A NEW EFFICIENT TMVP ALGORITHM AND AN APPLICATION FOR POST-QUANTUM CRYPTOGRAPHY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ANIL BURAK GÖKCE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY

SEPTEMBER 2023

Approval of the thesis:

A NEW EFFICIENT TMVP ALGORITHM AND AN APPLICATION FOR POST-QUANTUM CRYPTOGRAPHY

submitted by **ANIL BURAK GÖKCE** in partial fulfillment of the requirements for the degree of **Master of Science in Cryptography Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Selçuk-Kestel _____
Dean, Graduate School of **Applied Mathematics**

Assoc. Prof. Dr. Oğuz YAYLA _____
Head of Department, **Cryptography**

Assoc. Prof. Dr. Oğuz YAYLA _____
Supervisor, **Cryptography, METU**

Examining Committee Members:

Assoc. Prof. Dr. Fatih SULAK _____
Mathematics, Atılım University

Assoc. Prof. Dr. Oğuz YAYLA _____
Cryptography, METU

Prof. Dr. Murat CENK _____
Cryptography, METU

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ANIL BURAK GÖKCE

Signature :

ABSTRACT

A NEW EFFICIENT TMVP ALGORITHM AND AN APPLICATION FOR POST-QUANTUM CRYPTOGRAPHY

GÖKCE, Anıl Burak
M.S., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Oğuz YAYLA

September 2023, 49 pages

With the advancements in quantum computing, traditional cryptography is considered to have little life in the future. That is why NIST initiated a Post-quantum cryptography-related project in order to standardize quantum-secure cryptography. With the latest report on this project, the dominating quantum-secure problems appear to stem from lattice structures. Thus, efficient implementation techniques on multiplications of lattice elements, which is the bottleneck of lattice-based cryptography, emerged as an important topic. In this thesis, we suggest a new 5-way split TMVP algorithm and its application to lattice multiplications with an implementation of the lattice-based algorithm NTRU KEM. The results are promising, showing up to 34%, 35%, and 157% speed-up against Toom4-Karatsuba implementation in key generation, encapsulation, and decapsulation, respectively.

Keywords: Post-Quantum Cryptography, TMVP, Polynomial Multiplication, NTRU KEM, Lattice-based Cryptography

ÖZ

YENİ VE VERİMLİ BİR TMVP ALGORTİTMASI VE KUANTUM ERTESİ KRİPTOGRAFİYE BİR UYGULAMASI

GÖKCE, Anıl Burak
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Oğuz YAYLA

Eylül 2023, 49 sayfa

Kuantum bilgisayarlarının gelişimiyle birlikte, geleneksel kriptografinin gelecekte az bir zamanı kaldığı düşünülmektedir. Bu nedenle, NIST, kuantum ertesi kriptografiyle ilgili bir proje başlatarak kuantum-güvenli kriptografinin standardizasyonunu hedeflemiştir. Bu projenin en son raporuna göre, egemen kuantum-güvenli problemlerin kafes yapılarından kaynaklandığı görülmektedir. Dolayısıyla, kafes tabanlı kriptografinin en çok kaynağa ihtiyaç duyan konusu olan polinom çarpımlarının verimli uygulama teknikleri önemli bir konu olarak ortaya çıkmıştır. Bu tezde, yeni bir 5-yollu bölünmüş TMVP algoritması ve bu algoritmanın kafes tabanlı NTRU KEM'e uygulanmasını öneriyoruz. Sonuçlar Toom4-Karatsuba metoduna kıyasla anahtar oluşturma, kapsülleme ve dekapsülleme işlemlerinde sırasıyla 34%, 35%, ve 157% hız artışı göstermektedir.

Anahtar Kelimeler: Kuantum Ertesi Kriptografi, TMVP, Polinom Çarpması, NTRU KEM, Kafes Tabanlı Kriptografi

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to the following individuals who have played a crucial role in the completion of this thesis.

First and foremost, I would like to express my deepest appreciation to my beloved wife Neslihan. Not only has she been my pillar of strength and unwavering source of love and support, but she has also been an integral part of my academic journey. Her profound understanding of the subject matter and her invaluable insights have significantly contributed to the success of this thesis. From countless discussions to late-night brainstorming sessions, her academic assistance has been invaluable in shaping the direction of my research. Her unwavering belief in my abilities, coupled with her scholarly guidance, has been a constant source of motivation. I am truly fortunate to have such an exceptional partner who not only supports me emotionally but also enriches my academic pursuits.

I am immensely grateful to my supervisor, Murat Cenk. His guidance, expertise, and scholarly wisdom have been invaluable. His profound knowledge of the field, constructive feedback, and thoughtful suggestions have significantly enhanced the quality of this thesis. I am indebted to him for his patience, encouragement, and commitment to my academic growth.

I would also like to extend my sincere appreciation to my co-supervisor, Oğuz Yayla. His insightful perspectives, invaluable guidance, and firm support have greatly contributed to the success of this research endeavor.

Furthermore, I would like to express my deep gratitude to my entire family for their constant love, encouragement, and belief in my abilities. Their tireless support, both emotionally and morally, has been a source of strength throughout my academic journey.

In addition, I would like to thank all the friends and colleagues who have offered their support, encouragement, and insightful discussions during the course of this research. Their contributions and camaraderie have made this journey all the more enriching.

Finally, I extend my appreciation to the academic staff and administrative personnel of the Institute of Applied Mathematics. Their dedication to fostering a conducive research environment and providing access to valuable resources has been extremely helpful in the successful completion of this thesis.

To all those mentioned above and to countless others who have contributed in various ways, both seen and unseen, I offer my deepest gratitude. Without your support, this thesis would not have been possible.

TABLE OF CONTENTS

ABSTRACT vii

ÖZ ix

ACKNOWLEDGMENTS xi

TABLE OF CONTENTS xiii

LIST OF TABLES xvii

LIST OF FIGURES xviii

LIST OF ABBREVIATIONS xix

CHAPTERS

1 INTRODUCTION 1

1.1 The Arrival of Quantum Computers 2

1.2 NIST PQC Standardization Project 2

1.3 Efficiency of Lattice-based Algorithms 3

1.4 Literature Review of Multiplication in R_q 4

1.5 Our Contribution 5

1.6 Outline 5

2 BACKGROUND 7

2.1 Lattices and Some Hard Problems 7
2.1.1 Shortest Vector Problem 8
2.1.2 Closest Vector Problem 9
2.1.3 Learning With Error Problem 9
2.2 NTRU 10
2.2.1 Ideal Definitions 11
2.2.2 Parameter Sets 11
2.2.3 NTRU Public Key Encryption Scheme 13
2.2.4 NTRU Key Encapsulation Mechanism 15
2.3 Toeplitz Matrices and Their Usage in Multiplication in R_q . . 17
2.3.1 Toeplitz Matrices 18
2.3.2 Multiplication in R_q Using Toeplitz Matrices . . . 18
2.3.2.1 Case of $R_q = \mathbb{Z}_q[x]/(x^n - 1)$ 19
2.3.2.2 Case of $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ 20
2.4 Toeplitz Matrix-Vector Product (TMVP) Formulae 21
2.4.1 2×2 TMVP 21
2.4.2 3×3 TMVP 22
2.4.3 4×4 TMVP 23
2.4.4 Hybrid Usage of TMVP Formulae 24
3 NEW 5-WAY TOEPLITZ MATRIX-VECTOR PRODUCT ALGORITHM 25
3.1 Derivation 25

3.2 Complexity Calculation 30
3.2.1 Multi-evaluation 30
3.2.2 Products 31
3.2.3 Reconstruction 32
3.2.4 Total Complexity 32
3.3 Comparison with Other 5-way Split Algorithms 33
4 FASTER NTRU USING THE PROPOSED 5-WAY TMVP 35
4.1 NTRUHPS2048509 36
4.2 NTRUHPS2048677 36
4.3 NTRUHPS4096821 37
4.4 NTRUHRSS701 38
4.5 Implementation and Results 39
4.6 A Remark on Data Type Constraints 40
5 CONCLUSION 43
REFERENCES 45

LIST OF TABLES

Table 1.1 Candidates making it up to round 3 of NIST PQC Standardization Project [36] 3

Table 2.1 The defining parameter sets for different levels of NTRU submission [16] 13

Table 3.1 Five-Way Split Formula of TMVP —Multi-evaluation of the Toeplitz Matrix T 31

Table 3.2 Five-Way Split Formula of TMVP —Multi-evaluation of the vector A 31

Table 3.3 Five-Way Split Formula of TMVP —Recursive Products 32

Table 3.4 Five-Way Split Formula of TMVP —Reconstruction 32

Table 4.1 Number of operations vs hybrid multiplication method for the HPS2048509 36

Table 4.2 Number of operations vs hybrid multiplication method for the HPS2048677 37

Table 4.3 Number of operations vs hybrid multiplication method for the HPS4096821 38

Table 4.4 Number of operations vs hybrid multiplication method for the HRSS701 38

Table 4.5 Cycle counts comparison with NIST PQC 3rd round submission of NTRU[16] 40

LIST OF FIGURES

LIST OF ABBREVIATIONS

FFT	Fast Fourier Transform
GapSVP	Gap Shortest Vector Problem
KEM	Key Encapsulation Mechanism
LBC	Lattice-based Cryptography
LWE	Learning With Error
NIST	National Institute of Standards and Technology
NTT	Number Theoretic Transform
PKE	Public Key Encryption
PQC	Post Quantum Cryptography
$\mathbb{R}$	Set of real numbers
RLWE	Ring Learning With Error
SBP	Shortest Basis Problem
SVP	Shortest Vector Problem
TMVP	Toeplitz Matrix-Vector Product
$\mathbb{Z}$	Set of integers

CHAPTER 1

INTRODUCTION

Cryptography is the study of hidden writing, which enables humankind to share confidential data using insecure channels since ancient Egypt. On the other hand, requiring strong computational power, the real spike in this area happened after the invention of computers. Nowadays, encryption and decryption are not the only cryptographic algorithms. Digital signatures algorithms, key encapsulation mechanisms, symmetric and asymmetric encryption algorithms, and hash functions are some examples of cryptographic algorithms realized with the help of computers.

Public key cryptography is the branch of cryptography that includes cryptographic systems with public/private key pairs. As the names go, the public key is shared with everybody, while the private key is kept secret by the generator of the keys. Using this branch enables us to authenticate entities and/or share secret data with them. Diffie-Hellman Key Exchange [22], Digital Signature Algorithm [30], and RSA Encryption Scheme [42] are some examples of the widely known public key cryptosystems.

Besides the creative ways to use cryptography, it is crucial how efficiently we can use these ways. In cryptography, the algorithm's implementation platform determines the efficiency metrics. For example, the software implementations consider the efficiency of time consumed, while the hardware implementations value the efficiency of the area used. Therefore, research is constantly going on in this area. Some examples include [9], [13], and [14].

1.1 The Arrival of Quantum Computers

Quantum physics and, consequently, quantum computers have been researched for more than half a century. The first concrete examples of quantum circuits and computers were made real in the late 80s [28] and mid-90s [18], respectively. These scientific improvements allowed cryptanalysts to use quantum computers as a tool. Following this idea, Peter Shor invented an algorithm that changed the course of cryptography altogether [44]. With this algorithm, Shor described a way to break widely used public key encryption algorithms in a relatively short span of time in the presence of a quantum computer.

Though a quantum computer that can break the algorithms has not been created yet, it is expected to arrive soon. Therefore, researchers started working on different kinds of algorithms that are also resistant to such attacks that exploit the existence of quantum computers. For this aim, one of the promising areas is lattice-based cryptography. Scientists believe that cryptographic algorithms based on lattice problems will be used commonly in the era of quantum computers because of their strength against quantum attacks. As a result, efficient implementations of these algorithms have become an important research area.

