NEW EFFICIENT CHARACTERISTIC THREE POLYNOMIAL MULTIPLICATION ALGORITHMS AND THEIR APPLICATIONS TO NTRU PRIME

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

BY

ESRA YENİARAS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CRYPTOGRAPHY

JANUARY 2022

Approval of the thesis:

NEW EFFICIENT CHARACTERISTIC THREE POLYNOMIAL MULTIPLICATION ALGORITHMS AND THEIR APPLICATIONS TO NTRU PRIME

submitted by **ESRA YENİARAS** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Cryptography Department, Middle East Technical University** by,

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

Prof. Dr. Ferruh Özbudak _____
Head of Department, **Cryptography**

Assoc. Prof. Dr. Murat Cenk _____
Supervisor, **Cryptography, IAM, METU**

Examining Committee Members:

Assoc. Prof. Dr. Ali Doğanaksoy
Department of Mathematics, METU _____

Assoc. Prof. Dr. Murat Cenk
Cryptography, IAM, METU _____

Assoc. Prof. Dr. Barış Bülent Kırlar
Department of Mathematics, Süleyman Demirel University _____

Assist. Prof. Dr. Pelin Angın
Department of Computer Engineering, METU _____

Assist. Prof. Dr. A. Nurdan Saran
Department of Computer Engineering, Çankaya University _____

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: ESRA YENİARAS

Signature :

ABSTRACT

NEW EFFICIENT CHARACTERISTIC THREE POLYNOMIAL MULTIPLICATION ALGORITHMS AND THEIR APPLICATIONS TO NTRU PRIME

Yeniaras, Esra
Ph.D., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Murat Cenk

January 2022, 86 pages

Some of the post-quantum cryptographic protocols require polynomial multiplication in characteristic three fields, thus the efficiency of such multiplication algorithms gain more importance recently. In this thesis, we propose four new polynomial multiplication algorithms in characteristic three fields and we show that they are more efficient than the current state-of-the-art methods. We first analyze the well-known algorithms such as the schoolbook method, Karatsuba 2-way and 3-way split methods, Bernstein's three 3-way split method, Toom-Cook-like formulas, and other recent algorithms. We realize that there are not any 4-way or 5-way split multiplication algorithms in characteristic three fields unlike the binary (characteristic two) fields which have various 4, 5, or more split versions. We then propose three different 4-way split polynomial multiplication algorithms which are derived by using the interpolation technique in $\mathbb{F}_9$. Furthermore, we propose a new 5-way split polynomial multiplication algorithm and then compare the arithmetic complexities and the implementation results for all of the aforementioned methods. We show that the new 4-way and 5-way split algorithms provide a 48.6% reduction in the arithmetic complexity for multiplication over $\mathbb{F}_9$ and a 26.8% reduction for multiplication over $\mathbb{F}_3$ for the input size 1280. Moreover, the new 4-way and 5-way algorithms yield faster implementation results compared to the current state-of-the-art methods. We apply the

proposed methods to NTRU Prime protocol, a key encapsulation mechanism, submitted to NIST PQC Standardization Process by Bernstein *et al.*, which executes characteristic three polynomial multiplication in its decapsulation stage. We implement the new methods in C and observe a 26.85% speedup for `stnrup653` and a 35.52% speedup for `sntrup761` in the characteristic three polynomial multiplication step of the NTRU Prime decapsulation.

Keywords: Efficient polynomial multiplication, Characteristic three fields, Karatsuba, Interpolation, Post-quantum cryptography, Lattice-based cryptography, Key encapsulation mechanism, NTRU Prime

ÖZ

YENİ VERİMLİ KARAKTERİSTİK ÜÇ POLİNOM ÇARPIMI ALGORİTMALARI VE NTRU PRIME'A UYGULAMALARI

Yeniaras, Esra
Doktora, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Murat Cenk

Ocak 2022, 86 sayfa

Bazı kuantum-sonrası kriptografik protokoller karakteristik üç polinom çarpımı gerektirmektedir, böylece bu tarz çarpma algoritmalarının verimliği son zamanlarda daha çok önem kazanmaktadır. Bu tezde karakteristik üç cisimlerinde dört yeni polinom çarpımı algoritması tasarlanmıştır ve bunların şu dönemdeki en son yöntemlerden daha verimli oldukları gösterilmiştir. Öncelikle iyi bilinen okulkitabı yöntemi, Karatsuba 2-yönlü ve 3-yönlü ayrılmalı metotları, Bernstein'ın 3-yönlü ayrılmalı metodu, Toom-Cook benzeri formüller ve son zamanlardaki diğer algoritmalar analiz edilmiştir. Çeşitli 4, 5, veya fazla ayrılmalı algoritma versiyonlarına sahip olan ikili (karakteristik iki) cisimlerinden farklı olarak, karakteristik üç cisimlerinde hiç 4-yönlü veya 5-yönlü ayrılmalı çarpma algoritmalarının olmadığı farkedilmiştir. Sonrasında $\mathbb{F}_9$'da interpolasyon tekniği kullanılarak geliştirilen üç farklı 4-yönlü ayrılmalı polinom çarpımı algoritması tasarlanmıştır. Dahası, yeni bir 5-yönlü ayrılmalı polinom çarpımı algoritması tasarlanmış ve sonrasında tüm bahsigeçen yöntemlerin aritmetik karmaşıklığı ve implementasyon sonuçları karşılaştırılmıştır. Yeni 4-yönlü ve 5-yönlü ayrılmalı algoritmaların, 1280 girdi boyutu için, $\mathbb{F}_9$ üzerindeki çarpmaların aritmetik karmaşıklığında 48.6% ve $\mathbb{F}_3$ üzerindeki çarpmaların aritmetik karmaşıklığında da 26.8% oranında düşüş sağladığı gösterilmiştir. Dahası, yeni 4-yönlü ve 5-yönlü ayrılmalı algoritmalar şu dönemdeki en son yöntemlere kıyasla daha hızlı implementasyon sonuçları sağlamıştır. Tasarlanan metotlar, bir anahtar kapsülleme

mekanizması olarak Bernstein *et al.* tarafından NIST PQC Standartlaştırma Süreci'ne sunulan ve kapsülden çıkarma aşamasında karakeristik üç polinom çarpımı gerektiren, NTRU Prime protokolüne uygulanmıştır. Yeni metotların C'de implementasyonu yapılmış ve NTRU Prime anahtar çıkarmasındaki karakteristik üç polinom çarpımı adımında `strup653` için 26.85% 'lik bir hızlanma ve `strup761` için de 35.52% 'lik bir hızlanma gözlenmiştir.

Anahtar Kelimeler: Verimli polinom çarpımı, karakteristik üç cisimleri, Karatsuba, İnterpolasyon, Kuantum-sonrası kriptografi, Kafes-tabanlı kriptografi, Anahtar kapsülleme mekanizması, NTRU Prime

To Esin,

ACKNOWLEDGMENTS

First of all, I would like to thank my thesis advisor Murat Cenk for his patience, motivation, and perfect academic guidance during my doctoral studies. He has always been understanding and helpful in any difficulties I face. I owe him a lot for his endless support. I would also like to thank the lattice-based post-quantum cryptography team friends Hasan Bartu Yünüak, Yusuf Alper Bilgin, İrem Keskinkurt Paksoy and all other friends in the graduate school for sharing ideas and giving advice.

Although we could not finalize our studies studies with him because of the circumstances, I would like to give my special thanks to once my co-advisor Oğuz Yayla for his brilliant research ideas, suggestions, and time. I would also not forget to thank Ersan Akyıldız, Ali Doğanaksoy, Ferruh Özbudak and Muhiddin Uğuz for their valueble contributions, teaching us the most complicated topics in cryptography and being always there to support the students.

Thanks to the founders, all of the academicians, and the staff of the Institute of Applied Mathematics for providing us with this research environment to improve ourselves in our scientific and academic journey. Especially, thanks to Nejla Erdoğdu, Serkan Demiröz, and Ebru Gündoğdu for being the nicest secretaries an institute could ever have. They were always helpful, patient, and understanding.

Also, I would like to thank my office colleagues Eyüp Korkut, Necmi Ercan, Okan Çelik, Haydar Şelbeş, Kazım Gümüş, and our directors Erdoğan Gündoğdu, Zekai Yılmaz and Abdullah Saruhan at my workplace in the Ministry of Education for their support and understanding my absence during the hard times with the studies towards this PhD degree.

I would also like to express my deepest gratitude to my mom Nermin Yeniaras and my dad Orhan Yeniaras for their life-long love, motivation, and support in every discouraging situation. Without their moral support in the hardest struggles, I would not be able to finalize it. Also, I am grateful to my brother Erol Yeniaras for being a role model in academic and professional life. Even though he is thousands of miles away physically, he has always been there to support me in every situation.

Also, I should not forget my second-hand 2001 model car, "yeşil canavar", since it has never broken down during my rush in the work-school-home-hospitals-shopping cycle for the last five years and providing me with the best buddy wheel-shoulder to cry and/or smile. Finally, thanks to "Drago" for how hard he fought, I will always remember him with a light and smile on my face.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

$\mathbb{Z}^+$	The set of positive integers
$\mathbb{F}_3$	Finite field with 3 elements
$\mathbb{F}_9$	Finite field with 9 elements
$\mathbb{F}_q$	Finite field with q elements
ω	$\omega \in \mathbb{F}_9$ with $\omega^2 + 1 = 0$
$M_{q,\oplus}(n)$	Number of additions over $\mathbb{F}_q$ to multiply two polynomials of degree $n-1$.
$M_{q,\otimes}(n)$	Number of multiplications over $\mathbb{F}_q$ to multiply two polynomials of degree $n-1$.
$M_q(n)$	Total number of operations (additions/multiplications) over $\mathbb{F}_q$ to multiply two polynomials of degree $n-1$, in other words, $M_q(n) = M_{q,\oplus}(n) + M_{q,\otimes}(n)$.
`sntrup653`	Streamlined NTRU Prime Protocol for the parameter 653
`sntrup761`	Streamlined NTRU Prime Protocol for the parameter 761
TLS 1.3	Transport Layer Security Version 1.3
CECPO2	Combined Elliptic Curve and Post-Quantum 2, a quantum- secure modification to TLS 1.3 developed by Google
PQC	Post-Quantum Cryptography
NIST	National Institute for Standarts and Technology
KEM	Key Encapsulation Mechanism
RSA	Rivest-Shamir-Adleman
IFP	Integer Factorization Problem
DLP	Discrete Logarithm Problem
ECDH	Elliptic Curve Diffie-Hellman

CHAPTER 1

INTRODUCTION

With the recent advances in quantum computers [24, 30] some of the hardest mathematical problems such as the Integer Factorization Problem (IFP) [10] and the Discrete Logarithm Problem (DLP) [33, 34], which are the basic underlying hard problems for the most prevalent cryptographic protocols, are believed to be able to solved. Quantum factoring algorithm [35, 36], defined by Shor, makes the most widespread asymmetric cryptographic protocols such as Diffie-Hellman [20, 25], Rivest-Shamir-Adleman (RSA) [32], and Elliptic Curve Diffie-Hellman (ECDH) [1, 2, 19, 27, 31] vulnerable to attacks by sufficiently powerful quantum computers. Also, Grover's algorithm [22] reduces the search spaces for private keys significantly, causing the protocols to be more vulnerable to brute force attacks. Most of the sensitive data such as web pages, e-mails, bank accounts, health records, and personal social media accounts are protected by the aforementioned algorithms and any kind of vulnerability in these systems could have serious consequences. Therefore, researchers focus on developing secure protocols which are resistant to quantum attacks. By this means, NIST organizes a PQC Standardization Process [28] to which various quantum-resistant protocols are submitted for the competition. NTRU Prime [6–9] protocol by Bernstein *et al.* which competes in the NIST PQC Standardization Process is advanced to the third round of the competition as an alternative candidate [3,9].

