2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS) | 979-8-3503-3071-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICPADS60453.2023.00383

Efficient NTTRU Implementation on ARMv8

Zhuo Zhang
School of Computer Science
Fudan University
Shanghai, China
zhuozhang22@m.fudan.edu.cn

Jieyu Zheng*
School of Computer Science
Fudan University
Shanghai, China
jyzheng23@m.fudan.edu.cn

Yunlei Zhao
School of Computer Science
Fudan University
Shanghai, China
ylzhao@fudan.edu.cn

***Abstract*—To tackle the challenges introduced by quantum computers to traditional public key cryptography, the domain of post-quantum cryptography (PQC) has taken center stage. Within this domain, the evaluation of computational performance emerges as a pivotal yardstick. Notably, NTTRU stands for one of the most efficient PQC schemes for key encapsulation mechanisms (KEM). This paper introduces the first optimized implementation of NTTRU on ARMv8 architecture. By leveraging the capabilities of the NEON engine, we strategically optimize the core modules of NTTRU: NTT/INTT, polynomial base case multiplication, and polynomial inversion. These optimizations have resulted in remarkable performance gains of 7.37×, 6.10×, 5.91×, and 4.43×, respectively when compared to the reference implementation. For the whole implementation, we achieve performance improvement of 2.85×, 2.36×, and 3.27× in key generation, encapsulation, and decapsulation respectively.**

***Index Terms*—Post-quantum cryptography, Key encapsulation mechanism, NTTRU, NEON parallel optimization, Number Theoretic Transform**

I. Introduction

The emergence of quantum computers poses a considerable threat to traditional public key cryptosystems. Using Shor's algorithm on quantum computers can solve some popular hard problems, such as large integer factorization and discrete logarithm problems, in polynomial time [1]. In order to tackle this challenge, post-quantum cryptography emerged, specifically to study encryption algorithms that are resistant to quantum computers. For the purpose of finding a suitable quantum-resistant public key encryption algorithm, in 2016, the National Institute of Standards and Technology (NIST) initiated a post-quantum cryptography standardization process for key encapsulation mechanisms (KEM) and digital signature schemes [2].

In the third round of the NIST competition, four KEM finalists were selected, three of which are based on lattice. Lattice cryptography is currently the most popular candidate type of post-quantum cryptography, and it has an excellent performance in terms of security, communication bandwidth, and computational efficiency.

NTRU is the abbreviation of **N**umber **T**heory **R**esearch **U**nit, originally proposed by Hoffstein, Pipher & Silverman in 1996 [3], which is the first practical public key cryptosystem based on the lattice hardness assumptions over polynomial rings. Over the past 27 years, despite encountering numerous attacks and cryptanalyses, the NTRU cryptosystem has showcased remarkable resilience. Even though NIST did not choose NTRU-based KEM schemes for standardization, we should not overlook their great potential for PQC research and deployment because of their attractiveness. Actually, NTRU-based PKE/KEM schemes have already incorporated some standardizations and Internet protocols.

Performance is a critical factor in the deployment of PQC KEM schemes. Among lattice-based key encapsulation mechanisms, NTTRU stands out as one of the most efficient post-quantum secure KEM schemes. NTTRU, an indistinguishability under adaptive chosen ciphertext attack (IND-CCA2) secure NTRU-based key encapsulation scheme, was introduced by Lyubashevsky and Seiler in [4], which was unfortunately later than the NIST-PQC standardization process. NTTRU chose a ring that can perform Number-Theoretic Transformations (NTT) to speed up polynomial multiplication. Moreover, the NTTRU design inherently lends itself to efficient vectorized implementation. Notably, to the best of our knowledge, an implementation on ARMv8 for NTTRU remains open.

ARMv8 architecture has extensive applications in Internet of Thing (IoT) devices. However, the imminent advent of quantum computers poses a threat to this security paradigm. Given the widespread use of IoT and the paramount importance of data security within this domain, it becomes crucial to include the evaluation of NTTRU performance with the ARMv8 architecture.

II. Implementation Details

A. NTT Operations

a) Layer Merging: The intervals of butterfly operations in NTT can be presented as $2^{i-1} \cdot 3$, $i = 8, 7, 6, \ldots, 1$. The NTTRU polynomial ring contains 7 levels splitting: level 7, level 6, ..., and level 1. Each level has a corresponding butterfly interval, here we call interval length, for example, in level 2, the length is $2^{2-1} \cdot 3 = 6$. We implement the layer merging technique in both NTT and INTT which is shown in Fig. 1.

b) Shuffle Operation: Shuffle operation mainly uses matrix rotation instruction TRN1/TRN2 to implement the transposition of vectors. The TRN1/TRN2 instruction reads

* Jieyu Zheng is the corresponding author of this paper.