1.2 NIST PQC Standardization Project

NIST initiated a standardization project for post-quantum cryptographic algorithms in 2017 [17]. This project aims to standardize the key encapsulation mechanisms and the digital signature algorithms that are going to be used in the era of quantum computers. As of the third round of this project, the competing candidates were as follows:

Table 1.1: Candidates making it up to round 3 of NIST PQC Standardization Project [36]

Key Encapsulation Mechanisms	Digital Signature Algorithms
Classic McEliece	Crystals-Dilithium
Crystals-Kyber	FALCON
NTRU	Rainbow
Saber	GeMSS
BIKE	Picnic
FrodoKEM	Sphincs+
HQC	
NTRU Prime	
SIKE	

In 2022, the round 3 results were published by NIST [4]. In the KEM part of the project, Crystals-Kyber is chosen to be standardized. Besides that, BIKE, Classic McEliece, HQC, and SIKE are promoted to the 4th round of the project for further consideration. The remaining KEMs are not considered for standardization anymore. In the signature part of the project, Crystals-Dilithium, Falcon, and Sphincs+ are chosen to be standardized. The algorithm selection process of this part is finished at this point, and the rest of the candidate algorithms are excluded.

1.3 Efficiency of Lattice-based Algorithms

One of the most important applications of cryptography is in the communication area, especially communication over the Internet. Today, we live in the information age, and the exchange of information is extremely fast. In this high-speed communication, it becomes important that the security measures taken are not too slow. In other words, the efficiency of cryptographic algorithms is what enables us to communicate fast and securely. Therefore, the efficiency of cryptographic algorithms takes an important place in the academy. In order to contribute to this area, we look into more efficient implementations of lattice-based algorithms in this thesis.

Most of the time, the efficiency bottleneck of cryptographic algorithms appears to be the multiplication operation. Therefore, researchers focus on the multiplication of mathematical objects (such as integers [46], floating points [21], polynomials [12],

and field elements[13, 14]) when they work on the efficiency of cryptographic algorithms. When it comes to lattice-based algorithms, the issue is similar. On the cryptosystems that are based on lattices, the multiplication is performed in a ring of the form $R_q = \mathbb{Z}_q[x]/(x^n \pm 1)$. Thus, from now on, we will focus on the multiplication in R_q. Before proceeding, it should be noted that a multiplication operation in R_q is actually the same as a cyclic (or nega-cyclic, depending on the dividing polynomial term of R_q) convolution operation.

1.4 Literature Review of Multiplication in R_q

The efficiency of multiplication in R_q depends on the multiplication algorithm used. Before presenting some commonly used multiplication algorithms, we can divide these algorithms into three categories. These categories are NTT, polynomial multiplication algorithms followed by polynomial reduction, and Toeplitz Matrix-Vector Product.

As the first category, the NTT algorithm exists [40]. This algorithm is derived from FFT to be applied to finite rings. Note that NTT is applicable only because of the convolutional nature of multiplication in R_q. NTT has the lowest asymptotic complexity among all the possible multiplication algorithms. On the other hand, some requirements need to be satisfied by the underlying mathematical structure to employ NTT. This limits the use cases of NTT and directs researchers to design new multiplication algorithms. Some examples incorporating NTT for multiplication in R_q can be found in [11, 33, 43].

The second category is to employ regular polynomial multiplication algorithms. These algorithms first multiply the polynomials using efficient polynomial multiplication methods and then perform a modular polynomial reduction in order to compute the resulting element of R_q. The improvement of such algorithms started with Karatsuba Algorithm [29]. Karatsuba is a 2-way split algorithm, i.e., the sub-polynomials that are to be multiplied are half the size of the original polynomials. After this improvement, the generalization is proposed by [46], which defines the ways to generate 3-way, 4-way, etc. splitting algorithms.

The third category for multiplication in R_q is TMVP. In fact, TMVP is a matrix-vector product algorithm. Additionally, TMVP can be used to compute convolutions. Therefore, similar to NTT, TMVP can be applied in order to multiply polynomials in finite polynomial rings. The history of using TMVP for multiplication started with Fan and Hasan employing TMVP to implement multiplication in $GF(2^n)$ in 2007 [25]. Then, in 2013, Cenk et al. integrated this technique into the multiplication of binary polynomials [13]. In 2016 and 2018, Ali and Cenk implemented faster modular integer multiplication for prime modulus $p = 2^{521} - 1$, which is employed in very important curves such as P-521 and E-521 [5, 6]. In 2018, Taşkın and Cenk integrated TMVP for modular integer multiplication of Curve25519 [45]. Finally, Paksoy and Cenk incorporated TMVP in order to implement multiplication in R_q in 2020 [37].

1.5 Our Contribution

In this thesis, we focus on the employment of TMVP for multiplication in R_q. Moreover, we offer a new multiplication algorithm to be used in R_q. The proposed algorithm is to employ a 5-way Toeplitz-matrix Vector Product. The formulae for this algorithm are derived by benefiting from the papers [15], [47], and [49]. Using this algorithm for the multiplication of the elements of R_q in lattice-based algorithms enables us to build more efficient implementations of LBC algorithms.

1.6 Outline

After the introduction, the basics related to the main topic of this thesis are given in chapter 2. Then, the new 5-way TMVP algorithm formulae and analysis, and an application of this algorithm are given in the chapters 3 and 4, respectively. Finally, the thesis concludes in chapter 5.

CHAPTER 2

BACKGROUND

In this chapter, the preliminary knowledge of lattice-based cryptography and lattice-based algorithms is given. We start from the ground up, by giving the basic definitions about lattice-based cryptography, including the definition of lattices itself. This part also states some hard problems of Lattice-based Cryptography. Then, we proceed to introduce the lattice-based algorithm we focus on. This algorithm is the NTRU that is submitted to NIST PQC Standardization Project [16]. After that, as we will use Toeplitz-matrix Vector Product in our new algorithm, an entrance to this area of cryptography is made by bringing in the related definitions such as the definition of Toeplitz matrices. In addition, the technique to be utilized for performing the multiplication in R_q using Toeplitz-matrix vector product is presented in this part. Finally, in the last section of this chapter, the TMVP formulae existing in the literature are given, and an insight into the usage of TMVP in multiplication is presented.

2.1 Lattices and Some Hard Problems

In order to work on the efficient multiplication techniques for lattice-based cryptography, we first need to understand the basics of lattices. In this section, the fundamental definitions that are used in the topic of lattices are given. The first one is the most basic one; the definition of a lattice:

Definition 2.1.1. *Let $a_1, a_2, ..., a_n$ be linearly independent vectors in $\mathbb{R}^n$. Then, the set*

$$\{k \in \mathbb{R}^n : k = \sum_{i=1}^{n} k_i a_i \textit{ for some } k_1, k_2, ..., k_n \in \mathbb{Z}\}$$

*is called a **lattice** in* $\mathbb{R}^n$.

In the following part of this section, the algorithmic problems related to lattices are explained. Moreover, some insight into their hardness is presented with the help of various important references.

2.1.1 Shortest Vector Problem

Definition 2.1.2. *Given a lattice* $\mathcal{L}$ *in* $\mathbb{R}^n$*, let* $B = \{a_1, a_2, ..., a_n\}$ *be a basis of* $\mathcal{L}$*, and let* $\lambda(\mathcal{L})$ *denote the length of the shortest vector in the lattice, i.e.,* $\lambda(\mathcal{L}) = \min_{l \in \mathcal{L}} ||l||$*. Then, given* B*, the problem of finding a non-zero vector* v *in the lattice such that* $||v|| = \lambda(\mathcal{L})$ *is called **Shortest Vector Problem**. As the approximation version of this problem, given* B *and an approximation parameter* $\gamma > 1$*, the problem of finding a non-zero vector* v *in the lattice such that* $||v|| \leq \gamma \cdot \lambda(\mathcal{L})$ *is denoted as* $\textbf{SVP}_\gamma$.

Shortest Vector Problem is considered to be a hard problem in post-quantum cryptography as of today. In 1996, Ajtai laid the ground for some computationally hard problems related to lattices [2], namely Shortest Vector Problem and Shortest Basis Problem. Later, in 2001, Ajtai et al. discovered an algorithm that solves SVP in an n-dimensional rational lattice, whose time complexity is randomized $2^{O(n)}$ [3]. Clearly, this algorithm is not practical to use in order to solve the problem. After that, Micciancio proved that approximating the shortest vector with a factor of $\sqrt{2}$ is hard, which supports and improves the results of Ajtai [35].

Definition 2.1.3. *Given a lattice* $\mathcal{L}$ *in* $\mathbb{R}^n$*, let* $B = \{a_1, a_2, ..., a_n\}$ *be a basis of* $\mathcal{L}$*, and let* $\lambda(\mathcal{L})$ *denote the length of the shortest vector in the lattice, i.e.,* $\lambda(\mathcal{L}) = \min_{l \in \mathcal{L}} ||l||$*. Then, given* B *and a parameter* $\beta \geq 1$*, and a number* $d > 0$*, the problem of deciding whether* $\lambda(\mathcal{L}) \leq d$ *or* $\lambda(\mathcal{L}) \geq \beta \cdot d$ *is defined as* $\textbf{GapSVP}_\beta$.

Note. *An oracle solving* $\textbf{GapSVP}_\beta$ *problem should return yes when* $\lambda(\mathcal{L}) \leq d$*, no when* $\lambda(\mathcal{L}) \geq \beta \cdot d$*, and an error if neither of the conditions is satisfied.*

Basically, GapSVP is the decision version of SVP. This fact makes us feel that GapSVP should be as hard as SVP. Since science does not work with feelings, various pieces

of research are conducted after GapSVP's introduction into the literature in order to prove that it is indeed a hard problem. First, Micciancio and Goldwasser showed that "GAPSVP is NP-hard under deterministic nonuniform polynomial time reductions" [35]. Later, Lyubashevsky et al. closed the deal in 2009, by proving the equivalence of Unique Shortest Vector Problem and GapSVP up to a polynomial [31].

2.1.2 Closest Vector Problem

Definition 2.1.4. *Given a lattice $\mathcal{L}$ in $\mathbb{R}^n$, let $B = \{a_1, a_2, ..., a_n\}$ be a basis of $\mathcal{L}$. Given a vector $y \in \mathcal{L}$, Closest Vector Problem is the problem of finding the vector $x \in \mathcal{L}$ such that $||x - y|| = \min_{a \in \mathcal{L}} ||a - y||$. In other words, compliant with the name, given a vector $y \in \mathcal{L}$ CVP is the problem of finding the closest vector to y.*

The most groundbreaking study on Closest Vector Problem is the one conducted by Goldreich et al. in 1999 [26]. Basically, they reduced SVP to CVP, which shows that SVP is not harder than CVP. Furthermore, as we know that SVP is a hard problem, the key point of [26] is the fact that CVP is also hard in the post-quantum era. In addition, Micciancio conducted a study in 2001, which made two important contributions on CVP to the literature [34]. First, they gave a simple proof of the hardness of CVP. Second, they showed that CVP remains hard even if the unlimited amount of preprocessing of the chosen lattice is allowed before revealing the challenge vector y.