We realize that there exist some cryptographic protocols, including the quantum-resistant ones, that require characteristic three polynomial multiplications but yet there are only 2-way split or 3-way split polynomial multiplication algorithms in the characteristics three fields unlike the case over the binary fields. Lack of more

efficient multiplication methods in characteristic three fields can cause slowness in the run-times of these cryptographic protocols. Motivated by this fact, we construct new efficient 4-way and 5-way split polynomial multiplication algorithms which are specific to the characteristic three fields. Then, we observe the possible improvements with the help of these new 4-way and 5-way algorithms on the efficiency of Streamlined NTRU Prime KEM [6–9] since it executes a polynomial multiplication in $\mathbb{Z}_3[x]/(x^p - x - 1)$ for a prime integer p in its decapsulation stage. Remember that, while renovating the TLS 1.3 to a quantum-resistant form, Google-Couldfire CECPQ2 experiment integrates `ntruhrss701` into it. Then, NTRU Prime KEM is equipped with a batch key generation feature by Bernstein *et al.* and a more secure and faster alternative to `ntruhrss701` for TLS 1.3 is proposed in 2021 [5]. In the improvement of the efficiency of such protocols, faster characteristic three polynomial multiplication algorithms play an important role. Therefore, our aim in this thesis is to develop more efficient characteristic three polynomial multiplication algorithms than the current-state-of-the-art methods, and as an application, observe the efficiency gain in quantum-resistant NTRU Prime protocol.

In 2018, a 3-way split characteristic three polynomial multiplication algorithm A3 [17] is proposed by Cenk, Hasan, and Zadeh using interpolation technique which is similar to Toom-Cook's method [13, 16, 38]. A3 algorithm was more efficient than the other known methods at that time, such as the schoolbook method [4], refined Karatsuba 2-way method [4], improved Karatsuba 3-way method [26, 38, 41] and the unbalanced refined Karatsuba 2-way [4] method.

Then in 2020, we introduce two new 4-way split algorithms N1, N2, and a 5-way split algorithm V1 in [40]. N1, N2, and V1 are all more efficient than the A3 algorithm. The proposed N1, N2, and V1 algorithms outperform the other known algorithms in terms of both arithmetic complexity and implementation perspectives. N1, N2, and V1 algorithms provide a 40.35% reduction in arithmetic complexity for polynomial multiplication over $\mathbb{F}_9$ and a 16.61% reduction for polynomial multiplication over $\mathbb{F}_3$ for the input size $n = 1024$. We then apply various hybrid uses of N1, N2, and V1 algorithms [40] combined with the other known methods to NTRU Prime decapsulation for which Bernstein *et al.* propose a hybrid use of 5 levels of refined Karatsuba 2-way algorithm and 1 level of schoolbook algorithm in [7]. For the input size 761,

we name the hybrid algorithm by Bernstein *et al.* as Hybrid-1 and the hybrid algorithm that we propose as N1-Hybrid2. We observe that with the N1-Hybrid2 [40] algorithm, the reduction percentage for the implementation performance is 37.39% compared to Bernstein's Hybrid-1 [7] method. When it comes to the arithmetic complexity, the N1-Hybrid2 [40] is around 38% more efficient than Bernstein's Hybrid-1 algorithm. Therefore, N1-Hybrid2 can be a better alternative for the characteristic three polynomial multiplication in NTRU Prime decapsulation for the version of the protocol described in [7].

In 2021, by using the method of including the special points x and $x + 1$ in the evaluation step of the interpolation approach [4, 5], a new 3-way algorithm [5] for polynomial multiplication over $\mathbb{F}_3$, which we call B1, is proposed by Bernstein *et al.* B1 is more efficient than the A3 3-way algorithm over $\mathbb{F}_3$ but less efficient than it over $\mathbb{F}_9$. They use a hybrid version of B1, in the characteristic three polynomial multiplication step of the decapsulation phase of Streamlined NTRU Prime protocol in [5]. The hybrid approach defined by Bernstein *et al.* suggests using the B1 algorithm once and then refined Karatsuba 2-way algorithm down to the input size 16, and then the schoolbook method at the final level. We call these algorithms that are constructed according to Bernstein's approach B1-Hybrid1 for the input size $n = 653$ and B1-Hybrid2 for the input size $n = 761$ with zero-padded inputs when necessary.

Later in 2021, combining the techniques in [4, 5, 17], we propose another improved 4-way algorithm N3 in [39] and the unbalanced split version of the V1 5-way algorithm [40], which we name as U1 [39], for characteristic three polynomial multiplication. The N3 4-way algorithm is even better than the N1 and the N2 4-way methods in terms of arithmetic complexity. N3 also has a faster implementation runtime than Bernstein's B1 3-way algorithm. We report that U1 provides the fastest results of all aforementioned methods above. For instance, the new 4-way algorithm N3 and the unbalanced 5-way algorithm U1 together with the other known ones, yield a 48.6% decrease in the arithmetic complexity for polynomial multiplication over $\mathbb{F}_9$ and 26.8% decrease in the arithmetic complexity for polynomial multiplication over $\mathbb{F}_3$, for the input size $n = 1280$. Then, we apply the hybrid use of the U1 algorithm combined with the other known methods in the NTRU Prime decapsulation. We introduce the hybrid algorithms U1-Hybrid1 for input size $n = 653$ and U1-Hybrid2

for input size $n = 761$, by recursively calling U1 once and then calling some variants of Karatsuba-2 way, A3, B1, and the schoolbook methods. We then compare these new hybrid algorithms U1-Hybrid1 and U1-Hybrid2 to Bernstein's B1-Hybrid1 and B1-Hybrid1 methods and we obtain a 26.8% speedup for `sntrup653` [5, 9] and a 35.52% speedup for the `sntrup761` [5, 9] in the implementation run-time. Also, note that comparing the U1-Hybrid2 method with N1-Hybrid2 for the input size $n = 761$ the implementation improvement that comes with U1-Hybrid2 is 8.56% better than the implementation improvement that comes with the N1-Hybrid2 method in the characteristic three polynomial multiplication step of the Streamlined NTRU Prime decapsulation.

Organization of the thesis In chapter 2, we introduce the notation and the preliminaries that are used throughout this thesis. Chapter 3 gives a comprehensive description and analysis of the well-known characteristic three polynomial multiplication algorithms. Chapter 4 is reserved for introducing the new 4-way split characteristic three polynomial multiplication algorithms N1, N2, and N3. Then, in chapter 5 we introduce the new 5-way split characteristic three polynomial multiplication algorithm V1 and its unbalanced version which we call U1. Chapter 6 is kept for the comparative arithmetic complexity analysis and implementation run-times of the proposed algorithms with other well-known ones. In chapter 7, we introduce the hybrid use of the proposed methods and their applications into the characteristic three polynomial multiplication step of the Streamlined NTRU Prime decapsulation and compare the implementation results. We then concluded the thesis in chapter 8.

Software availability The software for all of the hybrid algorithms that are introduced in this thesis is available online at: `https://github.com/cryptoarith/F3Mul` and `https://github.com/esrayeniaras/NTRUPrimePolyMultF3`

CHAPTER 2

PRELIMINARIES

In this chapter, we introduce the notation and the preliminaries that are used throughout this thesis. In that sense, we give a presentation of the NTRU Prime Key Encapsulation Mechanism (KEM). Then, we give a brief remark regarding the calculation of the arithmetic complexities of the recursive algorithms. Also, it is assumed that all of the polynomials to be multiplied are from the characteristic three fields unless otherwise specified.

2.1 Notation

- ✓ SB: Schoolbook polynomial multiplication method [4].
- ✓ KA2: Refined Karatsuba 2-way polynomial multiplication method [4].
- ✓ A3: A 3-way split polynomial multiplication method, first described in [17].
- ✓ A2: Another polynomial multiplication method over $\mathbb{F}_9$, introduced in [17].
- ✓ UB: Unbalanced refined Karatsuba polynomial multiplication method [4] for odd values of n, where n is the input size.
- ✓ LT: Schoolbook recursion method [4]. We refer to it as the last term method.
- ✓ B1: A 3-way split polynomial multiplication method, first described in [4, 5].
- ✓ N1: A 4-way polynomial multiplication method, described in this thesis [40].
- ✓ N2: An improved 4-way polynomial multiplication method, described in this thesis [40].

- ✓ N3: Another improved 4-way polynomial multiplication method, described in this thesis [39].

- ✓ V1: A 5-way polynomial multiplication method, described in this thesis [40].

- ✓ U1: An unbalanced split 5-way polynomial multiplication method, described in this thesis [39].

- ✓ Assume that Δ is one of the polynomial multiplication methods listed above, then $\Delta\mathbb{F}_3$ and $\Delta\mathbb{F}_9$ are abbreviations for polynomial multiplications using the Δ method over $\mathbb{F}_3$ and $\mathbb{F}_9$ respectively.

- ✓ $M_{q,\oplus}(n)$: Number of additions over $\mathbb{F}_q$ to multiply two polynomials of degree $n-1$.

