementation of Kyber on Cortex-M4, Africacrypt2019

Stack Usage Comparision in bytes

Scheme	NewHope (This work)	NewHope [KRSS]	NH-Cmpct (This work)	Kyber (This work)	Kyber [BKS19]
512	**G**: 2 056 **E**: 2 864 **D**: 2 880	**G**: 5 960 **E**: 9 168 **D**: 10 296	**G**: 2 160 **E**: 2 984 **D**: 2 984	**G**: 2 392 **E**: 2 344 **D**: 2 360	**G**: 2 952 **E**: 2 552 **D**: 2 560
768	-	-	**G**: 2 600 **E**: 3 936 **D**: 3 936	**G**: 3 240 **E**: 2 856 **D**: 2 864	**G**: 3 848 **E**: 3 128 **D**: 3 072
1024	**G**: 3 072 **E**: 4 904 **D**: 4 920	**G**: 11 080 **E**: 17 360 **D**: 19 576	**G**: 3 176 **E**: 5 024 **D**: 5 024	**G**: 3 776 **E**: 3 744 **D**: 3 760	**G**: 4 360 **E**: 3 584 **D**: 3 592

[KRSS] PQM4, commit be0c421aaecaad4443071bfcf62ad397d4f40832.
[BKS19] L. Botros, M. Kannwisher, P. Schwabe, Memory-Efficient High-Speed Implementation of Kyber on Cortex-M4, Africacrypt2019

Cycle Count Comparision for Polynomial Multiplication Functions

Scheme	Dimension	NTT	NTT^{-1}	$\circ$
NewHope	512	28662	22836	4736
	512 ([KRSS])	29767	35813	5459
	1024	63387	49880	9396
	1024 ([KRSS])	59752	71942	10836
	1024 ([AJS16])	86769	97340	14977
NewHope-Compact	512	12799	13052	7052
	768	19647	21226	12749
	1024	25536	26039	18510
Kyber	256	6847	6975	2317
	256 ([BKS19])	7754	9377	3076

[KRSS] PQM4, commit be0c421aaecaad4443071bfcf62ad397d4f40832.
[AJS16] E. Alkım, P. Jakubeit, P. Schwabe, NewHope on ARM Cortex-M, Space 2016
[BKS19] L. Botros, M. Kannwisher, P. Schwabe, Memory-Efficient High-Speed Implementation of Kyber on Cortex-M4, Africacrypt2019

Conclusion

- We propose various optimizations:
  - More efficient modular reduction,
  - More aggresive layer merging,
  - More aggresive lazy reduction,
  - Optimized small-degree polynomial multiplications,
  - Reduce the stack usage of KeyGen by adding the error term on-the-fly,
  - Trade-off between secret-key size and speed.
- Unified framework to compare Kyber, NewHope, and NewHope-Compact.

Conclusion

- We propose various optimizations:
  - ☐ More efficient modular reduction,
  - ☐ More aggresive layer merging,
  - ☐ More aggresive lazy reduction,
  - ☐ Optimized small-degree polynomial multiplications,
  - ☐ Reduce the stack usage of KeyGen by adding the error term on-the-fly,
  - ☐ Trade-off between secret-key size and speed.
- Unified framework to compare KYBER, NEWHOPE, and NEWHOPE-COMPACT.
- Using the Gentleman-Sande butterfly in the NTT can be faster for NEWHOPE?
- Using 32-bit signed integers instead of 16-bit can be faster?

Cortex-M4 optimizations for $\{$R,M$\}$LWE schemes

Erdem Alkım **Yusuf Alper Bilgin** Murat Cenk François Gérard

Source code available online at
https://github.com/erdemalkim/NewHope-Compact-M4

y.alperbilgin@gmail.com

asel san