2.1.3 Learning With Error Problem

After this introductory study on lattices, Regev proposed a generalized hard problem in 2005 [41]; namely, learning with error problem:

Definition 2.1.5. *Given a prime number p, vectors $s, a_1, a_2, ..., a_k \in \mathbb{Z}_q^n$, and a prob-*

ability distribution $\chi : \mathbb{Z}_q \to (0, 1)$*, consider the list of equations*

$$
\begin{aligned}
< s, a_1 > + e_1 &= t_1 \\
< s, a_2 > + e_2 &= t_2 \\
&\vdots \\
< s, a_k > + e_k &= t_k
\end{aligned}
$$

where each e_i *is sampled from the distribution* χ*. Then, the problem of obtaining* s *given the pairs* $(a_1, t_1), (a_2, t_2), ..., (a_k, t_k)$ *is called* ***learning with error*** *problem, and denoted as* $\boldsymbol{LWE}_{p,\chi}$ *.*

Importantly, in [41], Regev showed that LWE is at least as hard as GapSVP. Furthermore, this result indicates that LWE is quantum-secure since GapSVP is believed to be quantum-secure [39].

Later, in 2010, Lyubashevsky et al. added a new dimension to LWE by introducing Ring Learning With Error Problem (RLWE) into the literature [32]. With this novelty, the usage of lattice-based cryptography has become practically efficient in employing state-of-the-art technology. Furthermore, in 2012, Banerjee et al. presented Learning With Rounding Problem [8]. This problem maintains the efficiency of RLWE while adding the deterministic generation of the error terms.

2.2 NTRU

In this thesis, we apply the new 5-way TMVP algorithm to the NTRU Key Encapsulation Mechanism, which is one of the lattice-based candidates of the NIST PQC Standardization Project for round 3 [36]. In this section, we first summarize the mathematical structures and parameters for the NTRU version submitted to NIST for the third round of the Standardization Project, which is given in [16]. Then, we continue with the detailed specifications of NTRU Public Key Encryption and NTRU Key Encapsulation Mechanism, as defined in [16].

2.2.1 Ideal Definitions

NTRU KEM employs ideal lattices. Therefore, we first need to define the modulus polynomials of the ideals. The required polynomials are defined as follows:

$$\phi_1 = x - 1$$
$$\phi_n = \frac{x^n - 1}{x - 1} = x^{n-1} + x^{n-2} + ... + 1 = \sum_{i=0}^{n-1} x^i$$

Notice that

$$\phi_1 \phi_n = (x - 1) \cdot \frac{x^n - 1}{x - 1} = x^n - 1.$$

After this, we can pass to the definitions of the ideals used by NTRU KEM:

$$R_q = \mathbb{Z}_q[x]/(\phi_1 \phi_n)$$
$$S_q = \mathbb{Z}_q[x]/(\phi_n)$$
$$S_p = \mathbb{Z}_p[x]/(\phi_n)$$

As defined above, R_q is the set of polynomials with a maximum degree of $n - 1$ and with coefficients modulo q. Similarly, S_q is the set of polynomials with a maximum degree of $n - 2$ and with coefficients modulo q. Finally, S_p is the set of polynomials with a maximum degree of $n - 2$ and with coefficients modulo p.

2.2.2 Parameter Sets

NTRU parameters consist of three integers, four sets, and a function, denoted as $(n, p, q, \mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m, \textit{Lift})$ [16]. From these parameters, n, p, and q are coprime positive integers, $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_r$, and $\mathcal{L}_m$ are sets of polynomials, and *Lift* is a one-to-one function on $\mathcal{L}_m$ into $\mathbb{Z}[x]$ which satisfies $\textit{Lift}(m) = m \mod 3$ for all $m \in \mathcal{L}_m$. Finally, in order to check whether a parameter set is correct, one needs to satisfy that all of the coefficients of the polynomial

$$(p \cdot r \cdot g + f \cdot \textit{Lift}(m)) \mod (\phi_1 \phi_n) \tag{2.1}$$

are in the set $\{-q/2, -q/2 + 1, ..., q/2 - 1\}$ for each $(f, g, r, m) \in (\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m)$.

There are two versions of NTRU parameter sets: HPS and HRSS.

HPS parameter set requires the following to hold:

- n is a prime, and 2 and 3 are of order $n-1$ in the multiplicative group of integers $(\mathbb{Z}_n, \times)$,
- $p = 3$,
- q is a power of 2,
- $\mathcal{L}_f = S_3$,
- $\mathcal{L}_g$ is the set consisting of the elements of S_3 that have exactly $q/16 - 1$ coefficients equaling 1 and $q/16 - 1$ coefficients equaling -1,
- $\mathcal{L}_r = S_3$,
- $\mathcal{L}_m$ is the set consisting of the elements of S_3 that have exactly $q/16 - 1$ coefficients equaling 1 and $q/16 - 1$ coefficients equaling -1,
- *Lift*(m) is defined as the identity function.

Let $\mathcal{T}_+$ be the set consisting of the elements $v = \sum_i v_i x^i$ of S_3 that satisfy $\sum_i v_i v_{i+1} \geq 0$. Then, HRSS parameter set requires the following to hold:

- n is a prime, and 2 and 3 are of order $n-1$ in the multiplicative group of integers $(\mathbb{Z}_n, \times)$,
- $p = 3$,
- $q = 2^{\lceil 7/2 + \log_2(n) \rceil}$,
- $\mathcal{L}_f = \mathcal{T}_+$,
- $\mathcal{L}_g = \{\phi_1 \cdot v : v \in \mathcal{T}_+\}$,
- $\mathcal{L}_r = S_3$,
- $\mathcal{L}_m = S_3$,

- *Lift*$(m) = \phi_1 \cdot \underline{S3}(m/\phi_1)$ where $S3$ function reduces the input polynomial to S_3 with coefficients from $\{-1, 0, 1\}$.

In the NTRU submission [16], there exist four different parameter sets. Table 2.1 displays the information in detail:

Table 2.1: The defining parameter sets for different levels of NTRU submission [16]

Level Name	n	p	q
HPS2048509	509	3	2048
HPS2048677	677	3	2048
HPS4096821	821	3	4096
HRSS701	701	3	8192

2.2.3 NTRU Public Key Encryption Scheme

Note that our main goal for this section is to present NTRU Key Encapsulation Mechanism. In order to achieve this, NTRU PKE is needed to be explained first because NTRU KEM is based on it. Therefore, we will first introduce NTRU PKE. NTRU PKE consists of three algorithms; Key Generation, Encryption, and Decryption.

The pseudocode of the key generation of the submitted NTRU scheme is presented in Algorithm 1. In the first step, two ternary polynomials, namely f and g, are sampled with the help of $Sample\_fg$ function defined in [16]. Next, the inverse of the polynomial f in the ring S_q is computed as f_q. Then, using f_q, the public key h is calculated in R_q, as shown in the figure. After that, two more inverses are computed. These are h_q, which is the inverse of h in S_q, and f_p, which is the inverse of f in S_p. Finally, using the calculated polynomials, the public key and the private key are constructed as h and (f, f_p, h_q), respectively.

Algorithm 1 Pseudocode of the key generation algorithm of the NTRU PKE scheme [16]

Require: $seed$

Ensure: $((f, f_p, h_q), h)$

1: $(f, g) \leftarrow Sample\_fg(seed)$
2: $f_q \leftarrow 1/f \mod (q, \phi_n)$
3: $h \leftarrow (3 \cdot g \cdot f_q) \mod (q, \phi_1\phi_n)$
4: $h_q \leftarrow 1/h \mod (q, \phi_n)$
5: $f_p \leftarrow 1/f \mod (p, \phi_n)$
6: **return** $((f, f_p, h_q), h)$

The pseudocode of the NTRU encryption algorithm is shown in Algorithm 2. This algorithm gets a plaintext (r, m) and the public key h as inputs. Note that the plaintext components r and m are polynomials in $\mathcal{L}_r$ and $\mathcal{L}_m$, respectively. Moreover, Lift is a function that maps the polynomials in $\mathcal{L}_m$ to a polynomial in R_q. The encryption process starts with the lifting of the polynomial m. Then, the ciphertext polynomial c in R_q is computed as indicated in line 2 of the pseudocode in Algorithm 2.

Algorithm 2 Pseudocode of the encryption algorithm of the NTRU PKE scheme [16]

Require: $(h, (r, m))$

Ensure: c

1: $m' \leftarrow Lift(m)$
2: $c \leftarrow (r \cdot h + m') \mod (q, \phi_1\phi_n)$
3: **return** c

The pseudocode of the NTRU decryption algorithm is shown in Algorithm 3. In the first step, a validity check for the ciphertext polynomial is performed. Because of the nature of NTRU, the sum of the coefficients of c must be 0 in Z_q. Therefore, invalid (and potentially harmful) ciphertexts get eliminated at this point by making the decryption process return fail. Via the second and third steps of the pseudocode below, a candidate plaintext polynomial m is calculated. Then, the other component of the candidate plaintext, namely the polynomial r, is computed using m and the ciphertext polynomial c in steps 4 and 5. Finally, if this (r, m) pair is in $\mathcal{L}_r \times \mathcal{L}_m$,

the candidate plaintext is accepted and returned. Otherwise, the decryption process returns fail.

Algorithm 3 Pseudocode of the decryption algorithm of the NTRU PKE scheme [16]

Require: $((f, f_p, h_q), c)$

Ensure: $(r, m, fail)$

1: if $c \neq 0(\mod (q, \phi_1))$ **return** $(0, 0, 1)$
2: $a \leftarrow (c \cdot f) \mod (q, \phi_1 \phi_n)$
3: $m \leftarrow (a \cdot f_p) \mod (p, \phi_n)$
4: $m' \leftarrow Lift(m)$
5: $r \leftarrow ((c - m') \cdot h_q) \mod (q, \phi_n)$
6: **if** $(r, m) \in (\mathcal{L}_r \times \mathcal{L}_m)$ **then**
7: **return** $(r, m, 0)$
8: **else**
9: **return** $(0, 0, 1)$
10: **end if**

2.2.4 NTRU Key Encapsulation Mechanism

Now that NTRU PKE is defined, the way to obtain NTRU KEM is fairly straightforward. NTRU KEM is produced using a transformation, which converts any public key encryption scheme to a key encapsulation mechanism [16]. The resulting key generation, encapsulation, and decapsulation algorithms are presented in this section.

NTRU KEM Key Generation algorithm consists of two stages. It is presented in Algorithm 4. First, this algorithm runs NTRU PKE Key Generation algorithm in order to create the secret and public polynomials for PKE. Second, it samples 256-bit secret s from the uniform distribution. As a result, the private key is formed as (f, f_p, h_q, s), and the public key is simply the polynomial h.

Algorithm 4 Pseudocode of the key generation algorithm of the NTRU KEM scheme [16]

Require: $seed$

Ensure: $((f, f_p, h_q, s), h)$

1: $((f, f_p, h_q), h) \leftarrow$ Algorithm1$(seed)$

2: $s \leftarrow_{\$} \{0,1\}^{256}$

3: **return** $((f, f_p, h_q, s), h)$

NTRU KEM Encapsulation algorithm is given in Algorithm 5. The procedure is described as follows. First, the algorithm samples $coins$ from a uniform distribution randomly. Then, the algorithm runs $Sample\_rm$ function, which performs the generation of r and m polynomials conforming to the NTRU equation, which is Eq. (2.1), and the parameter set. And then, the algorithm runs NTRU PKE Encryption algorithm in order to encrypt the message (r, m) and generates the ciphertext c. After that, k is computed as the hash of (r, m), where H_1 is the hash function SHA3_256 [24]. Eventually, the algorithm returns the ciphertext c and the shared secret k.