- ✓ $M_{q,\otimes}(n)$: Number of multiplications over $\mathbb{F}_q$ to multiply two polynomials of degree $n-1$.

- ✓ $M_q(n)$: Total number of operations (additions/multiplications) over $\mathbb{F}_q$ to multiply two polynomials of degree $n-1$, in other words, $M_q(n) = M_{q,\oplus}(n) + M_{q,\otimes}(n)$.

2.2 NTRU Prime Key Encapsulation Mechanism

NTRU Prime [6–9] by Bernstein *et al.*, is a lattice-based, quantum-resistant key encapsulation protocol submitted and advanced through Rounds 1-3 of the NIST Post-Quantum Standardization Process [3, 28]. There are two components of the NTRU Prime, one is Streamlined NTRU Prime and the other one is NTRU LPRime. In the decapsulation phase, the Streamlined NTRU Prime Key Encapsulation Mechanism performs multiplication in $\mathbb{Z}_3[x]/(x^p - x - 1)$ for a prime p, so that we can apply the proposed multiplication methods to it. Below, we explain the details of the protocol.

2.2.1 Parameters

The parameters of Streamlined NTRU Prime are given as (p, q, ω) with prime p and q, where $q \geq 17$, $0 < \omega \leq p$, $2p \geq 3\omega$, $q \geq 16\omega + 1$. Also, we assume that $(x^p - x - 1)$ is irreducible in $\mathbb{Z}_q[x]$. The rings $\mathbb{Z}[x]/(x^p - x - 1)$, $\mathbb{Z}_3[x]/(x^p - x - 1)$ and the field $\mathbb{Z}_q[x]/(x^p - x - 1)$ are abbreviated as $\mathcal{R}$, $\mathcal{R}/3$, and $\mathcal{R}/q$ respectively. Note that, when $p = 761$, $q = 4591$ and $\omega = 286$ the cryptosystem is abbreviated as `sntrup761` and when $p = 653$, $q = 4621$ and $\omega = 288$ the cryptosystem is abbreviated as `sntrup653`. A comprehensive description of the NTRU Prime protocol for all three-round submissions to the NIST PQC Standardization process can be found in [6, 8, 9]. One can observe from Algorithm 3 that the decapsulation stage executes a characteristic three polynomial multiplication at step 5.

2.2.2 Key Generation, Encapsulation, and Decapsulation

Key generation, encapsulation, and decapsulation algorithms are given by Algorithm 1, 2, and 3 respectively,followed by some necessary definitions that are used throughout the pseudocodes.

Algorithm 1 Streamlined NTRU Prime Key Generation KeyGen()

1: **Output:** (P_k, S_k)
2: **do**
3: $\quad$ g $\xleftarrow{\$}$ **small**
4: **while** $g^{-1} \notin \mathcal{R}/3$
5: $f \leftarrow$**short**
6: $h \leftarrow g/(3f) \in \mathcal{R}/q$
7: $\underline{\text{h}} \leftarrow$ Encode(h)
8: $\underline{\text{k}} \leftarrow$ Encode$((f, 1/g))$
9: $p \leftarrow$**short**
10: **return** $(P_k, S_k) = (\underline{\text{h}}, (\underline{\text{k}}, \underline{\text{h}}, p))$

Algorithm 2 Streamlined NTRU Prime Key Encapsulation Encap($\underline{\text{h}}$)

Input: $P_k = \underline{\text{h}}$
Output: $C = (C, \text{HashSession}(1, \underline{\text{r}}, C))$
$h \leftarrow \text{Decode}(\underline{\text{h}})$
r $\xleftarrow{\$}$**short**
$\underline{\text{r}} \leftarrow \text{Encode}(r)$
$c \leftarrow \text{Round}(h.r) \in \mathcal{R}$
$\underline{\text{c}} \leftarrow \text{Encode}(c)$
$C \leftarrow (\underline{\text{c}}, \text{HashConfirm}(\underline{\text{r}}, \underline{\text{h}}))$
return $C = (C, \text{HashSession}(1, \underline{\text{r}}, C))$

Algorithm 3 Streamlined NTRU Prime Decapsulation Decap(C, S_k)

Input: (C, S_k)
Output: $\text{HashSession}(1, \underline{\text{r}}, C)$ or $\text{HashSession}(0, p, C)$
$c \leftarrow \text{Decode}(\underline{\text{c}})$
$e \leftarrow (\text{Rounded}(c.(3f)) \bmod 3) \in \mathcal{R}/3$
$r' \leftarrow \text{Lift}(e.g^{-1}) \in \mathcal{R}/q$
$c' \leftarrow \text{Round}(h.r')$
$\underline{\text{c}}' \leftarrow \text{Encode}(c')$
$C' \leftarrow (\underline{\text{c}}', \text{HashConfirm}(\underline{\text{r}}', \underline{\text{h}}))$
if $C' == C$ **then**
    return $\text{HashSession}(1, \underline{\text{r}}, C)$
else
    return $\text{HashSession}(0, p, C))$
end if

- **Small** is described as a type of polynomial with all of its coefficients are in $\{-1, 0, 1\}$.
- **Short** is defined as a small, **weight**$-\omega$ polynomial of $\mathcal{R}$. A **weight**$-\omega$ polynomial is a polynomial with exactly ω of its coefficients are nonzero.
- **Rounded()** is defined as a function that takes any polynomial in $\mathcal{R}/q$ to a rounded polynomial, i.e., $a_0 + a_1x + ... + a_{p-1}x^{p-1} \in \mathcal{R}$ such that, if $q \in 1 + 3\mathbb{Z}$ then for each i, $a_i \in \{-(q-1)/2, ..., -6, -3, 0, 3, 6, ..., (q-1)/2\}$ and if

$q \in 2+3\mathbb{Z}$ then for each i, $a_i \in \{-(q+1)/2, ..., -6, -3, 0, 3, 6, ..., (q+1)/2\}$.

- **Round()** is a function that takes any polynomial in $\mathcal{R}/q$ to rounded polynomial then takes its coefficients to the nearest multiple of 3, producing a polynomial in $\mathcal{R}$.
- **Lift()** is a function that maps any polynomial of $\mathcal{R}/3$ to a small polynomial in $\mathcal{R}/q$ by simply reducing it modulo q.
- **Encode()/Decode()** Let $M = (m_0, ..., m_{n-1})$ and $R = (r_0, ..., r_{n-1})$ be sequences of integers and assume that for each i, $0 \leq r_i \leq m_i \leq 2^{14}$. Then $S =$ Encode(R, M) is a sequence of bytes and Decode(Encode$(R, M), M) = R$. In other words, Encode() converts the ring elements to strings and Decode() converts the strings to the ring elements vice versa. One can refer to [8] for the algorithmic details of the encoding and decoding parameters.
- **Hashing** Hash(z) is defined as the first 32 bits of SHA-512(z). Hash$_b$ for $b \in \{0, 1, ..., 255\}$ is Hash with the input prefixed by one byte value b, i.e., Hash$(b, z) =$ Hash$_b(z)$. Another function HashConfirm$(r, \underline{\text{h}})$ is defined as Hash$_2$(Hash$_3(r)$, Hash$_4(h)$). Moreover the function HashSession(b, r, C) is defined as Hash$_b$(Hash$_3(r), C)$ for $b \in \{0, 1\}$ [8].

2.3 Calculation of the Arithmetic Complexity for the Recursive Algorithms

Note that, since x^2+1 is an irreducible polynomial over $\mathbb{F}_3$ then $\mathbb{F}_9 \cong \mathbb{F}_3[x]/(x^2+1)$, thus we can represent the elements of $\mathbb{F}_9$ as polynomials of degree less than 2. Let's define $\omega \in \mathbb{F}_9$ such that $\omega^2 + 1 = 0$.

Table 2.1: Comparison of basic operations, $a, b, c, d \in \mathbb{F}_3$

Operation	$\mathbb{F}_3$ cost	$\mathbb{F}_9$ cost
$(a + b\omega) + (c + d\omega) = (a + c) + (b + d) \cdot \omega$	2 Adds	1 Add
$(a + b\omega) \cdot (c + d\omega) = (ac - bd) + (bc + ad) \cdot \omega$	2 Adds+4 Mults	1 Mult
$\omega \cdot a$, $1 \cdot a$, $(-1) \cdot a$	0	0
$\omega \cdot (a + b\omega) = -b + a\omega$	0	0

For $a, b, c, d \in \mathbb{F}_3$, one can convey from Table 2.1 that one addition in $\mathbb{F}_9$ corresponds to two additions in $\mathbb{F}_3$ whereas one multiplication corresponds to two additions and four multiplications in it. We also note that multiplication of an element in $\mathbb{F}_9$ by ω,

1, or -1 is cost-free.

The following formula presents a quick result of the *Master Theorem* [17, 18] which is used to calculate the arithmetic complexity of recursive algorithms.

Remark 1 Let $M(n)$ be a recursive algorithm, $a, b, \mu \in \mathbb{Z}$, $n = b^{\mu}$, $a \neq 1$, $a, b, \mu > 0$ and $M(1) = e$ such that,

$$M(n) = aM(n/b) + cn + d + fn^{\kappa}$$

(i) If $a \neq b$ and $f = 0$ then the associated complexity is given by,

$$M(n) = \left(e + \frac{bc}{a-b} + \frac{d}{a-1}\right) n^{log_b a} - \frac{bc}{a-b}n + \frac{d}{a-1}$$

(ii) If $a = b$, then $M(n)$ is given by,

$$M(n) = \frac{fb^{\kappa}}{b^{\kappa}-a}n^{\kappa} + \left(e - \frac{fb^{\kappa}}{b^{\kappa}-a} + \frac{d}{a-1}\right) n + cn \log_b n - \frac{d}{a-1}$$

(iii) If $a \neq b$, then the associated complexity is given by,

$$M(n) = \frac{fb^{\kappa}}{b^{\kappa}-a}n^{\kappa} + \left(e + \frac{bc}{a-b} - \frac{fb^{\kappa}}{b^{\kappa}-a} + \frac{d}{a-1}\right) n^{log_b a} - \left(\frac{bc}{a-b}\right) n - \frac{d}{a-1}$$

One can refer to [14, 17, 21] for the details of the proof.

CHAPTER 3

SOME KNOWN POLYNOMIAL MULTIPLICATION ALGORITHMS AND RECENT IMPROVEMENTS

In this section, we summarize the well-known polynomial multiplication algorithms and their improved versions in characteristic three fields. We will examine 2-way split algorithms, 3-way split ones, and the other types of algorithms in each section, respectively. We also assume that the polynomial-size n is a power of 2 for 2-way split algorithms and a power of 3 for 3-way split ones unless otherwise specified.