2690-5965/23/$31.00 ©2023 IEEE
DOI 10.1109/ICPADS60453.2023.00383

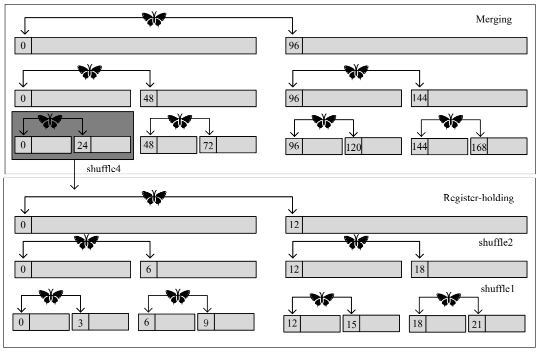

Fig. 1: Memory Access Optimization in NTT.

even/odd elements from two source registers starting from the low bit and puts them into the destination register in succession. There are 6 registers involved in each shuffle operation, resulting in a total of 48 coefficients. By shuffle operation in NTT/INTT, 480 LD1/ST1 instructions are saved at the cost of 288 TRN1/TRN2 instructions.

c) Lazy Reduction: In the standard NTT, after each butterfly operation, reduction is performed to ensure that no overflow occurs afterward. Since $q = 7681$ in NTTRU, to avoid overflowing, it is recommended to maintain the coefficients within the range of $(-4q, 4q)$. Compared with the standard implementation, lazy reduction saves 240 reduction operations in the NTT. In INTT, analyzing the size of coefficients is more complex because the inputs for INTT are not in their natural order, and it saves 216 Barrett reduction operations because of lazy reduction.

B. Polynomial Operations

Polynomial base case multiplication is a type of modular multiplication that involves two polynomials generated by incomplete NTT, and we utilize the Karatsuba [5] method to speed up it. The polynomial inversion needs to compute the polynomial matrix determinant inversion, which we have chosen to optimize by loop unrolling.

III. Performance Results

We offer a performance analysis of our refined implementation. Our benchmarking is conducted on the Raspberry Pi 4B (RPi 4 Model B), equipped with ARMv8-A instruction sets, a Cortex-A72 (1.8 GHz) CPU, and 4GB of RAM. The platform further supports ARMv8-A NEON SIMD instructions.

A. Performance of Polynomial Arithmetic

TABLE I: Comparison of polynomial arithmetic (Cycles)

Schemes	NTT	INTT	Basemul	Inversion
Ref-C [6]	35741	31315	13130	44111
This Work	4852	5130	2222	9963
Speedup	7.37	6.10	5.91	4.43

TABLE. I shows the performance results on ARM Cortex-A72 for the C reference implementation and NEON-optimized implementation. In the NTT, INTT, polynomial base multiplication, and polynomial inversion, it shows 7.37×, 6.10×, 5.91×, and 4.43× faster than reference implementation.

B. Performance of scheme

We embedded the functions parallelized using the NEON engine into the NTTRU scheme for testing. TABLE. II shows the results. Based on our proposed optimized implementation strategies, 2.85×, 2.36×, and 3.27× performance improvements are achieved in the key generation (KeyGen), encapsulation (Encaps), and decapsulation (Decaps) of NTTRU, compared with the C language reference implementation.

TABLE II: Comparison of NTTRU schemes (Cycles)

Schemes	KeyGen	Encaps	Decaps
Ref-C [6]	176888	135796	170352
This Work	62185	57555	52111
Speedup	2.85	2.36	3.27

IV. Conclusion

This paper presented a software implementation of parallelized optimization of the NTTRU scheme on ARMv8. Our NEON-optimized implementation outperforms the reference implementation in C language. Performance improvements of 7.37×, 6.10×, 5.91×, and 4.43× are achieved in NTT, INTT, polynomial base multiplication, and polynomial inversion operations, respectively. For the whole scheme, the performance improvement in KeyGen, Encaps, and Decaps are 2.85×, 2.36×, and 3.27×, respectively. The optimization strategies selected in this paper are available for most processors based on the ARMv8 architecture. Besides, layer merging and lazy reduction used in this paper can apply to other post-quantum cryptographic schemes using NTT based on specific rings.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[2] G. Alagic, G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, Y.-K. Liu, C. Miller, D. Moody, R. Peralta *et al.*, "Status report on the first round of the nist post-quantum cryptography standardization process," 2019.

[3] J. Hoffstein, "Ntru: a new high speed public key cryptosystem," *presented at the rump session of Crypto 96*, 1996.

[4] V. Lyubashevsky and G. Seiler, "Nttru: truly fast ntru using ntt," *Cryptology ePrint Archive*, 2019.

[5] D. Hankerson and A. Menezes, "Elliptic curve cryptography," in *Encyclopedia of Cryptography, Security and Privacy*. Springer, 2021, pp. 1–2.

[6] V. Lyubashevsky and G. Seiler, "Nttru: Truly fast ntru using ntt," https://github.com/gregorseiler/NTTRU, 2019.