Algorithm 5 Pseudocode of the encapsulation algorithm of the NTRU KEM scheme [16]

Require: h

Ensure: (c, k)

1: $coins \leftarrow_{\$} \{0,1\}^{256}$

2: $(r, m) \leftarrow Sample\_rm(coins)$

3: $c \leftarrow$ Algorithm 2$(h, (r, m))$

4: $k \leftarrow H_1(r, m)$

5: **return** (c, k)

The last NTRU KEM algorithm is the decapsulation algorithm. This algorithm is as shown in Algorithm 6. As can be seen below, the first step in decapsulation is to decrypt the ciphertext. After that, the algorithm computes k_1 using the hash function H_1 and assigning the input as (r, m). H_1 is defined as SHA3_256, just like encapsulation. Then, the algorithm computes k_2 as the hash of (s, c) using the hash function H_2, which is actually again SHA3_256 [24]. In the end, the algorithm returns k_1 if

$fail$ is 0, and returns k_2 otherwise.

Algorithm 6 Pseudocode of the decapsulation algorithm of the NTRU KEM scheme [16]

Require: (f, f_p, h_q, s)

Ensure: k

1: $(r, m, fail) \leftarrow$ Algorithm 3$((f, f_p, h_q), c)$
2: $k_1 \leftarrow H_1(r, m)$
3: $k_2 \leftarrow H_2(s, c)$
4: **if** $fail = 0$ **then**
5: **return** k_1
6: **else**
7: **return** k_2
8: **end if**

At this point, it is important to note that the shared secret k returned by NTRU KEM Encapsulation algorithm is the same as k_1 that is computed in NTRU KEM Decapsulation algorithm. Therefore, the shared secrets of the two parties (that are encapsulating and decapsulating) are the same as long as $fail$ returned by the decryption operation is 0. In other words, the secret is successfully shared between the parties whenever the decryption operation does not fail.

On the other hand, the decapsulation algorithm returns k_2, which is a pseudorandom value that gives no information about the shared secret, if the decryption fails. This step is added in order to comply with NIST's requirements on the standardization of post-quantum key encapsulation mechanism algorithms.

2.3 Toeplitz Matrices and Their Usage in Multiplication in R_q

In this section, the introduction to Toeplitz matrices and their usage for efficient multiplication in cryptography is done. In the first part of this section, the definition of Toeplitz matrices is given. In the second part, the technique that describes how to use TMVP for the multiplication operation in R_q is explained.

2.3.1 Toeplitz Matrices

First of all, the definition of Toeplitz matrices is given as follows:

Definition 2.3.1. *Let $T_{n\times n} = [t_{i,j}]$ where $i, j = 0, 1, \ldots n-1$ be a matrix. Then, T is called a **Toepltiz Matrix** if for all i, j $t_{i,j} = t_{(i-1),(j-1)}$ holds.*

For further intuition on Toeplitz matrices, the form of an $n \times n$ Toeplitz matrix T is shown below as an example:

$$\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ a_n & a_0 & a_1 & \ddots & a_{n-2} \\ a_{n+1} & a_n & a_0 & \ddots & a_{n-3} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{2(n-1)} & \cdots & a_{n+1} & a_n & a_0 \end{bmatrix} \tag{2.2}$$

At this point, an important property of Toepltiz matrices that we need to mention is the fact that the addition of two Toeplitz matrices results in a Toeplitz matrix. In other words, Toeplitz matrices are closed under addition.

2.3.2 Multiplication in R_q Using Toeplitz Matrices

In literature, there exists a way to employ Toeplitz-matrix vector products (TMVP) when multiplying polynomials in rings of the form $R_q = \mathbb{Z}_q[x]/(x^n \pm 1)$ [37]. This method stems from the fact that multiplication in R_q is actually equivalent to a cyclic (or negacyclic, depending on the quotient polynomial) convolution of the coefficient arrays of the polynomials. Therefore, the representation of this convolution as a matrix-vector product turns out to be the result of the multiplication of the two polynomials in R_q.

2.3.2.1 Case of $R_q = \mathbb{Z}_q[x]/(x^n - 1)$

Assume that $A(x), B(x) \in R_q = \mathbb{Z}_q[x]/(x^n - 1)$, i.e.,

$$A(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} \text{ and,}$$
$$B(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}.$$

We are going to convert one of the polynomials to a Toeplitz matrix and the other to a vector. The first polynomial, say A, is to be converted to a Toeplitz matrix. Given A is defined as above, the matrix that is the result of the conversion is as follows:

$$T = \begin{bmatrix} a_0 & a_{n-1} & a_{n-2} & \cdots & a_1 \\ a_1 & a_0 & a_{n-1} & \ddots & a_2 \\ a_2 & a_1 & a_0 & \ddots & a_3 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{n-1} & \cdots & a_2 & a_1 & a_0 \end{bmatrix}$$

Then, the second polynomial, which is B in this case, is transformed into a vector in the most trivial way, as follows:

$$V = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ \vdots \\ b_{n-1} \end{bmatrix}$$

Finally, the transformation of the polynomial multiplication $A(x) \cdot B(x)$ in R_q into the TMVP is realized by the Eq. (2.3).

$$T \cdot V = \begin{bmatrix} a_0 & a_{n-1} & a_{n-2} & \cdots & a_1 \\ a_1 & a_0 & a_{n-1} & \ddots & a_2 \\ a_2 & a_1 & a_0 & \ddots & a_3 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{n-1} & \cdots & a_2 & a_1 & a_0 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ \vdots \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ \vdots \\ c_{n-1} \end{bmatrix} \tag{2.3}$$

As a result, the multiplication outcome of this TMVP reveals the coefficients of the polynomial $C(x) = A(x) \cdot B(x)$ in R_q, where

$$C(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}.$$

2.3.2.2 Case of $R_q = \mathbb{Z}_q[x]/(x^n + 1)$

Assume that $A(x), B(x) \in R_q = \mathbb{Z}_q[x]/(x^n + 1)$, i.e.,

$$A(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} \text{ and,}$$
$$B(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}.$$

A very similar procedure to the previous section is employed. The only difference is at the step of the conversion of A to a Toeplitz matrix. Given A is defined as above, this time, the matrix that is the result of the conversion is as follows:

$$T = \begin{bmatrix} a_0 & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \ddots & -a_2 \\ a_2 & a_1 & a_0 & \ddots & -a_3 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{n-1} & \cdots & a_2 & a_1 & a_0 \end{bmatrix}$$

Then, the second polynomial B is transformed into a vector in the same way as in the previous section:

$$V = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ \vdots \\ b_{n-1} \end{bmatrix}$$

Finally, the transformation of the multiplication $A(x) \cdot B(x)$ in R_q into the TMVP is realized by the Eq. (2.4).

$$T \cdot V = \begin{bmatrix} a_0 & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \ddots & -a_2 \\ a_2 & a_1 & a_0 & \ddots & -a_3 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{n-1} & \cdots & a_2 & a_1 & a_0 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ \vdots \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ \vdots \\ c_{n-1} \end{bmatrix} \tag{2.4}$$

As a result, again, the multiplication outcome of this TMVP reveals the coefficients of the polynomial $C(x) = A(x) \cdot B(x)$ in R_q, where

$$C(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}.$$

2.4 Toeplitz Matrix-Vector Product (TMVP) Formulae

In this section, we will present the TMVP algorithms that are available in the literature. There exist 2-way, 3-way, and 4-way split algorithms that compute the product of a Toeplitz matrix and a vector [49]. In 2007, Fan and Hasan used 2-way split and 3-way split TMVP in order to increase the efficiency of multiplication in extended binary fields [25]. Later in 2012, Hasan and Negre generalized the TMVP formulae for binary fields [27]. Then, Ali and Cenk utilized 2-way TMVP and 3-way TMVP and built a highly efficient multiple precision integer multiplication algorithm [6]. Finally, Paksoy and Cenk came up with the formulization of 4-way split TMVP algorithm and applied it to Saber in 2020 [37]. In this section, we will cover these algorithms and their computational complexities.

2.4.1 2×2 TMVP

Let $T_{2\times2}$ be a Toeplitz matrix and $A_{2\times1}$ be a vector. Then, the product $T \cdot A$ can be computed using the following equations:

$$T.A = \begin{bmatrix} T_1 & T_0 \\ T_2 & T_1 \end{bmatrix} \cdot \begin{bmatrix} A_0 \\ A_1 \end{bmatrix} = \begin{bmatrix} Q_1 + Q_2 \\ Q_1 + Q_3 \end{bmatrix} \tag{2.5}$$

where

$$
\begin{aligned}
Q_1 &= T_1(A_0 + A_1) \\
Q_2 &= (T_0 - T_1)A_1 \\
Q_3 &= (T_2 - T_1)A_0
\end{aligned}
\tag{2.6}
$$

From Eqs. (2.5)-(2.6), we see that there exist 3 multiplications, 3 additions, and 2 double additions. Thus, we denote the complexity of this algorithm as $3M+3A+2A_d$. Moreover, the 2-TMVP formula can be generalized to matrices of size $n = 2^l$ for $l \in \mathbb{N}^+$. Then, the complexity of the algorithm is calculated as

$$
M(n) = 3M(n/2) + 3n - 1 \tag{2.7}
$$

2.4.2 3×3 TMVP

Let $T_{3\times3}$ be a Toeplitz matrix and $A_{3\times1}$ be a vector. Then, the product $T \cdot A$ can be computed using the following equations:

$$
T.A = \begin{bmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \\ T_4 & T_3 & T_2 \end{bmatrix} \cdot \begin{bmatrix} A_0 \\ A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} Q_3 + Q_4 + Q_6 \\ Q_2 - Q_4 + Q_5 \\ Q_1 - Q_2 - Q_3 \end{bmatrix} \tag{2.8}
$$

where

$$
\begin{aligned}
Q_1 &= (T_4 + T_3 + T_2)A_0 \\
Q_2 &= T_3(A_0 - A_1) \\
Q_3 &= T_2(A_0 - A_2) \\
Q_4 &= T_1(A_1 - A_2) \\
Q_5 &= (T_1 + T_2 + T_3)A_1 \\
Q_6 &= (T_0 + T_1 + T_2)A_2
\end{aligned}
\tag{2.9}
$$

From Eqs. (2.8)-(2.9), we see that there exist 6 multiplications, 9 additions, and 6 double additions. To further reduce the complexity of this algorithm, we can eliminate some recomputations performed during the computations of Q_i values. For example, the summation operation T_1+T_2 is required both in the computation of Q_5 and that of Q_6. We can compute $T_1 + T_2$ once during the computation of Q_5, and store it so that

we can use it later again in the computation of Q_6, decreasing the number of required addition operations by one. Therefore, we denote the complexity of this algorithm as $6M + 8A + 6A_d$.