3.1 Schoolbook Algorithm

Let $A(x)$ and $B(x)$ be defined as follows:

$$\left.\begin{aligned} A(x) &= a_0 + a_1x + a_2x^2 + ... + a_{n-1}x^{n-1} \\ B(x) &= b_0 + b_1x + b_2x^2 + ... + b_{n-1}x^{n-1} \end{aligned}\right\} \tag{3.1}$$

The schoolbook algorithm computes the coefficients of the product

$$C(x) = A(x).B(x) = \sum_{i=0}^{2n-2} c_i x^i$$

as $c_i = \sum_{j+k=i}^{2n-2} a_j b_k$ where $0 \leq j, k < n$. Thus, the arithmetic complexity of the schoolbook algorithm can be given as follows,

$$\left.\begin{aligned} M_3(n) &= 2n^2 - 2n + 1 \\ M_{3,\oplus}(n) &= (n-1)^2 \\ M_{3,\otimes}(n) &= n^2 \end{aligned}\right\} \tag{3.2}$$

3.2 Karatsuba 2-Way Split Algorithm

Let $A(x)$ and $B(x)$ be degree $2n-1$ polynomials for some $n \geq 1$, $n \in \mathbb{Z}^+$. Also let's define $y = x^n$ and $C(x) = A(x)B(x)$. 2-way multiplication methods basically depend on dividing the polynomials into two equivalent parts and perform the multiplication recursively on these equally half-size parts as below:

$$\left.\begin{aligned} A_0 &= a_0 + a_1 x + ... + a_{n-1}x^{n-1} \\ A_1 &= a_n + a_{n+1}x + ... + a_{2n-1}x^{n-1} \\ B_0 &= b_0 + b_1 x + ... + b_{n-1}x^{n-1} \\ B_1 &= b_n + b_{n+1}x + ... + b_{2n-1}x^{n-1} \end{aligned}\right\} \tag{3.3}$$

then

$$\left.\begin{aligned} A(x) &= A_0 + yA_1 \\ B(x) &= B_0 + yB_1 \end{aligned}\right\} \tag{3.4}$$

and the multiplication becomes,

$$A(x)B(x) = (A_0 + yA_1)(B_0 + yB_1)$$

so that it can be done with the half-size polynomials using Karatsuba 2-way algorithm as follows:

Let's define the products of the half-sized polynomials as P_0, P_1 and P_2 then,

$$\left.\begin{aligned} P_0 &= A_0 B_0 \\ P_1 &= (A_0 + A_1)(B_0 + B_1) \\ P_2 &= A_1 B_1 \end{aligned}\right\} \tag{3.5}$$

and the final result of the multiplication $C(x) = A(x)B(x)$ can be found as,

$$C(x) = P_0 + (P_1 - P_0 - P_2)x^n + P_2 x^{2n} \tag{3.6}$$

The associated complexity of this recursive algorithm can be given by:

$$\left.\begin{aligned} M_3(2n) &\leq 3M_3(n) + 8n - 4, M_3(1) = 1 \\ M_{3,\otimes}(2n) &\leq 3M_{3,\otimes}(n), M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(2n) &\leq 3M_{3,\oplus}(n) + 8n - 4, M_{3,\oplus}(1) = 0 \end{aligned}\right\} \tag{3.7}$$

Using Remark 1, we can explicitly calculate the complexity as follows,

$$\left.\begin{aligned} M_3(n) &\leq 7n^{\log_2 3} - 8n + 2 \\ M_{3,\otimes}(n) &\leq n^{\log_2 3} \\ M_{3,\oplus}(n) &\leq 6n^{\log_2 3} - 8n + 2 \end{aligned}\right\} \tag{3.8}$$

Moreover, assuming that $A(x)$ and $B(x)$ are degree $n + k - 1$ polynomials where $1 \leq k \leq n$, A_0, B_0 are degree $n - 1$ polynomials and A_1, B_1 are degree $k - 1$ polynomials for $n/2 \leq k$ and $n \geq 2$, then the cost analysis of the Karatsuba 2-way algorithm yields,

$$M_3(n + k) \leq 2M_3(n) + M_3(k) + 4n + 4k - 4 \tag{3.9}$$

3.3 Improved (Refined) Karatsuba 2-Way Split Algorithm (KA2)

Bernstein [4] introduced an improved version of the Karatsuba 2-way algorithm by using the same settings given in (3.3)-(3.5) but following a different approach in the rest, more precisely this time $C(x)$ is defined as,

$$C(x) = (y - 1)(yP_2 - P_0) + yP_1 \tag{3.10}$$

then the complexity for the improved (refined) Karatsuba 2-way algorithm (KA2) can be found as:

$$\left.\begin{aligned} M_3(2n) &\leq 3M_3(n) + 7n - 3, M_3(1) = 1 \\ M_{3,\otimes}(2n) &\leq 3M_{3,\otimes}(n), M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(2n) &\leq 3M_{3,\oplus}(n) + 7n - 3, M_{3,\oplus}(1) = 0 \end{aligned}\right\} \tag{3.11}$$

by Remark 1 we get,

$$\left.\begin{aligned} M_3(n) &\leq 6.5n^{\log_2 3} - 7n + 1.5 \\ M_{3,\otimes}(n) &\leq n^{\log_2 3} \\ M_{3,\oplus}(n) &\leq 5.5n^{\log_2 3} - 7n + 1.5 \end{aligned}\right\} \tag{3.12}$$

Moreover, assuming that $A(x)$ and $B(x)$ are degree $n + k - 1$ polynomials where $1 \leq k \leq n$, A_0, B_0 are degree $n - 1$ polynomials and A_1, B_1 are degree $k - 1$ polynomials, where $n/2 \leq k$ and $n \geq 2$. Then the cost analysis of the refined [4] Karatsuba gives,

$$M_3(n + k) \leq 2M_3(n) + M_3(k) + 3n + 4k - 3 \tag{3.13}$$

As we compare the arithmetic complexities in (3.8) and (3.12), we can observe that, the improved version of Karatsuba 2-way algorithm is approximately 7% more efficient than the previous one.

3.4 Unbalanced Refined Karatsuba 2-way Algorithm (UB)

This method [4] can be used when the input size is of the form $2n - 1$ for $n \geq 2$, i.e., it is odd. Since the 2-way split can also be done within two *unequal* parts, an odd value would fit in this situation. Assume that $A(x)$ and $B(x)$ are two degree $2n - 2$ polynomials for $n \in \mathbb{Z}^+$, instead of equal ones, we divide $A(x)$ and $B(x)$ into two unequal pieces, as follows:

$$\left.\begin{aligned} A_0 &= a_0 + a_1x + ... + a_{n-1}x^{n-1} \\ A_1 &= a_n + a_{n+1}x + ... + a_{2n-2}x^{n-2} \\ B_0 &= b_0 + b_1x + ... + b_{n-1}x^{n-1} \\ B_1 &= b_n + b_{n+1}x + ... + b_{2n-2}x^{n-2} \end{aligned}\right\} \tag{3.14}$$

One can observe that the polynomial A_0 contains one more element than the polynomial A_1. By using (3.5) and (3.10), we get the following complexity for the unbal-

anced refined Karatsuba 2-way algorithm UB:

$$\left.\begin{aligned} M_3(2n-1) &= 2M_3(n) + M_3(n-1) + 7n - 7, M_3(1) = 1 \\ M_{3,\otimes}(n) &= 2M_{3,\otimes}(n) + M_{3,\otimes}(n-1), M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(n) &= 2M_{3,\oplus}(n) + M_{3,\oplus}(n-1) + 7n - 7, M_{3,\oplus}(1) = 0 \end{aligned}\right\} \tag{3.15}$$

Remark 2 Note that, from (3.5) that the highest degree coefficients of both P_0 and P_1 are the same, thus one multiplication gets cost-free for the UB algorithm.

3.5 Schoolbook Recursion or the Last Term Method (LT)

This algorithm [4] is the recursive version of the classical schoolbook algorithm. Assume that $A(x)$ and $B(x)$ are defined as in (3.1), we can write $A(x)$ and $B(x)$ as follows:

$$\left.\begin{aligned} A(x) &= A_{n-2}(x) + a_{n-1}x^{n-1} \\ B(x) &= B_{n-2}(x) + b_{n-1}x^{n-1} \end{aligned}\right\} \tag{3.16}$$

where

$$\left.\begin{aligned} A_{n-2}(x) &= a_0 + a_1x + a_2x^2 + ... + a_{n-2}x^{n-2} \\ B_{n-2}(x) &= b_0 + b_1x + b_2x^2 + ... + b_{n-2}x^{n-2} \end{aligned}\right\} \tag{3.17}$$

then the algorithm recursively perform the following multiplication

$$\left.\begin{aligned} A(x)B(x) = A_{n-2}(x)B_{n-2}(x) &+ b_{n-1}x^{n-1}A_{n-2}(x) \\ &+ a_{n-1}x^{n-1}B_{n-2}(x) + a_{n-1}b_{n-1}x^{2n-2} \end{aligned}\right\} \tag{3.18}$$

Note that, $A_{n-2}(x)$ and $B_{n-2}(x)$ are obtained by deleting the last terms of $A(x)$ and $B(x)$. Therefore, for the schoolbook recursion, we use the abbreviation LT referring to these last terms.

The complexity of the LT algorithm is,

$$\begin{aligned} M_3(n) &\leq M_3(n-1) + 4n - 4, M_3(1) = 1 \\ M_{3,\otimes}(n) &\leq M_{3,\otimes}(n-1) + (2n-1), M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(n) &\leq M_{3,\oplus}(n-1) + (2n-3), M_{3,\oplus}(1) = 0 \end{aligned} \tag{3.19}$$

3.6 Karatsuba Like 3-way Algorithm

This algorithm [37] uses a 3-way approach for multiplying polynomials. Let $A(x)$ and $B(x)$ be two degree $3n-1$ polynomials for some $n \geq 2$, $n \in \mathbb{Z}^+$, $y = x^n$ and $C(x) = A(x)B(x)$. We divide both $A(x)$ and $B(x)$ into three parts as follows:

$$\left.\begin{aligned} A_0 &= a_0 + a_1 x + ... + a_{n-1}x^{n-1} \\ A_1 &= a_n + a_{n+1}x + ... + a_{2n-1}x^{n-1} \\ A_2 &= a_{2n} + a_{2n+1}x + ... + a_{3n-1}x^{n-1} \\ B_0 &= b_0 + b_1 x + ... + b_{n-1}x^{n-1} \\ B_1 &= b_n + b_{n+1}x + ... + b_{2n-1}x^{n-1} \\ B_2 &= b_{2n} + b_{2n+1}x + ... + b_{3n-1}x^{n-1} \end{aligned}\right\} \tag{3.20}$$

then

$$\left.\begin{aligned} A(x) &= A_0 + yA_1 + y^2 A_2 \\ B(x) &= B_0 + yB_1 + y^2 B_2 \end{aligned}\right\} \tag{3.21}$$

thus the recursive multiplication becomes

$$A(x)B(x) = (A_0 + yA_1 + y^2 A_2)(B_0 + yB_1 + y^2 B_2)$$

with six 1/3 sized products as given below,

$$\left.\begin{aligned} P_0 &= A_0 B_0 \\ P_1 &= A_1 B_1 \\ P_2 &= A_2 B_2 \\ P_3 &= (A_0 + A_1)(B_0 + B_1) \\ P_4 &= (A_0 + A_2)(B_0 + B_2) \\ P_5 &= (A_1 + A_2)(B_1 + B_2) \end{aligned}\right\} \tag{3.22}$$

With the help of the Chinese Remainder Theorem [38] and the methods described in [26] and [37] the result becomes,

$$\begin{aligned} C(x) = P_0 &+ (P_3 - P_0 - P_1)x^n + (P_4 + P_1 - P_0 - P_2)x^{2n} \\ &+ (P_5 - P_1 - P_2)x^{3n} + P_2 x^{4n} \end{aligned} \tag{3.23}$$

this algorithm is associated with the following complexity:

$$\left.\begin{aligned} M_3(3n) &\leq 6M_3(n) + 24n - 11, M_3(1) = 1 \\ M_{3,\otimes}(3n) &\leq 6M_{3,\otimes}(n), M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(3n) &\leq 6M_{3,\oplus}(n) + 24n - 11, M_{3,\oplus}(1) = 0 \end{aligned}\right\} \tag{3.24}$$

by Remark 1 we get,

$$\left.\begin{aligned} M_3(n) &\leq 6.8n^{\log_3 6} - 8n + 2.2 \\ M_{3,\otimes}(n) &\leq n^{\log_3 6} \\ M_{3,\oplus}(n) &\leq 5.8n^{\log_3 6} - 8n + 2.2 \end{aligned}\right\} \tag{3.25}$$

Moreover, assuming that $A(x)$ and $B(x)$ are degree $2n + k - 1$ polynomials where $1 \leq k \leq n$, A_0, A_1, B_0, B_1 are degree $n - 1$ polynomials and A_2, B_2 are degree $k - 1$ polynomials for $n/2 \leq k$ and $n \geq 2$. Then the cost analysis of the Karatsuba like 3-way algorithm yields,

$$M_3(2n + k) \leq 5M_3(n) + M_3(k) + 14n + 4k - 11 \tag{3.26}$$

3.7 Improved Karatsuba Like 3-way Algorithm

By reconstructing the Karatsuba like 3-way algorithm as described in [41], we get the improved Karatsuba 3-way algorithm. Let $A(x)$ and $B(x)$ be two degree $3n-1$ polynomials for some $n \geq 2$, $n \in \mathbb{Z}^+$ both of which are divided into three parts as described in (3.20)-(3.21) such that $C(x) = A(x)B(x)$. Also let, $P_i = P_{iL} + x^n P_{iH}$ for $0 \leq i \leq 5$ such that the degree of each P_{iL} is $n-1$, the degree of each P_{iH} is $n-2$, and the degree of each P_i is $2n-2$. Inserting these values in (3.22) we get the following result:

$$\left.\begin{aligned} C = {} & P_{0L} + x^n(P_{0H} - P_{0L} - P_{1L} + P_{3L}) \\ & + x^{2n}(-P_{0L} - P_{0H} + P_{1L} - P_{1H} - P_{2L} + P_{3H} + P_{4L}) \\ & + x^{3n}(-P_{0H} + P_{1H} - P_{2L} - P_{2H} + P_{4H} + P_{5L} - P_{1L}) \\ & + x^{4n}(-P_{1H} + P_{2L} - P_{2H} + P_{5H}) + x^{5n}P_{2H} \end{aligned}\right\} \tag{3.27}$$

and the associated complexity for the algorithm is given by,

$$\left.\begin{aligned} M_3(3n) &\leq 6M_3(n) + 22n - 9, M_3(1) = 1 \\ M_{3,\otimes}(3n) &\leq 6M_{3,\otimes}(n), M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(3n) &\leq 6M_{3,\oplus}(n) + 22n - 9, M_{3,\oplus}(1) = 0 \end{aligned}\right\} \tag{3.28}$$

thus by Remark 1, we get,

$$\left.\begin{aligned} M_3(n) &\leq 6.53n^{\log_3 6} - 7.33n + 1.8 \\ M_{3,\otimes}(n) &\leq n^{\log_3 6} \\ M_{3,\oplus}(n) &\leq 5.53n^{\log_3 6} - 7.33n + 1.8 \end{aligned}\right\} \tag{3.29}$$

Moreover assuming that $A(x)$ and $B(x)$ are degree $2n+k-1$ polynomials where $1 \leq k \leq n$, $k > n/2$, A_0, A_1, B_0, B_1 are degree $n-1$ polynomials, and A_2, B_2 are degree $k-1$ polynomials. Then the cost analysis of the improved Karatsuba 3-way algorithm yeilds,

$$M_3(2n+k) \leq 5M_3(n) + M_3(k) + 12n + 6k - 6 \tag{3.30}$$

Note that, the complexity given by (3.29) is approximately 4% less than the one in (3.25), which shows that the improved Karatsuba 3-way algorithm is more arithmetically efficient than the original version.

3.8 3-way Algorithm with Five Multiplications (A1)

A1, a 3-way algorithm with five multiplications, is first described in [17]. It is derived by the interpolation method and has similar results to that of Toom-Cook's Formula [13–16, 38]. Remember that $\mathbb{F}_9 \cong \mathbb{F}_3[x]/(x^2+1)$ and since $\mathbb{F}_3$ does not have enough points for the interpolation method with five multiplications, we borrow the element ω from $\mathbb{F}_9$ such that $\omega^2 = -1$. Let $A(x)$ and $B(x)$ be two polynomials described as in (3.20)-(3.21) such that $C(x) = A(x)B(x)$. Then evaluating $C(x) = A(x)B(x)$ at $\{0, 1, -1, \omega, \infty\}$ gives us the following system of linear equations in $\mathbb{F}_9$,

$$x = 0 \Rightarrow P_0 = A_0.B_0 = C_0$$

$$x = 1 \Rightarrow P_1 = (A_0 + A_1 + A_2).(B_0 + B_1 + B_2) = C_0 + C_1 + ... + C_4$$

$$x = -1 \Rightarrow P_2 = (A_0 - A_1 + A_2).(B_0 - B_1 + B_2) = C_0 - C_1 + C_2 + ... + C_4$$

$$x = \omega \Rightarrow P_3 = (A_0 + A_1\omega - A_2).(B_0 + B_1\omega - B_2) = C_0 + C_1\omega + ... + C_4$$

$$x = \infty \Rightarrow P_4 = A_2.B_2 = C_4$$

Solving this equation system yields,

$$\left.\begin{aligned} C_0 &= P_0 \\ C_1 &= (P_1 - P_2) - (-P_0 + P_1 + P_2 - P_3 - P_4)\omega \\ C_2 &= -(P_0 + P_1 + P_2 + P_4) \\ C_3 &= (P_1 - P_2) + (-P_0 + P_1 + P_2 - P_3 - P_4)\omega \\ C_4 &= P_4 \end{aligned}\right\} \tag{3.31}$$

where,

$$C(x) = C_0 + C_1x^n + C_2x^{2n} + C_3x^{3n} + C_4x^{4n}$$

The arithmetic complexity associated with this algorithm in $\mathbb{F}_9$ is,

$$\left.\begin{aligned} M_9(3n) &\leq 5M_9(n) + 60n - 24, M_9(1) = 6 \\ M_{9,\otimes}(3n) &\leq 5M_{9,\otimes}(n), M_{9,\otimes}(1) = 4 \\ M_{9,\oplus}(3n) &\leq 5M_{9,\oplus}(n) + 60n - 24, M_{9,\oplus}(1) = 2 \end{aligned}\right\} \tag{3.32}$$

Applying Remark 1 we get,

$$\left.\begin{aligned} M_9(n) &\leq 30n^{\log_3 5} - 30n + 6 \\ M_{9,\otimes}(n) &\leq 4n^{\log_3 5} \\ M_{9,\oplus}(n) &\leq 26n^{\log_3 5} - 30n + 6 \end{aligned}\right\} \tag{3.33}$$

The arithmetic complexity associated with this algorithm in $\mathbb{F}_3$ is,

$$\left.\begin{aligned} M_3(3n) &\leq 4M_3(n) + M_9(n) + 24n - 10, M_3(1) = 1 \\ M_{3,\otimes}(3n) &\leq 4M_{3,\otimes}(n) + M_{9,\otimes}(n), M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(3n) &\leq 4M_{3,\oplus}(n) + M_{9,\oplus}(n) + 24n - 10, M_{3,\oplus}(1) = 0 \end{aligned}\right\} \tag{3.34}$$

by Remark 1,

$$\left.\begin{aligned} M_3(n) &\leq 30n^{\log_3 5} - 36.33n^{\log_3 4} + 6n + 1.33 \\ M_{3,\otimes}(n) &\leq 4n^{\log_3 5} - 3n^{\log_3 4} \\ M_{3,\oplus}(n) &\leq 26n^{\log_3 5} - 33.33n^{\log_3 4} + 6n + 1.33 \end{aligned}\right\} \tag{3.35}$$

Moreover, assuming that $A(x)$ and $B(x)$ are degree $2n + k - 1$ polynomials where $1 \leq k \leq n$, A_0, A_1, B_0, B_1 are degree $n - 1$ polynomials and A_2, B_2 are degree $k - 1$ polynomials. Then, the cost analysis of the A1 3-way algorithm in Table A.1 (See Appendix A) yields,

$$\left.\begin{aligned} M_3(2n + k) &\leq 3M_3(n) + M_3(k) + M_9(n) + 18n + 6k - 10 \\ M_9(2n + k) &\leq 4M_9(n) + M_9(k) + 48n + 12k - 24 \end{aligned}\right\} \tag{3.36}$$

Algorithm A1 is more efficient than the Karatsuba 3-way improved algorithm after $n = 729$. The percentage of cost reduction increases significantly when the polynomial-size increases. For $n = 3^7$ the reduction is 4%, for $n = 3^8$ the reduction is 14% and for $n = 3^{10}$ the reduction reaches up to 55%.