Furthermore, the 3-TMVP formula can be generalized to matrices of size $n = 3^l$ for $l \in \mathbb{N}^+$. Then, the complexity of the algorithm is calculated as

$$M(n) = 6M(n/3) + 5n - 1 \tag{2.10}$$

2.4.3 4×4 TMVP

Let $T_{4\times4}$ be a Toeplitz matrix and $A_{4\times1}$ be a vector. Then, the product $T \cdot A$ can be computed using following equations [37]:

$$\begin{aligned} T.A &= \begin{bmatrix} T_3 & T_2 & T_1 & T_0 \\ T_4 & T_3 & T_2 & T_1 \\ T_5 & T_4 & T_3 & T_2 \\ T_6 & T_5 & T_4 & T_3 \end{bmatrix} \cdot \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{bmatrix} \\ &= \begin{bmatrix} Q_1 - Q_2 + 8Q_3 - 8Q_4 + 27Q_5 + Q_6 \\ Q_1 + Q_2 + 4Q_3 + 4Q_4 + 9Q_5 \\ Q_1 - Q_2 + 2Q_3 - 2Q_4 + 3Q_5 \\ Q_0 + Q_1 + Q_2 + Q_3 + Q_4 + Q_5 \end{bmatrix} \end{aligned} \tag{2.11}$$

where

$$\begin{aligned} Q_0 &= \frac{(12T_6 - 4T_5 - 15T_4 + 5T_3 + 3T_2 - T_1)A_0}{12} \\ Q_1 &= \frac{(12T_5 + 8T_4 - 7T_3 - 2T_2 + T_1)(A_0 + A_1 + A_2 + A_3)}{12} \\ Q_2 &= \frac{(-12T_5 + 16T_4 - T_3 - 4T_2 + T_1)(A_0 - A_1 + A_2 - A_3)}{24} \\ Q_3 &= \frac{(-6T_5 - T_4 + 7T_3 + T_2 - T_1)(A_0 + 2A_1 + 4A_2 + 8A_3)}{24} \\ Q_4 &= \frac{(-6T_5 - 5T_4 - 5T_3 + 5T_2 - T_1)(A_0 - 2A_1 + 4A_2 - 8A_3)}{120} \\ Q_5 &= \frac{(4T_5 - 5T_3 + T_1)(A_0 + 3A_1 + 9A_2 + 27A_3)}{120} \\ Q_6 &= (-12T_5 + 4T_4 + 15T_3 - 5T_2 - 3T_1 + T_0)A_3 \end{aligned} \tag{2.12}$$

According to Eqs. (2.11) and (2.12), there exist 43 addition, 7 multiplication, and 18 double addition operations in 4-way split TMVP algorithm aside from the scalar multiplication and shift operations. Similar to the 3-way case, it is possible to eliminate some operations to lower the total count of the operations required. With the employment of Algorithms 3 and 4 from [37], the total number of operations for the algorithm turns out to be $7M + 32A + 13A_d$.

Moreover, similar to the previous versions of TMVP, this version is also capable of being generalized to sizes $n = 4^l$ for $l \in \mathbb{N}^+$. Subsequently, the generalized complexity of the algorithm is presented by [37] as

$$M(n) = 7M(n/4) + 33n/2 - 21. \tag{2.13}$$

2.4.4 Hybrid Usage of TMVP Formulae

Although the asymptotic complexity of the higher-level TMVP algorithms is better, their performance decreases with the decreasing sizes of multiplications. Moreover, each algorithm, including the schoolbook multiplication algorithm, has a region where that algorithm is the most efficient one. Therefore, the most efficient multiplication method emerges from the combined usage of the algorithms.

In literature, various examples of hybrid employment of multiplication algorithms exist. For example, Bernstein came up with the idea of combined utilization of 2-way, 3-way, and 4-way split multiplication algorithms in 2009 [10]. In another instance, Adikari et al. used TMVP-2 and TMVP-3 algorithms together in order to achieve better cryptoprocessors [1]. Yet another occurrence of hybrid employment appears in [37]. In this work, Paksoy et al. employed TMVP-2, TMVP-3, and TMVP-4 in order to efficiently implement the multiplication operation used in Saber algorithm.

CHAPTER 3

NEW 5-WAY TOEPLITZ MATRIX-VECTOR PRODUCT ALGORITHM

In this chapter, we will present the original aspect of the thesis. Explicitly, we will give the information on the 5-way TMVP algorithm. We start with the derivation of the formulae for this algorithm. The derivation stems from some previous papers on TMVP and efficient multiplication techniques. The resulting formulae are presented as a Toeplitz-matrix vector product in Eq. (3.11). Then, we continue with the complexity calculation of this algorithm. After computing the complexity of various parts of the algorithm and summing the results, we get the total complexity of the algorithm as $M(n) = 13M(n/5) + (46n/5 - 18)A + 19n/5A_d$ for n being a power of 5. In the final part of this chapter, we give some insights into the differences between our 5-way TMVP algorithm and other 5-way split algorithms that are present in the literature.

3.1 Derivation

In this section, an efficient computation of the product of a Toeplitz matrix and a vector, whose entries are integers, is considered. In this regard, we give the 5-way TMVP formulae and their complexity evaluation. Note that such TMVP can be extended to be used in the multiplication of the Toeplitz matrix and vector of size $n = 5^l$ for $l = 2, 3, 4 \ldots$. In addition, 5-way TMVP can be used as the top-level split algorithm whenever the size of the multiplication is a multiple of 5.

The construction of the 5-way TMVP formulae is based on the technique provided in

the publications [15], [47], and [49]. In 2008, Cenk et al. discovered the formulae that enable more efficient implementation of multiplication in $\mathbb{F}_2$ [15]. Then, Venkatesan and Kumar used the formulae in [15] and the techniques in [49] in order to implement convolution operation efficiently in hardware [47]. In this thesis, we compute the 5-way TMVP formulae using the formulae in [47] and the technique in [49].

Let A and B be two degree-4 polynomials as follows:

$$A(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 \,,$$
$$B(x) = b_0 + b_1x + b_2x^2 + b_3x^3 + b_4x^4.$$

Moreover, the product polynomial $C(x) = A(x) \cdot B(x)$ can be shown as

$$C(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4x^4 + c_5x^5 + c_6x^6 + c_7x^7 + c_8x^8.$$

At this stage, the Eqs. (3.1) and (3.2) provide the coefficients of $C(x)$ [47].

$$\begin{aligned}
&P_1 = a_0b_0 \qquad P_2 = a_1b_1 \qquad P_3 = a_2b_2 \qquad P_4 = a_3b_3 \qquad P_5 = a_4b_4 \\
&P_6 = (a_0 - a_1)(b_0 - b_1) \qquad P_7 = (a_0 + a_2)(b_0 + b_2) \\
&P_8 = (a_2 + a_4)(b_2 + b_4) \qquad P_9 = (a_3 - a_4)(b_3 - b_4) \\
&P_{10} = (a_0 - a_2 - a_3)(b_0 - b_2 - b_3) \qquad P_{11} = (a_1 + a_2 - a_4)(b_1 + b_2 - b_4) \\
&P_{12} = (a_0 - a_1 - a_3 + a_4)(b_0 - b_1 - b_3 + b_4) \\
&P_{13} = (a_0 - a_1 - a_2 - a_3 + a_4)(b_0 - b_1 - b_2 - b_3 + b_4)
\end{aligned} \tag{3.1}$$

$$\begin{aligned}
c_0 &= P_1 \\
c_1 &= -P_6 + P_2 + P_1 \\
c_2 &= P_7 - P_1 + P_2 - P_3 \\
c_3 &= P_{13} - P_{12} - P_{10} + P_8 + P_4 - P_3 + P_1 - P_5 \\
c_4 &= P_{13} - P_{11} - P_{10} - P_6 - P_9 + P_1 + P_2 + P_3 + P_3 + P_4 + P_5 \\
c_5 &= P_{13} - P_{12} - P_{11} + P_7 + P_5 - P_3 + P_2 - P_1 \\
c_6 &= P_8 - P_5 - P_3 + P_4 \\
c_7 &= -P_9 + P_5 + P_4 \\
c_8 &= P_5
\end{aligned} \tag{3.2}$$

Moreover, we have the calculation of the coefficients of C stemming directly from the schoolbook multiplication technique as follows:

$$\begin{aligned}
c_0 &= a_0b_0 \\
c_1 &= a_0b_1 + a_1b_0 \\
c_2 &= a_0b_2 + a_1b_1 + a_2b_0 \\
c_3 &= a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0 \\
c_4 &= a_0b_4 + a_1b_3 + a_2b_2 + a_3b_1 + a_4b_0 \\
c_5 &= a_1b_4 + a_2b_3 + a_3b_2 + a_4b_1 \\
c_6 &= a_2b_4 + a_3b_3 + a_4b_2 \\
c_7 &= a_3b_4 + a_4b_3 \\
c_8 &= a_4b_4
\end{aligned} \tag{3.3}$$

Therefore, using Eqs. (3.2) and (3.3), we get the following set of equations:

$$\begin{aligned}
&a_0b_0 = P_1 \\
&a_0b_1 + a_1b_0 = -P_6 + P_2 + P_1 \\
&a_0b_2 + a_1b_1 + a_2b_0 = P_7 - P_1 + P_2 - P_3 \\
&a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0 = P_{13} - P_{12} - P_{10} + P_8 + P_4 - P_3 + P_1 - P_5 \\
&a_0b_4 + a_1b_3 + a_2b_2 + a_3b_1 + a_4b_0 = P_{13} - P_{11} - P_{10} - P_6 - P_9 + P_1 + P_2 \\
&\qquad\qquad\qquad\qquad\qquad\qquad + P_3 + P_3 + P_4 + P_5 \\
&a_1b_4 + a_2b_3 + a_3b_2 + a_4b_1 = P_{13} - P_{12} - P_{11} + P_7 + P_5 - P_3 + P_2 - P_1 \\
&a_2b_4 + a_3b_3 + a_4b_2 = P_8 - P_5 - P_3 + P_4 \\
&a_3b_4 + a_4b_3 = -P_9 + P_5 + P_4 \\
&a_4b_4 = P_5
\end{aligned} \tag{3.4}$$

From here, we sum the equations after multiplying the first equation by T_8, the second by T_7, and so on until the last one, which is multiplied by T_0. Then, we rearrange the left-hand side of the resulting equation with respect to b_i. The resulting equation is as

follows:

$$
\begin{aligned}
&(T_8a_0 + T_7a_1 + T_6a_2 + T_5a_3 + T_4a_4)b_0 \\
&\quad + (T_7a_0 + T_6a_1 + T_5a_2 + T_4a_3 + T_3a_4)b_1 \\
&\quad + (T_6a_0 + T_5a_1 + T_4a_2 + T_3a_3 + T_4a_4)b_2 \\
&\quad + (T_5a_0 + T_4a_1 + T_3a_2 + T_2a_3 + T_1a_4)b_3 \\
&\quad + (T_4a_0 + T_3a_1 + T_2a_2 + T_1a_3 + T_0a_4)b_4 \\
&= T_8P_1 + T_7(-P_6 + P_2 + P_1) + T_6(P_7 - P_1 + P_2 - P_3) \\
&\quad + T_5(P_{13} - P_{12} - P_{10} + P_8 + P_4 - P_3 + P_1 - P_5) \\
&\quad + T_4(P_{13} - P_{11} - P_{10} - P_6 - P_9 + P_1 + P_2 + 2P_3 + P_4 + P_5) \\
&\quad + T_3(P_{13} - P_{12} - P_{11} + P_7 + P_5 - P_3 + P_2 - P_1) \\
&\quad + T_2(P_8 - P_5 - P_3 + P_4) \\
&\quad + T_1(-P_9 + P_5 + P_4) + T_0P_5
\end{aligned}
\tag{3.5}
$$

Afterward, denoting the equation's left-hand expression as $f(T, a, b)$, we transform its right-hand expression. In explicit, we break down the nine products on the right-hand and sort the results with respect to P_i. Thus, we get Eq. (3.6).

$$
\begin{aligned}
f(T, a, b) =&(T_8 + T_7 - T_6 + T_5 + T_4 - T_3)P_1 + (T_7 + T_6 + T_4 + T_3)P_2 \\
&+ (-T_6 - T_5 + 2T_4 - T_3 - T_2)P_3 + (T_5 + T_4 + T_2 + T_1)P_4 \\
&+ (-T_5 + T_4 + T_3 - T_2 + T_1 + T_0)P_5 \\
&+ (-T_7 - T_4)P_6 + (T_6 + T_3)P_7 \\
&+ (T_5 + T_2)P_8 + (-T_1 - T_4)P_9 \\
&+ (-T_5 - T_4)P_{10} + (-T_4 - T_3)P_{11} \\
&+ (-T_5 - T_3)P_{12} + (T_5 + T_4 + T_3)P_{13}.
\end{aligned}
\tag{3.6}
$$