3.9 Improved 3-way Algorithm with Five Multiplications (A3)

A3 algorithm, which is an improved version of A1, is also a 3-way algorithm [17] with five multiplications. Let $A(x)$ and $B(x)$ be two polynomials described as in (3.20)-(3.21) such that $C(x) = A(x)B(x)$. If we switch the interpolation points of A1 from $\{0, 1, -1, \omega, \infty\}$ to $\{0, 1, \omega, -\omega, \infty\}$ then we get,

$x = 0 \Rightarrow P_0 = A_0.B_0 = C_0$

$x = 1 \Rightarrow P_1 = (A_0 + A_1 + A_2).(B_0 + B_1 + B_2) = C_0 + C_1 + ... + C_4$

$x = \omega \Rightarrow P_2 = (A_0 + A_1\omega - A_2).(B_0 + B_1\omega - B_2) = C_0 + C_1\omega + ... + C_4$

$x = -\omega \Rightarrow P_3 = (A_0 - A_1\omega - A_2).(B_0 - B_1\omega - B_2) = C_0 - C_1\omega - ... + C_4$

$x = \infty \Rightarrow P_4 = A_2.B_2 = C_4$

Assume that $P_2 = P_{2,0} + \omega P_{2,1}$ and $P_3 = P_{3,0} + \omega P_{3,1}$ then one can observe that $P_{2,0} = P_{3,0}$ and $P_{2,1} = -P_{3,1}$. Therefore, once we compute P_2 we do not need to compute P_3. By using these equalities we get the following formula for $C(x)$,

$$\left.\begin{aligned} C_0 &= P_0 \\ C_1 &= -P_0 - P_1 - P_{2,0} - P_4 - P_{2,1} \\ C_2 &= P_0 - P_{2,0} + P_4 \\ C_3 &= -P_0 - P_1 - P_{2,0} - P_4 + P_{2,1} \\ C_4 &= P_4 \end{aligned}\right\} \tag{3.37}$$

where,

$$C(x) = C_0 + C_1x^n + C_2x^{2n} + C_3x^{3n} + C_4x^{4n}$$

The complexity associated with A3 in $\mathbb{F}_9[x]$ is exactly the same as the complexity of A1 in $\mathbb{F}_9[x]$. On the other hand, the complexity of A3 in $\mathbb{F}_3[x]$ is given by (3.38)-(3.39)

and it is less costly than that of A1.

$$\left.\begin{aligned} M_3(3n) &\leq 3M_3(n) + M_9(n) + 22n - 10, M_3(1) = 1 \\ M_{3,\otimes}(3n) &\leq 3M_{3,\otimes}(n) + M_{9,\otimes}(n), M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(3n) &\leq 3M_{3,\oplus}(n) + M_{9,\oplus}(n) + 22n - 10, M_{3,\oplus}(1) = 0 \end{aligned}\right\} \tag{3.38}$$

again by Remark 1,

$$\left.\begin{aligned} M_3(n) &\leq 15n^{\log_3 5} - 2.66n\log_3 n - 16n + 2 \\ M_{3,\otimes}(n) &\leq 2n^{\log_3 5} - n \\ M_{3,\oplus}(n) &\leq 13n^{\log_3 5} - 2.66n\log_3 n - 15n + 2 \end{aligned}\right\} \tag{3.39}$$

Moreover, assuming that $A(x)$ and $B(x)$ are degree $2n + k - 1$ polynomials where $1 \leq k \leq n$, A_0, A_1, B_0, B_1 are degree $n-1$ polynomials and A_2, B_2 are degree $k-1$ polynomials. Then the cost analysis of the A3 3-way algorithm in Table A.2 (See Appendix A) yields,

$$\left.\begin{aligned} M_3(2n + k) &\leq 2M_3(n) + M_3(k) + M_9(n) + 16n + 6k - 10 \\ M_9(2n + k) &\leq 4M_9(n) + M_9(k) + 48n + 12k - 24 \end{aligned}\right\} \tag{3.40}$$

A3 algorithm outperforms KA2 when $n \geq 400$. For $n = 709$ the percentage of reduction in arithmetic complexity is 7%.

3.10 Bernstein's 3-way Algorithm (B1)

Bernstein *et al.* proposed a 3-way algorithm in [5] using Lagrange interpolation and multi-evaluation for the five elements $\{0, 1, -1, x, \infty\}$ of $\mathcal{R} \cup \{\infty\}$ where $\mathcal{R} = \mathbb{F}_3[x]$. Let $A(x)$ and $B(x)$ are both in $\mathbb{F}_3[x]$ are two degree $3n-1$ polynomials and $y = x^n$. Also let $A_0, A_1, A_2, B_0, B_1, B_2$ are defined as is in (3.20)-(3.21). Then, we get the

products of the evaluations of $A(y)$ and $B(y)$ as follows:

$$\left.\begin{aligned} P_0 &= A_0 B_0 \\ P_1 &= (A_0 + A_1 + A_2)(B_0 + B_1 + B_2) \\ P_2 &= (A_0 - A_1 + A_2)(B_0 - B_1 + B_2) \\ P_3 &= (A_0 + A_1 x + A_2 x^2)(B_0 + B_1 x + B_2 x^2) \\ P_4 &= A_2 B_2 = C_4 \end{aligned}\right\} \tag{3.41}$$

By using Lagrange Interpolation, the formula for $C(x)$ can be found as,

$$\left.\begin{aligned} V &= [(P_1 + P_2).x + (P_1 - P_2) + P_3/x]/(x^2 - 1) \\ U &= V + P_0/x - P_4.x \\ C &= P_0 - [U + (P_1 - P_2)].x^n - [P_0 + (P_1 + P_2) + P_4)].x^{2n} + U.x^{3n} + P_4.x^{4n} \end{aligned}\right\} \tag{3.42}$$

One can refer to Table A.3 for the multi-evaluation and reconstruction cost analysis of B1 (See Appendix A). The formula in (3.42) can be applied only once at a time, since the five products in (3.41) simultaneously involve polynomials of degree $n-1$ and $n+1$. In order to have a fully recursive method, we express the product of degree $n+1$ polynomials in terms of one product of degree $n-1$ polynomials plus some additional non-recursive computations as below:

$$\left.\begin{aligned} M_3(n+2) &= M_3(n) + 8n + 4 \\ M_{3,\otimes}(n+2) &= M_{3,\otimes}(n) + 4n + 4 \\ M_{3,\oplus}(n+2) &= M_{3,\oplus}(n) + 4n \\ M_9(n+2) &= M_9(n) + 32n + 24 \\ M_{9,\otimes}(n+2) &= M_{9,\otimes}(n) + 16n + 16 \\ M_{9,\oplus}(n+2) &= M_{9,\oplus}(n) + 16n + 8 \end{aligned}\right\} \tag{3.43}$$

Then, we calculate the recursive and associated asymptotic complexities as follows:

$$\left.\begin{aligned} M_9(3n) &\leq 5M_9(n) + 104n - 10, M_9(1) = 6 \\ M_{9,\otimes}(3n) &\leq 5M_9(n) + 88n - 26, M_{9,\otimes}(1) = 4 \\ M_{9,\oplus}(3n) &\leq 5M_{9,\oplus}(n) + 16n + 16, M_{9,\oplus}(1) = 2 \\ M_3(3n) &\leq 5M_3(n) + 44n - 13, M_3(1) = 0 \\ M_{3,\otimes}(3n) &\leq 5M_{3,\otimes}(n) + 4n + 4, M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(3n) &\leq 5M_{3,\oplus}(n) + 40n - 17, M_{3,\oplus}(1) = 0 \end{aligned}\right\} \tag{3.44}$$

$$\left.\begin{aligned} M_9(n) &\leq 55.5n^{\log_3 5} - 52n - 2.5 \\ M_{9,\otimes}(n) &\leq 16n^{\log_3 5} - 8n + 4 \\ M_{9,\oplus}(n) &\leq 39.5n^{\log_3 5} - 44n - 6.5 \\ M_3(n) &\leq 19.75n^{\log_3 5} - 22n - 3.25 \\ M_{3,\otimes}(n) &\leq 4n^{\log_3 5} - 2n + 1 \\ M_{3,\oplus}(n) &\leq 15.75n^{\log_3 5} - 20n - 4.25 \end{aligned}\right\} \tag{3.45}$$

Moreover, assuming that $A(x)$ and $B(x)$ are degree $2n + k - 1$ polynomials where $1 \leq k \leq n - 2$, A_0, A_1, B_0, B_1 are degree $n - 1$ polynomials and A_2, B_2 are degree $k - 1$ polynomials and $n/2 \leq k$. Then, the cost analysis of the B1 3-way algorithm in Table A.3 (See Appendix A) yields,

$$\left.\begin{aligned} M_3(2n + k) &\leq 4M_3(n) + M_3(k) + 36n + 8k - 18 \\ M_9(2n + k) &\leq 4M_9(n) + M_9(k) + 88n + 16k - 20 \end{aligned}\right\} \tag{3.46}$$

Observe from (3.44) that B1 has five products all of which are performed in $\mathbb{F}_3$, whereas A3 has only four products in $\mathbb{F}_3$ and one product in $\mathbb{F}_9$ as can be seen in (3.38). This results in the B1 algorithm being more efficient than A3 in $\mathbb{F}_3$. However, when it comes to performing multiplication in $\mathbb{F}_9$, A3 is more efficient than B1 because one of the A3 multiplications gets cost-free due to the fact that P_2 can be calculated from P_3 since $P_{2,0} = P_{3,0}$ and $P_{2,1} = -P_{3,1}$. This fact is justified by Table 6.2 and Table 6.3 which indicates that in terms of the implementation cycle counts, B1 is faster than A3 in $\mathbb{F}_3$ but it is slower than it in $\mathbb{F}_9$.