Note that, by Eq. (3.1), each P_i is of the form $L_i(a)L_i(b)$. Moreover, for each term on the right side of Eq. (3.6), the coefficient of P_i is a linear expression $L'_i(T)$. Therefore, we can replace $L'_i(T)P_i$ with $Q_iL_i(b)$ for each $i = 1, 2, ..., 13$, where $Q_i = L'_i(T)L_i(a)$ are as follows:

$$
\begin{aligned}
Q_1 &= (T_8 + T_7 - T_6 + T_5 + T_4 - T_3)a_0 \\
Q_2 &= (T_7 + T_6 + T_4 + T_3)a_1 \\
Q_3 &= (-T_6 - T_5 + 2T_4 - T_3 - T_2)a_2 \\
Q_4 &= (T_5 + T_4 + T_2 + T_1)a_3 \\
Q_5 &= (-T_5 + T_4 + T_3 - T_2 + T_1 + T_0)a_4 \\
Q_6 &= (-T_7 - T_4)(a_0 - a_1) \\
Q_7 &= (T_6 + T_3)(a_0 + a_2) \\
Q_8 &= (T_5 + T_2)(a_2 + a_4) \\
Q_9 &= (-T_4 - T_1)(a_3 - a_4) \\
Q_{10} &= (-T_5 - T_4)(a_0 - a_2 - a_3) \\
Q_{11} &= (-T_4 - T_3)(a_1 + a_2 - a_4) \\
Q_{12} &= (-T_5 - T_3)(a_0 - a_1 - a_3 + a_4) \\
Q_{13} &= (T_5 + T_4 + T_3)(a_0 - a_1 - a_2 - a_3 + a_4)
\end{aligned}
\tag{3.7}
$$

After the replacement of each $L_i'(T)P_i$ by $Q_iL_i(b)$, we get Eq. (3.8).

$$
\begin{aligned}
f(T,a,b) = {} & Q_1b_0 + Q_2b_1 + Q_3b_2 + Q_4b_3 + Q_5b_4 + Q_6(b_0 - b_1) \\
& + Q_7(b_0 + b_2) + Q_8(b_2 + b_4) + Q_9(b_3 - b_4) + Q_{10}(b_0 - b_2 - b_3) \\
& + Q_{11}(b_1 + b_2 - b_4) + Q_{12}(b_0 - b_1 - b_3 + b_4) \\
& + Q_{13}(b_0 - b_1 - b_2 - b_3 + b_4)
\end{aligned}
\tag{3.8}
$$

Then, we reorder the new expression's right side in terms of b_i and get

$$
\begin{aligned}
f(T,a,b) = {} & (Q_1 + Q_6 + Q_7 + Q_{10} + Q_{12} + Q_{13})b_0 \\
& + (Q_2 - Q_6 + Q_{11} - Q_{12} - Q_{13})b_1 \\
& + (Q_3 + Q_7 + Q_8 - Q_{10} + Q_{11} - Q_{13})b_2 \\
& + (Q_4 + Q_9 - Q_{10} - Q_{12} - Q_{13})b_3 \\
& + (Q_5 + Q_8 - Q_9 - Q_{11} + Q_{12} + Q_{13})b_4.
\end{aligned}
\tag{3.9}
$$

After that, we can equate the coefficients of b_i in Eq. (3.9) with the coefficients of b_i

in the left-hand expression of Eq. (3.5). This results in the following five equations:

$$\begin{aligned}
T_8a_0 + T_7a_1 + T_6a_2 + T_5a_3 + T_4a_4 &= Q_1 + Q_6 + Q_7 + Q_{10} + Q_{12} + Q_{13} \\
T_7a_0 + T_6a_1 + T_5a_2 + T_4a_3 + T_3a_4 &= Q_2 - Q_6 + Q_{11} - Q_{12} - Q_{13} \\
T_6a_0 + T_5a_1 + T_4a_2 + T_3a_3 + T_2a_4 &= Q_3 + Q_7 + Q_8 - Q_{10} + Q_{11} - Q_{13} \\
T_5a_0 + T_4a_1 + T_3a_2 + T_2a_3 + T_1a_4 &= Q_4 + Q_9 - Q_{10} - Q_{12} - Q_{13} \\
T_4a_0 + T_3a_1 + T_2a_2 + T_1a_3 + T_0a_4 &= Q_5 + Q_8 - Q_9 - Q_{11} + Q_{12} + Q_{13}
\end{aligned} \tag{3.10}$$

Finally, Eq. (3.10) can be transformed to a matrix-vector product equation because it is actually a linear system of equations. The result of the transformed matrix-vector product is actually a Toepltiz matrix-vector product since the matrix is a Toepltiz matrix. As a result, the right-hand side indicates the required five-way split formulae in order to compute the product. The explicit TMVP equation is given below:

$$\begin{bmatrix} T_4 & T_3 & T_2 & T_1 & T_0 \\ T_5 & T_4 & T_3 & T_2 & T_1 \\ T_6 & T_5 & T_4 & T_3 & T_2 \\ T_7 & T_6 & T_5 & T_4 & T_3 \\ T_8 & T_7 & T_6 & T_5 & T_4 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} Q_5 + Q_8 - Q_9 - Q_{11} + Q_{12} + Q_{13} \\ Q_4 + Q_9 - Q_{10} - Q_{12} - Q_{13} \\ Q_3 + Q_7 + Q_8 - Q_{10} + Q_{11} - Q_{13} \\ Q_2 - Q_6 + Q_{11} - Q_{12} - Q_{13} \\ Q_1 + Q_6 + Q_7 + Q_{10} + Q_{12} + Q_{13} \end{bmatrix} \tag{3.11}$$

3.2 Complexity Calculation

In this section, the complexity of the proposed 5-way TMVP algorithm is calculated. The calculation is divided into three parts. The complexities of multi-evaluation step, sub-multiplication step, and reconstruction step is calculated in the following sections in respective order.

3.2.1 Multi-evaluation

First, we have the multi-evaluation computations on matrix T. The computations are given in Table 3.1. Note that a Toeplitz matrix can be defined by its first column and first row. In total, we need to keep track of $2k - 1$ integers in order to store and

operate on an $k \times k$ Toeplitz matrix with integer elements. Therefore, the addition of two Toeplitz matrices costs $2k - 1$ integer additions.

Table 3.1: Five-Way Split Formula of TMVP —Multi-evaluation of the Toeplitz Matrix T

Underlying Computations	Addition Cost
$R_1 = -(T_7 + T_4)$	$2n/5 - 1$
$R_2 = T_6 + T_3$	$2n/5 - 1$
$R_3 = T_5 + T_2$	$2n/5 - 1$
$R_4 = -(T_4 + T_1)$	$2n/5 - 1$
$R_5 = -(T_5 + T_4)$	$2n/5 - 1$
$R_6 = -(T_4 + T_3)$	$2n/5 - 1$
$R_7 = -(T_5 + T_3)$	$2n/5 - 1$
$R_8 = T_4 - R_7$	$2n/5 - 1$
$R_9 = -R_3 - R_6 + T_0 + T_1$	$6n/5 - 3$
$R_{10} = R_3 - R_4$	$2n/5 - 1$
$R_{11} = 2T_4 - R_2 - R_3$	$4n/5 - 2$
$R_{12} = R_2 - R_1$	$2n/5 - 1$
$R_{13} = T_8 + T_5 - R_1 - R_2$	$6n/5 - 3$
Total	$(36n/5 - 18)\ A$

Second, we have the multi-evaluation computations on vector A. These computations are shown in Table 3.2

Table 3.2: Five-Way Split Formula of TMVP —Multi-evaluation of the vector A

Underlying Computations	Addition Cost
$R'_1 = a_0 - a_1$	$n/5$
$R'_2 = a_0 + a_2$	$n/5$
$R'_3 = a_2 + a_4$	$n/5$
$R'_4 = a_3 - a_4$	$n/5$
$R'_5 = a_0 - a_2 - a_3$	$2n/5$
$R'_6 = a_1 + a_2 - a_4$	$2n/5$
$R'_7 = R'_1 - R'_4$	$n/5$
$R'_8 = R'_7 - a_2$	$n/5$
Total	$2n\ A$

3.2.2 Products

In 5-way TMVP algorithm, there exist thirteen sub-multiplication operations. The operations are presented in Table 3.3.

Table 3.3: Five-Way Split Formula of TMVP —Recursive Products

Underlying Computations	Multiplication Cost
$Q_1 = R_{13}a_0$	$M(n/5)$
$Q_2 = R_{12}a_1$	$M(n/5)$
$Q_3 = R_{11}a_2$	$M(n/5)$
$Q_4 = R_{10}a_3$	$M(n/5)$
$Q_5 = R_9a_4$	$M(n/5)$
$Q_6 = R_1R'_1$	$M(n/5)$
$Q_7 = R_2R'_2$	$M(n/5)$
$Q_8 = R_3R'_3$	$M(n/5)$
$Q_9 = R_4R'_4$	$M(n/5)$
$Q_{10} = R_5R'_5$	$M(n/5)$
$Q_{11} = R_6R'_6$	$M(n/5)$
$Q_{12} = R_7R'_7$	$M(n/5)$
$Q_{13} = R_8R'_8$	$M(n/5)$
Total	$13M(n/5)$

3.2.3 Reconstruction

In the reconstruction step, the product $T \cdot A$ is reconstructed. The computations required for this step are provided in Table 3.4.

Table 3.4: Five-Way Split Formula of TMVP —Reconstruction

Underlying Computations	Addition Cost
$U_1 = Q_{12} + Q_{13}$	$n/5$
$U_2 = Q_9 - U_1$	$n/5$
$U_3 = Q_5 + Q_8 - Q_{11} - U_2$	$3n/5$
$U_4 = Q_4 - Q_{10} + U_2$	$2n/5$
$U_5 = Q_3 + Q_7 + Q_8 - Q_{10} + Q_{11} - Q_{13}$	n
$U_6 = Q_2 - Q_6 + Q_{11} - U_1$	$3n/5$
$U_7 = Q_1 + Q_6 + Q_7 + Q_{10} + U_1$	$4n/5$
Total	$19n/5\ A_d$

3.2.4 Total Complexity

We calculate the total complexity by adding the costs shown in Tables 3.1, 3.2, 3.3, and 3.4. After adding values in these tables, the result turns out to be as follows:

$$M(n) = 13M(n/5) + (46n/5 - 18)A + 19n/5A_d \tag{3.12}$$

Furthermore, we can generalize this result for matrix-vector products in $\mathbb{Z}_q$ of size $n = 5^l$ for $l = 1, 2, 3, \ldots$. Note that, since we are in $\mathbb{Z}_q$, $A = A_d = 1$. Thus, the computation of the resulting complexity is as follows:

$$\begin{aligned} M(n) &= 13M(n/5) + (46n/5 - 18) + 19(n/5) \\ M(n) &= 13M(n/5) + 13n - 18 \end{aligned} \tag{3.13}$$

3.3 Comparison with Other 5-way Split Algorithms

When it comes to 5-way split algorithms, the first one that comes to mind is the 5-way version of Toom-Cook multiplication algorithm [19]. On the other hand, this algorithm requires various division operations. Furthermore, this requirement reveals two obstacles when the mathematical structure is R_q. Firstly, it introduces a number of multiplication operations in the finite field that the quotient polynomial ring relies on. This excessive amount of operations decreases the efficiency of the algorithm for the use case o R_q. Secondly, when the divisor is not relatively prime with the base field size (q of R_q in NTRU), it is impossible to calculate the result theoretically. When the best case scenario of q being a power of 2 happens, we can divide a number in $\mathbb{Z}_q$ by 2 by shifting it right by 1, as long as the least significant bit of the number is 0.