3.11 Another Multiplication Algorithm (A2)

As mentioned in [17], when the coefficients of the polynomials are in $\mathbb{F}_9$, we can use another algorithm called A2. Let $A(x)$ and $B(x)$ are both in $\mathbb{F}_9[x]$ then we can rewrite $A(x)$ and $B(x)$ as sums of their w parts and $w - free$ parts. Let $A_0,\ A_1,\ B_0,\ B_1$ be degree $n-1$ polynomials in $\mathbb{F}_3[x]$ such that,

$$\left.\begin{aligned} A &= A_0 + \omega A_1 \\ B &= B_0 + \omega B_1 \end{aligned}\right\} \tag{3.47}$$

then,

$$\begin{aligned} AB =& (A_0 + A_1\omega)(B_0 + B_1\omega) = A_0B_0 - A_1B_1 \\ &+ ((A_0 + A_1)(B_0 + B_1) - A_0B_0 - A_1B_1)\omega \end{aligned} \tag{3.48}$$

the complexity of the A2 algorithm can be found as,

$$M_9(n) \leq 3M_3(n) + 8n - 3,\ M_9(1) = 6,\ M_3(1) = 1 \tag{3.49}$$

Through the sections 3.1 - 3.11, we have examined eleven of the recent well-known algorithms that can be used for polynomial multiplication in characteristic three fields. Now we can describe our new 4-way and 5-way split formulas that are more efficient than all of the algorithms we have discussed above.

CHAPTER 4

NEW 4-WAY SPLIT POLYNOMIAL MULTIPLICATION ALGORITHMS

In this chapter, we propose three new 4-way split multiplication algorithms N1, N2, and N3 which are all derived by using the interpolation method in $\mathbb{F}_9[x]$.

4.1 New 4-way Multiplication Algorithm (N1)

N1 is a 4-way algorithm with seven 1/4 sized multiplications. Assume that,

$$\left.\begin{aligned} A(x) &= a_0 + a_1x + a_2x^2 + ... + a_{4n-1}x^{4n-1} \\ B(x) &= b_0 + b_1x + b_2x^2 + ... + b_{4n-1}x^{4n-1} \end{aligned}\right\} \tag{4.1}$$

are two polynomials of degree $4n-1$ where $n = 4^k$ for some $k \geq 0$. Let $y = x^n$, $C(x) = A(x)B(x)$ and,

$$\left.\begin{aligned} A_0 &= a_0 + a_1x + ... + a_{n-1}x^{n-1} \\ A_1 &= a_n + a_{n+1}x + ... + a_{2n-1}x^{n-1} \\ A_2 &= a_{2n} + a_{2n+1}x + ... + a_{3n-1}x^{n-1} \\ A_3 &= a_{3n} + a_{3n+1}x + ... + a_{4n-1}x^{n-1} \\ B_0 &= b_0 + b_1x + ... + b_{n-1}x^{n-1} \\ B_1 &= b_n + b_{n+1}x + ... + b_{2n-1}x^{n-1} \\ B_2 &= b_{2n} + b_{2n+1}x + ... + b_{3n-1}x^{n-1} \\ B_3 &= b_{3n} + b_{3n+1}x + ... + b_{4n-1}x^{n-1} \end{aligned}\right\} \tag{4.2}$$

then,

$$\left.\begin{aligned} A(x) &= A_0 + yA_1 + y^2A_2 + y^3A_3 \\ B(x) &= B_0 + yB_1 + y^2B_2 + y^3B_3 \end{aligned}\right\} \tag{4.3}$$

thus, the result of the multiplication becomes,

$$\left.\begin{aligned} C(x) &= (A_0 + yA_1 + y^2A_2 + y^3A_3)(B_0 + yB_1 + y^2B_2 + y^3B_3) \\ &= C_0 + C_1y + C_2y^2 + C_3y^3 + C_4y^4 + C_5y^5 + C_6y^6 \end{aligned}\right\} \tag{4.4}$$

In order to compute the coefficients of $C(x)$ we need seven evaluation points. Actually, we can select any seven points from the elements of $\mathbb{F}_9$ together with ∞. In this section we use the method in Section 3.9 that is, the conjugate points generate similar products and once we obtain one of them, the other one is obtained easily. So we choose six points from $\mathbb{F}_9$ that are pairwisely conjugate elements. We use $\{\omega, -\omega, \omega+1, -\omega+1, -\omega-1, \omega-1, \infty\}$ as the points of evaluation for interpolation and we get the following system of equations,

$$\left.\begin{aligned} P_0 &= [(A_0 - A_2) + \omega(A_1 - A_3)].[(B_0 - B_2) + \omega(B_1 - B_3)] = C(\omega) \\ P_1 &= [(A_0 - A_2) - \omega(A_1 - A_3)].[(B_0 - B_2) - \omega(B_1 - B_3)] = C(-\omega) \\ P_2 &= [(A_0 + A_1 + A_3) + \omega(A_1 - A_2 - A_3)].[(B_0 + B_1 + B_3) + \omega(B_1 - B_2 - B_3)] = C(\omega+1) \\ P_3 &= [(A_0 + A_1 + A_3) + \omega(-A_1 + A_2 + A_3)].[(B_0 + B_1 + B_3) + \omega(-B_1 + B_2 + B_3)] = C(-\omega+1) \\ P_4 &= [(A_0 - A_1 - A_3) + \omega(-A_1 - A_2 + A_3)].[(B_0 - B_1 - B_3) + \omega(-B_1 - B_2 + B_3)] = C(-\omega-1) \\ P_5 &= [(A_0 - A_1 - A_3) + \omega(A_1 + A_2 - A_3)].[(B_0 - B_1 - B_3) + \omega(B_1 + B_2 - B_3)] = C(\omega-1) \\ P_6 &= A_3.B_3 = C_6 \end{aligned}\right\} \tag{4.5}$$

Let,

$$\left.\begin{aligned} P_0 &= P_{0,0} + \omega P_{0,1} \\ P_1 &= P_{1,0} + \omega P_{1,1} \\ P_2 &= P_{2,0} + \omega P_{2,1} \\ P_3 &= P_{3,0} + \omega P_{3,1} \\ P_4 &= P_{4,0} + \omega P_{4,1} \\ P_5 &= P_{5,0} + \omega P_{5,1} \end{aligned}\right\} \tag{4.6}$$

then one can observe that,

$$\left.\begin{aligned} P_{0,0} &= P_{1,0} \\ P_{0,1} &= -P_{1,1} \\ P_{2,0} &= P_{3,0} \\ P_{2,1} &= -P_{3,1} \\ P_{4,0} &= P_{5,0} \\ P_{4,1} &= -P_{5,1} \end{aligned}\right\} \tag{4.7}$$

Notice that with the help of (4.7), we avoid three unnecessary multiplications, i.e., instead of calculating the six P_i for $0 \leq i \leq 5$ multiplications, it will be sufficient to calculate P_0, P_2, and P_4. Thus, three multiplications in $\mathbb{F}_9[x]$ get cost-free.

Using the interpolation technique we get the following results for the N1 Algorithm:

$$\left.\begin{aligned} C_0 &= P_0 + P_1 - P_2 - P_3 - P_4 - P_5 + P_6 + \omega(-P_2 + P_3 - P_4 + P_5) \\ C_1 &= -P_2 - P_3 + P_4 + P_5 + \omega(-P_0 + P_1) \\ C_2 &= P_6 + \omega(P_2 - P_3 + P_4 - P_5) \\ C_3 &= -P_2 - P_3 + P_4 + P_5 + \omega(-P_2 + P_3 + P_4 - P_5) \\ C_4 &= P_0 + P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + \omega(-P_2 + P_3 - P_4 + P_5) \\ C_5 &= \omega(-P_0 + P_1 - P_2 + P_3 + P_4 - P_5) \\ C_6 &= P_6 \end{aligned}\right\} \tag{4.8}$$

Inserting the equalities of (4.6)-(4.7) into (4.8) we get,

$$\left.\begin{aligned} C_0 &= -P_{0,0} + P_{2,0} + P_{4,0} + P_6 - P_{2,1} - P_{4,1} \\ C_1 &= P_{2,0} - P_{4,0} - P_{0,1} \\ C_2 &= P_6 + P_{2,1} + P_{4,1} \\ C_3 &= P_{2,0} - P_{4,0} - P_{2,1} + P_{4,1} \\ C_4 &= -P_{0,0} - P_{2,0} - P_{4,0} + P_6 - P_{2,1} - P_{4,1} \\ C_5 &= -P_{0,1} - P_{2,1} + P_{4,1} \\ C_6 &= P_6 \end{aligned}\right\} \tag{4.9}$$

Table 4.1 and Table 4.2 demonstrate the costs of multi evaluation and reconstruction for the N1 algorithm in $\mathbb{F}_9[x]$ and $\mathbb{F}_3[x]$ respectively. From these tables, we calculate the associated complexities of N1 algorithm for $\mathbb{F}_3[x]$ and $\mathbb{F}_9[x]$ as follows:

$$\left.\begin{aligned} M_9(4n) &\leq 7M_9(n) + 144n - 52, M_9(1) = 6 \\ M_{9,\otimes}(4n) &\leq 7M_{9,\otimes}(n), M_{9,\otimes}(1) = 4 \\ M_{9,\oplus}(4n) &\leq 7M_{9,\oplus}(n) + 144n - 52, M_{9,\oplus}(1) = 2 \\ M_3(4n) &\leq M_3(n) + 3M_9(n) + 44n - 18, M_3(1) = 1 \\ M_{3,\otimes}(4n) &\leq M_{3,\otimes}(n) + 3M_{9,\otimes}(n), M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(4n) &\leq M_{3,\oplus}(n) + 3M_{9,\oplus}(n) + 44n - 18, M_{3,\oplus}(1) = 0 \end{aligned}\right\} \tag{4.10}$$

then we get the explicit complexities as follows,

$$\left.\begin{aligned} M_9(n) &\leq 45.33n^{\log_4 7} - 48n - 8.66 \\ M_{9,\otimes}(n) &\leq 4n^{\log_4 7} \\ M_{9,\oplus}(n) &\leq 41.33n^{\log_4 7} - 48n - 8.66 \\ M_3(n) &\leq 22.66n^{\log_4 7} - 33.33n - 44\log_4 n + 11.66 \\ M_{3,\otimes}(n) &\leq 2n^{\log_4 7} - 1 \\ M_{3,\oplus}(n) &\leq 20.66n^{\log_4 7} - 33.33n - 44log_4 n + 12.66 \end{aligned}\right\} \tag{4.11}$$

Moreover, assuming that $A(x)$ and $B(x)$ are degree $3n + k - 1$ polynomials where $1 \leq k \leq n$, $A_0, A_1, A_2, B_0, B_1, B_2$ are degree $n - 1$ polynomials and A_3, B_3 are degree $k - 1$ polynomials. Then, the cost analysis of the N1 4-way algorithm in Table A.4 and Table A.5 (See Appendix A) yield,

$$\left.\begin{aligned} M_3(3n + k) &\leq M_3(k) + 3M_9(n) + 36n + 8k - 18 \\ M_9(3n + k) &\leq 6M_9(n) + M_9(k) + 124n + 20k - 52 \end{aligned}\right\} \tag{4.12}$$

N1 algorithm is less costly than KA2 for $n \geq 280$ in $\mathbb{F}_3[x]$ and for $n \geq 28$ in $\mathbb{F}_9[x]$. Also N1 is more efficient than A3 for $n \geq 1020$ in $\mathbb{F}_3[x]$ and for $n \geq 84$ in $\mathbb{F}_9[x]$.