Fortunately, the proposed TMVP algorithm does not contain any divisions, allowing us to avoid all the mentioned complications.

CHAPTER 4

FASTER NTRU USING THE PROPOSED 5-WAY TMVP

In this part of the thesis, various hybrid multiplication techniques are considered for each NTRU parameter set defined in [16]. Moreover, the 5-way TMVP algorithm described in chapter 3 has been chosen as the highest level for each hybrid multiplication method so that the effect of the new algorithm can be observed at its full potential. Note that the sizes of the polynomials of NTRU are not multiples of 5. Thus, in order to employ the 5-way TMVP algorithm, a padding operation is to be done. The padded length is denoted as n' throughout this chapter.

For each version of NTRU, an exploration is performed to find the hybrid multiplication method that has the highest software performance. The exploration consists of two stages. In the first stage, several hybrid multiplication methods are analyzed with respect to the number of addition and multiplication operations they contain. The tables showing these numbers are presented in the following sections. In the second stage, promising hybrid methods are implemented, and their performances are compared. Thus, the chosen hybrid method is ensured to be the one having the highest performance among the considered methods.

Each hybrid multiplication method contains a series of TMVP algorithms followed by the schoolbook matrix-vector product algorithm. This is because it is known that the schoolbook algorithm outperforms the other algorithms when the size is small [38].

4.1 NTRUHPS2048509

For this parameter set, the polynomials to be multiplied lie in the ring

$$\mathbb{Z}_{2048}[x]/(x^{509}-1).$$

In order to employ 5-way TMVP, the polynomials are padded to different multiples of 5. Table 4.1 presents the number of operations for hybrid multiplication methods that are used for the HPS2048509 version.

Table 4.1: Number of operations vs hybrid multiplication method for the HPS2048509

Method	#Addition Operations	#Multiplication Operations	#Total Operations
$510 \xrightarrow{\text{5-way}} 102 \xrightarrow{\text{3-way}} 34 \xrightarrow{\text{2-way}} 17$	77516	67626	145142
$520 \xrightarrow{\text{5-way}} 104 \xrightarrow{\text{4-way}} 26 \xrightarrow{\text{2-way}} 13$	76466	46137	122603
$525 \xrightarrow{\text{5-way}} 105 \xrightarrow{\text{5-way}} 21 \xrightarrow{\text{3-way}} 7$	70832	49686	120518
$540 \xrightarrow{\text{5-way}} 108 \xrightarrow{\text{4-way}} 27 \xrightarrow{\text{3-way}} 9$	78864	44226	123090

As can be seen from the table above, the least number of operations is achieved when the padded length is set to 525. However, the difference in the number of operations between padding to 520, 525, and 540 is not significant. Therefore, we decided to implement these three methods and use the implementation results as the tie-breaker. The implementation results show that the most performant method appears when padding is done up to 520. As a result, the hybrid method given in Eq. (4.1) is employed in the implementation.

$$520 \xrightarrow{\text{5-way TMVP}} 104 \xrightarrow{\text{4-way TMVP}} 26 \xrightarrow{\text{2-way TMVP}} 13 \tag{4.1}$$

4.2 NTRUHPS2048677

For this parameter set, the polynomials to be multiplied lie in the ring

$$\mathbb{Z}_{2048}[x]/(x^{677}-1).$$

In order to employ 5-way TMVP, some polynomial padding methods are employed. Table 4.2 presents the number of operations for hybrid multiplication methods that are employed for the HPS2048677 version.

Table 4.2: Number of operations vs hybrid multiplication method for the HPS2048677

Method	#Addition Operations	#Multiplication Operations	#Total Operations
$680 \xrightarrow{\text{5-way}} 136 \xrightarrow{\text{4-way}} 34 \xrightarrow{\text{2-way}} 17$	118582	78897	197479
$690 \xrightarrow{\text{5-way}} 138 \xrightarrow{\text{3-way}} 46 \xrightarrow{\text{2-way}} 23$	137084	123786	260870
$700 \xrightarrow{\text{5-way}} 140 \xrightarrow{\text{5-way}} 28 \xrightarrow{\text{4-way}} 7$	155170	57967	213137
$700 \xrightarrow{\text{5-way}} 140 \xrightarrow{\text{5-way}} 28 \xrightarrow{\text{2-way}} 14$	121201	99372	220573

According to Table 4.2, there are three candidates for the most efficient algorithm. It is common knowledge that the multiplication operation is more costly than the addition operation. This knowledge brings the third option of Table 4.2 forth. On the other hand, most of today's processors can pipeline the instructions, which virtually equalizes the cost of addition and multiplication operations. Therefore, the first and last methods presented in Table 4.2 are also considered for the implementation stage. After the implementations are finished, it is seen that the highest performance is achieved with the method of padding to 680, which is given in Eq. (4.2).

$$680 \xrightarrow{\text{5-way TMVP}} 136 \xrightarrow{\text{4-way TMVP}} 34 \xrightarrow{\text{2-way TMVP}} 17 \tag{4.2}$$

4.3 NTRUHPS4096821

For this parameter set, the polynomials to be multiplied lie in the ring

$$\mathbb{Z}_{4096}[x]/(x^{821} - 1).$$

In order to employ 5-way TMVP, different padding strategies are considered. Table 4.3 presents the number of operations for hybrid multiplication methods that are considered for the HPS4096821 version.

Table 4.3: Number of operations vs hybrid multiplication method for the HPS4096821

Method	#Addition Operations	#Multiplication Operations	#Total Operations
$825 \xrightarrow{\text{5-way}} 165 \xrightarrow{\text{5-way}} 33 \xrightarrow{\text{3-way}} 11$	155344	122694	278038
$840 \xrightarrow{\text{5-way}} 168 \xrightarrow{\text{4-way}} 42 \xrightarrow{\text{2-way}} 21$	169434	120393	289827
$840 \xrightarrow{\text{5-way}} 168 \xrightarrow{\text{4-way}} 42 \xrightarrow{\text{3-way}} 14$	160789	107016	267805
$840 \xrightarrow{\text{5-way}} 168 \xrightarrow{\text{4-way}} 42 \xrightarrow{\text{3-way}} 14 \xrightarrow{\text{2-way}} 7$	146593	80262	226855
$850 \xrightarrow{\text{5-way}} 170 \xrightarrow{\text{5-way}} 34 \xrightarrow{\text{2-way}} 17$	172921	146523	319444

Looking at Table 4.3, the only method that can be eliminated turns out to be padding up to 850. Therefore, all the methods shown in Table 4.3 are realized to see the fastest method. The results show that the fastest multiplication occurs when the method Eq. (4.3) is employed.

$$840 \xrightarrow{\text{5-way TMVP}} 168 \xrightarrow{\text{4-way TMVP}} 42 \xrightarrow{\text{2-way TMVP}} 21 \tag{4.3}$$

4.4 NTRUHRSS701

For this parameter set, the polynomials to be multiplied lie in the ring

$$\mathbb{Z}_{8192}[x]/(x^{701} - 1).$$

In order to employ 5-way TMVP, a few padding methods are examined. Table 4.4 presents the number of operations for hybrid multiplication methods that are examined for the HRSS701 version.

Table 4.4: Number of operations vs hybrid multiplication method for the HRSS701

Method	#Addition Operations	#Multiplication Operations	#Total Operations
$720 \xrightarrow{\text{5-way}} 144 \xrightarrow{\text{4-way}} 36 \xrightarrow{\text{2-way}} 18$	130476	88452	218928
$725 \xrightarrow{\text{5-way}} 145 \xrightarrow{\text{5-way}} 29$	157030	142129	299159
$750 \xrightarrow{\text{5-way}} 150 \xrightarrow{\text{5-way}} 30 \xrightarrow{\text{5-way}} 6$	127963	79092	207055
$750 \xrightarrow{\text{5-way}} 150 \xrightarrow{\text{5-way}} 30 \xrightarrow{\text{3-way}} 10$	131174	101400	232574
$750 \xrightarrow{\text{5-way}} 150 \xrightarrow{\text{5-way}} 30 \xrightarrow{\text{2-way}} 15$	137427	114075	251502

Table 4.4 suggests that the least number of operations is achieved when the padding is done all the way through 750 and $5-5-5$ TMVP configuration is employed. However, the implementation results show that the first configuration $(5-4-2)$ outperforms all the other competitors. This is because of the fact that modern processors handle schoolbook multiplication best when the size is close to 16 [37]. As a result, the chosen hybrid configuration for HRSS701 is the one presented in Eq. (4.4).

$$720 \xrightarrow{\text{5-way}} 144 \xrightarrow{\text{4-way}} 36 \xrightarrow{\text{2-way}} 18 \tag{4.4}$$

4.5 Implementation and Results

The multiplication algorithms are implemented using the C language without using AVX/AVX2 instructions. Thus, it is called C-reference implementation. The results are obtained by integrating the implemented hybrid multiplication algorithms to the reference implementation provided by the NTRU team via their GitHub project[1]. The software is compiled using make files provided in the GitHub project, and the resulting program is run on a computer with an Intel i3-12100F processor.

The timing results are presented in Table 4.5. The results for the Toom4 column stem from the execution of the code obtained from the GitHub project. This project employs a hybrid multiplication technique in order to multiply two polynomials: a 4-way Toom-Cook method followed by two layers of Karatsuba methods. Thus the column name Toom4 is used.

[1] The link to the GitHub project is `https://github.com/jschanck/ntru`

Table 4.5: Cycle counts comparison with NIST PQC 3rd round submission of NTRU[16]

Variant	Algorithm	Toom4[16]	This work	Speed-up
NTRU-HPS2048509	Key Generation	1326070	1040750	27%
	Encapsulation	128971	105322	22%
	Decapsulation	134416	70921	90%
NTRU-HPS2048677	Key Generation	2044569	1523904	34%
	Encapsulation	179637	141305	27%
	Decapsulation	194402	75514	157%
NTRU-HPS4096821	Key Generation	2755285	2472278	11%
	Encapsulation	213529	198014	8%
	Decapsulation	220813	162560	36%
NTRU-HRSS701	Key Generation	2096825	1751068	20%
	Encapsulation	107203	79549	35%
	Decapsulation	202758	118973	70%

As can be seen in Table 4.5, we achieved a significant speed-up in the execution times of all NTRU primitives. A further observation of the table shows that the speed-up of decapsulation is much higher than that of others. This is because there exist various kinds of time-consuming operations in key generation and encapsulation primitives, while the only time-consuming operation in decapsulation is the multiplication operation. As a result, the speed-up in key generation ranges from 11% to 34%, and in encapsulation ranges from 8% to 35%, while in decapsulation the speed-up ranges from 36% to 157%.

4.6 A Remark on Data Type Constraints

The source code provided in the GitHub project uses the `uint16_t` data type in order to hold the coefficients of polynomials. That is why all the implementations and tests regarding this study are conducted conforming to this preference. On the other hand, it is possible to use larger data types as the holder of the coefficients of polynomials, such as `uint32_t` and `uint64_t`, when the implementations are compiled for 32-bit or 64-bit processors. This is rather important because the size of the data type limits the number of 4-way TMVP steps that one can use in a hybrid method. Further insight into this limitation could be gained from [37]. One can see

examples of such algorithms as follows:

$$512 \xrightarrow{\text{4-way}} 128 \xrightarrow{\text{4-way}} 32 \xrightarrow{\text{4-way}} 8 \tag{4.5}$$

$$704 \xrightarrow{\text{4-way}} 176 \xrightarrow{\text{4-way}} 44 \xrightarrow{\text{4-way}} 11 \tag{4.6}$$

$$832 \xrightarrow{\text{4-way}} 208 \xrightarrow{\text{4-way}} 52 \xrightarrow{\text{4-way}} 13 \tag{4.7}$$

Indeed, the methods in Eqs. (4.5), (4.6), and (4.7) outperform their competitors described in the previous sections. However, as mentioned before, these methods are not applicable when the size of the data type is 16-bit or less. As a result, the application of the findings of this research on constrained devices can be a further research topic.

CHAPTER 5

CONCLUSION

With advancements in the area of quantum computing, post-quantum cryptography stands out as an aspect that is worth studying. First of all, studies like [2] and [41] discovered some problems that are hard even in the presence of a full-scale quantum computer. Then, NIST decided to perform the PQC Standardization Process [17], whose aim is to encourage studies on post-quantum cryptography and standardize the post-quantum public key cryptographic primitives. The process is still ongoing with very few finalists [4].

Most of the algorithms in the NIST PQC Standardization Process depend on hard lattice problems, such as Shortest Vector Problem, Learning with Error Problem, and Ring Learning with Error Problem. Some examples for such candidate algorithms are NTRU[16], Crystals-Dilithium [23], and Saber [48]. This also indicates that lattice-based cryptography is the top candidate for future public key cryptography. Therefore, the efficient algorithms and implementation related to LBC need to be researched in order to advance the practical use cases of LBC further.

In this thesis, we worked on the efficiency of polynomial multiplication in a ring of the form $R_q = \mathbb{Z}_q[x]/(x^n \pm 1)$, which is the operation that the efficiency of lattice-based schemes mostly depends on. There exist several polynomial multiplication techniques in the literature.These are Toom-Cook [19], NTT [20], and TMVP [13]. In the literature, various techniques are implemented in various situations. For example, the Toom-Cook method is employed in [48] in order to implement the multiplication required by the ring of Saber. Similarly, NTT is utilized in the algorithm details of Kyber [7]. Finally, instances of the employment of TMVP for multiplication in

lattices can be seen in [37].

Among these, we suggested TMVP for the multiplication of lattice elements. In addition, we formulated a new 5-way split TMVP technique in order to employ it in lattice-based schemes. The formulation is performed with the help of a transformation technique, which is described in [49]. This transformation technique paves the way to convert a degree-n polynomial multiplication to a size-5 Toeplitz-matrix vector product. Moreover, the total number of operations of the formulated technique turns out to be

$$M(n) = 13M(n/5) + (46n/5 - 18)A + 19n/5A_d.$$

After the formulization of 5-way split TMVP, we tested it against a commonly used multiplication algorithm, that is Toom-Cook. NTRU implementation suggested by [16] performs the multiplication operation using a Toom-Cook 4-way split followed by two levels of Karatsuba. We call this algorithm Toom4.

In order to test the suggested TMVP-5 algorithm, we replaced Toom4 with TMVP-5 in the NTRU implementation and ran both versions to see the differences in time. The results showed that we are able to improve the time efficiency of software NTRU implementations using this new technique. According to the benchmark results, the suggested algorithm executes faster than Toom4. The results indicate an increase in the execution speed. Quantitatively, we have speed-ups of

- 27% in key generation, 22% in encapsulation, and 90% in decapsulation algorithms of NTRU-HPS2048509,
- 34% in key generation, 27% in encapsulation, and 157% in decapsulation algorithms of NTRU-HPS2048677,
- 11% in key generation, 8% in encapsulation, and 36% in decapsulation algorithms of NTRU-HPS4096821, and
- 20% in key generation, 35% in encapsulation, and 70% in decapsulation algorithms of NTRU-HRSS701.

With an improvement in the execution time, the 5-way split TMVP algorithm looks promising for future implementations of lattice-based schemes.

REFERENCES

[1] J. Adikari, M. A. Hasan, and C. Negre, Towards faster and greener cryptoprocessor for eta pairing on supersingular elliptic curve over, in *International Conference on Selected Areas in Cryptography*, pp. 166–183, Springer, 2012.

[2] M. Ajtai, Generating hard instances of lattice problems, in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 99–108, 1996.

[3] M. Ajtai, R. Kumar, and D. Sivakumar, A sieve algorithm for the shortest lattice vector problem, in *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pp. 601–610, 2001.

[4] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, C. Miller, D. Moody, R. Peralta, et al., Status report on the third round of the NIST post-quantum cryptography standardization process, US Department of Commerce, NIST, 2022.

[5] S. Ali and M. Cenk, A New Algorithm for Residue Multiplication Modulo, in *International Conference on Information Security and Cryptology*, pp. 181–193, Springer, 2016.

[6] S. Ali and M. Cenk, Faster residue multiplication modulo 521-bit Mersenne prime and an application to ECC, IEEE Transactions on Circuits and Systems I: Regular Papers, 65(8), pp. 2477–2490, 2018.

[7] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, CRYSTALS-Kyber algorithm specifications and supporting documentation, NIST PQC Round, 2(4), pp. 1–43, 2019.

[8] A. Banerjee, C. Peikert, and A. Rosen, Pseudorandom functions and lattices, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 719–737, Springer, 2012.

[9] D. J. Bernstein, Curve25519: new Diffie-Hellman speed records, in *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9*, pp. 207–228, Springer, 2006.

[10] D. J. Bernstein, Batch binary edwards, in *Annual International Cryptology Conference*, pp. 317–336, Springer, 2009.

[11] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, High-speed NTT-based polynomial multiplication accelerator for post-quantum cryptography, in *2021 IEEE 28th Symposium on Computer Arithmetic (ARITH)*, pp. 94–101, IEEE, 2021.

[12] D. G. Cantor and E. Kaltofen, On fast multiplication of polynomials over arbitrary algebras, Acta Informatica, 28(7), pp. 693–701, 1991.

[13] M. Cenk, M. A. Hasan, and C. Negre, Efficient subquadratic space complexity binary polynomial multipliers based on block recombination, IEEE Transactions on Computers, 63(9), pp. 2273–2287, 2013.

[14] M. Cenk, C. Negre, and M. A. Hasan, Improved Three-Way Split Formulas for Binary Polynomial and Toeplitz Matrix Vector Products, Technical Report 30, CACR, Waterloo, ON, Canada, 2011, see http://www.cacr.math.uwaterloo.ca/techreports/2011/cacr2011-30.pdf.

[15] M. Cenk and F. Ozbudak, Improved polynomial multiplication formulas over if_2 using chinese remainder theorem, IEEE Transactions on computers, 58(4), pp. 572–576, 2008.

[16] C. Chen, O. Danba, J. Hoffstein, A. Hülsing, J. Rijneveld, J. M. Schanck, P. Schwabe, W. Whyte, and Z. Zhang, Algorithm Specifications And Supporting Documentation, Brown University and Onboard security company, Wilmington USA, 2019.

[17] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. A. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography*, volume 12, US Department of Commerce, National Institute of Standards and Technology . . . , 2016.

[18] I. L. Chuang and Y. Yamamoto, Simple quantum computer, Physical Review A, 52(5), pp. 3489–3496, nov 1995.

[19] S. A. Cook, *On the minimum computation time of functions*, Ph.D. thesis, 1966, uRL: `http://cr.yp.to/bib/entries.html#1966/cook`.

[20] R. de Sousa and A. Catarino, Description and implementation of number theoretic transforms, Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF, 1978.

[21] T. J. Dekker, A floating-point technique for extending the available precision, Numerische Mathematik, 18(3), pp. 224–242, 1971.

[22] W. Diffie and M. E. Hellman, New directions in cryptography, in *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pp. 365–390, 2022.

[23] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, Crystals-dilithium: A lattice-based digital signature scheme, IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 238–268, 2018.

[24] M. J. Dworkin, SHA-3 standard: Permutation-based hash and extendable-output functions, 2015.

[25] H. Fan and M. A. Hasan, A new approach to subquadratic space complexity parallel multipliers for extended binary fields, IEEE Transactions on Computers, 56(2), pp. 224–233, 2007.

[26] O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert, Approximating shortest lattice vectors is not harder than approximating closest lattice vectors, Information Processing Letters, 71(2), pp. 55–61, 1999.

[27] M. A. Hasan and C. Negre, Multiway splitting method for Toeplitz matrix vector product, IEEE Transactions on Computers, 62(7), pp. 1467–1471, 2012.

[28] K. Igeta and Y. Yamamoto, Quantum mechanical computers with single atom and photon fields, in *International Conference on Quantum Electronics*, p. TuI4, Optica Publishing Group, 1988.

[29] A. A. Karatsuba and Y. P. Ofman, Multiplication of many-digital numbers by automatic computers, in *Doklady Akademii Nauk*, volume 145, pp. 293–294, Russian Academy of Sciences, 1962.

[30] C. F. Kerry and P. D. Gallagher, Digital signature standard (DSS), FIPS PUB, pp. 186–4, 2013.

[31] V. Lyubashevsky and D. Micciancio, On bounded distance decoding, unique shortest vectors, and the minimum distance problem, in *Annual International Cryptology Conference*, pp. 577–594, Springer, 2009.

[32] V. Lyubashevsky, C. Peikert, and O. Regev, On ideal lattices and learning with errors over rings, in *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pp. 1–23, Springer, 2010.

[33] A. C. Mert, E. Öztürk, and E. Savaş, FPGA implementation of a run-time configurable NTT-based polynomial multiplication hardware, Microprocessors and Microsystems, 78, p. 103219, 2020.

[34] D. Micciancio, The hardness of the closest vector problem with preprocessing, IEEE Transactions on Information Theory, 47(3), pp. 1212–1215, 2001.

[35] D. Micciancio, The shortest vector in a lattice is hard to approximate to within some constant, SIAM journal on Computing, 30(6), pp. 2008–2035, 2001.

[36] D. Moody, G. Alagic, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, R. Peralta, R. Perlner, A. Robinson, D. Smith-Tone, and J. Alperin-Sheriff, Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process, 2020-07-22 2020.

[37] I. K. Paksoy and M. Cenk, Tmvp-based multiplication for polynomial quotient rings and application to saber on arm cortex-m4, Cryptology ePrint Archive, 2020.

[38] I. K. Paksoy and M. Cenk, Faster NTRU on ARM Cortex-M4 with TMVP-based multiplication, IEEE Transactions on Circuits and Systems I: Regular Papers, 69(10), pp. 4083–4092, 2022.

[39] C. Peikert, Public-key cryptosystems from the worst-case shortest vector problem, in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 333–342, 2009.

[40] J. M. Pollard, The fast fourier transform in a finite field, Mathematics of computation, 25(114), pp. 365–374, 1971.

[41] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, Journal of the ACM (JACM), 56(6), pp. 1–40, 2009.

[42] R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, 21(2), pp. 120–126, 1978.

[43] G. Seiler, Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography, Cryptology ePrint Archive, 2018.

[44] P. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, 1994.

[45] H. K. Taşkin and M. Cenk, Speeding up curve25519 using toeplitz matrix-vector multiplication, in *Proceedings of the Fifth Workshop on Cryptography and Security in Computing Systems*, pp. 1–6, 2018.

[46] A. L. Toom, The complexity of a scheme of functional elements realizing the multiplication of integers, in *Soviet Mathematics Doklady*, volume 3, pp. 714–716, 1963.

[47] A. Venkatesan and S. V. Kumar, Efficient implementation of fast convolution in ASIP, in *Recent Advances in Space Technology Services and Climate Change 2010 (RSTS & CC-2010)*, pp. 203–207, IEEE, 2010.

[48] I. F. Vercauteren, SABER: Mod-LWR Based KEM (Round 2 Submission).

[49] S. Winograd, *Arithmetic Complexity of Computations*, Society For Industrial & Applied Mathematics, U.S., 1980.