Table 4.1: Costs of multi-evaluation and reconstruction for N1 in $\mathbb{F}_9[x]$

Computations	Cost for multiplication in $\mathbb{F}_9[x]$
$R_0 = A_0 - A_2$, $R'_0 = B_0 - B_2$	$4n$
$R_1 = A_1 - A_3$, $R'_1 = B_1 - B_3$	$4n$
$R_2 = A_1 + A_3$, $R'_2 = B_1 + B_3$	$4n$
$R_3 = A_0 + R_2$, $R'_3 = B_0 + R'_2$	$4n$
$R_4 = R_1 - A_2$, $R'_4 = R'_1 - B_2$	$4n$
$R_5 = A_0 - R_2$, $R'_5 = B_0 - R'_2$	$4n$
$R_6 = -A_2 - R_1$, $R'_6 = -B_2 - R'_1$	$4n$
$R_7 = R_0 + \omega R_1$, $R'_7 = R'_0 + \omega R'_1$	$4n$
$R_8 = R_3 + \omega R_4$, $R'_8 = R'_3 + \omega R'_4$	$4n$
$R_9 = R_5 + \omega R_6$, $R'_9 = R'_5 + \omega R'_6$	$4n$
$R_{10} = R_0 - \omega R_1$, $R'_{10} = R'_0 - \omega R'_1$	$4n$
$R_{11} = R_3 - \omega R_4$, $R'_{11} = R'_3 - \omega R'_4$	$4n$
$R_{12} = R_5 - \omega R_6$, $R'_{12} = R'_5 - \omega R'_6$	$4n$
$P_0 = R_7 R'_7$	$M_9(n)$
$P_1 = R_{10} R'_{10}$	$M_9(n)$
$P_2 = R_8 R'_8$	$M_9(n)$
$P_3 = R_{11} R'_{11}$	$M_9(n)$
$P_4 = R_9 R'_9$	$M_9(n)$
$P_5 = R_{12} R'_{12}$	$M_9(n)$
$P_6 = A_3 B_3$	$M_9(n)$
$U_1 = P_0 + P_1$	$2(2n-1)$
$U_2 = -P_0 + P_1$	$2(2n-1)$
$U_3 = P_2 + P_3$	$2(2n-1)$
$U_4 = -P_2 + P_3$	$2(2n-1)$
$U_5 = P_4 + P_5$	$2(2n-1)$
$U_6 = -P_4 + P_5$	$2(2n-1)$
$U_7 = U_3 + U_5$	$2(2n-1)$
$U_8 = -U_3 + U_5$	$2(2n-1)$
$U_9 = U_4 + U_6$	$2(2n-1)$
$U_{10} = U_4 - U_6$	$2(2n-1)$
$U_{11} = U_1 - U_7$	$2(2n-1)$
$U_{12} = U_{11} + P_6$	$2(2n-1)$
$U_{13} = U_1 + U_7$	$2(2n-1)$
$U_{14} = U_{13} + P_6$	$2(2n-1)$
$C_0 = U_{12} + \omega U_9$	$2(2n-1)$
$C_1 = U_8 + \omega U_2$	$2(2n-1)$
$C_2 = P_6 - \omega U_9$	$2(2n-1)$
$C_3 = U_8 + \omega U_{10}$	$2(2n-1)$
$C_4 = U_{14} + \omega U_9$	$2(2n-1)$
$C_5 = \omega(U_2 + U_{10})$	$2(2n-1)$
$C = C_0 + C_1 x^n \quad + C_2 x^{2n} + ... + C_6 x^{6n}$	$12(n-1)$
TOTAL:	$M_9(4n) = 7M_9(n) + 144n - 52$

Table 4.2: Costs of multi-evaluation and reconstruction for N1 in $\mathbb{F}_3[x]$

Computations	Cost for multiplication in $\mathbb{F}_3[x]$
$R_0 = A_0 - A_2$, $R'_0 = B_0 - B_2$	$2n$
$R_1 = A_1 - A_3$, $R'_1 = B_1 - B_3$	$2n$
$R_2 = A_1 + A_3$, $R'_2 = B_1 + B_3$	$2n$
$R_3 = A_0 + R_2$, $R'_3 = B_0 + R'_2$	$2n$
$R_4 = R_1 - A_2$, $R'_4 = R'_1 - B_2$	$2n$
$R_5 = A_0 - R_2$, $R'_5 = B_0 - R'_2$	$2n$
$R_6 = -A_2 - R_1$, $R'_6 = -B_2 - R'_1$	$2n$
$R_7 = R_0 + \omega R_1$, $R'_7 = R'_0 + \omega R'_1$	0
$R_8 = R_3 + \omega R_4$, $R'_8 = R'_3 + \omega R'_4$	0
$R_9 = R_5 + \omega R_6$, $R'_9 = R'_5 + \omega R'_6$	0
$R_{10} = R_0 - \omega R_1$, $R'_{10} = R'_0 - \omega R'_1$	0
$R_{11} = R_3 - \omega R_4$, $R'_{11} = R'_3 - \omega R'_4$	0
$R_{12} = R_5 - \omega R_6$, $R'_{12} = R'_5 - \omega R'_6$	0
$P_0 = R_7 R'_7$	$M_9(n)$
$P_1 = R_{10} R'_{10}$	0
$P_2 = R_8 R'_8$	$M_9(n)$
$P_3 = R_{11} R'_{11}$	0
$P_4 = R_9 R'_9$	$M_9(n)$
$P_5 = R_{12} R'_{12}$	0
$P_6 = A_3 B_3$	$M_3(n)$
$U_0 = P_{2,1} + P_{4,1}$	$(2n-1)$
$U_1 = P_{4,1} - P_{2,1}$	$(2n-1)$
$U_2 = P_{2,0} + P_{4,0}$	$(2n-1)$
$U_3 = P_{2,0} - P_{4,0}$	$(2n-1)$
$U_4 = P_6 - P_{0,0}$	$(2n-1)$
$U_5 = U_4 - U_0$	$(2n-1)$
$C_0 = U_5 + U_2$	$(2n-1)$
$C_1 = U_3 - P_{0,1}$	$(2n-1)$
$C_2 = U_0 + P_6$	$(2n-1)$
$C_3 = U_1 + U_3$	$(2n-1)$
$C_4 = U_5 - U_2$	$(2n-1)$
$C_5 = U_1 - P_{0,1}$	$(2n-1)$
$C = C_0 + C_1 x^n \quad + C_2 x^{2n} + ... + C_6 x^{6n}$	$6(n-1)$
TOTAL:	$M_3(4n) = M_3(n) + 3M_9(n) + 44n - 18$

4.2 An Improved 4-way Multiplication Algorithm (N2)

The new 4-way algorithm N1 from the previous section can be improved if we choose different interpolation points. This time we use $\{0, 1, \omega+1, -\omega+1, -\omega-1, \omega-1, \infty\}$ as the interpolation evaluation points that produces,

$$M_3(4n) = 3M_3(n) + 2M_9(n) + O(n)$$

Assume that the same settings through (4.1)-(4.4) for $A(x)$, $B(x)$, and $C(x)$ are satisfied. This time we get the products of $1/4$ sized polynomials as follows,

$$\left.\begin{aligned}
&P_0 = A_0.B_0 = C(0)\\
&P_1 = (A_0 + A_1 + A_2 + A_3).(B_0 + B_1 + B_2 + B_3) = C(1)\\
&P_2 = [(A_0 + A_1 + A_3) + \omega(A_1 - A_2 - A_3)].[(B_0 + B_1 + B_3) + \omega(B_1 - B_2 - B_3)] = C(\omega + 1)\\
&P_3 = [(A_0 + A_1 + A_3) + \omega(-A_1 + A_2 + A_3)].[(B_0 + B_1 + B_3) + \omega(-B_1 + B_2 + B_3)] = C(-\omega + 1)\\
&P_4 = [(A_0 - A_1 - A_3) + \omega(-A_1 - A_2 + A_3)].[(B_0 - B_1 - B_3) + \omega(-B_1 - B_2 + B_3)] = C(-\omega - 1)\\
&P_5 = [(A_0 - A_1 - A_3) + \omega(A_1 + A_2 - A_3)].[(B_0 - B_1 - B_3) + \omega(B_1 + B_2 - B_3)] = C(\omega - 1)\\
&P_6 = A_3.B_3 = C_6
\end{aligned}\right\} \tag{4.13}$$

Also let P_i for $2 \leq i \leq 5$ hold the equalities in (4.6), then,

$$\left.\begin{aligned}
P_{2,0} &= P_{3,0}\\
P_{2,1} &= -P_{3,1}\\
P_{4,0} &= P_{5,0}\\
P_{4,1} &= -P_{5,1}
\end{aligned}\right\} \tag{4.14}$$

From (4.14) P_3 and P_5 can be derived out of P_2 and P_4 thus, it is sufficient to calculate the latter two multiplications only. In this way, we save two $\mathbb{F}_9[x]$ multiplications. Interpolation regarding the N2 algorithm gives us the following results,

$$\left.\begin{aligned}
C_0 &= P_0\\
C_1 &= -P_0 - P_1 - P_2 - P_3 - P_4 - P_5 - P_6 + \omega(-P_2 + P_3)\\
C_2 &= P_6 + \omega(P_2 - P_3 + P_4 - P_5)\\
C_3 &= -P_2 - P_3 + P_4 + P_5 + \omega(-P_2 + P_3 + P_4 - P_5)\\
C_4 &= P_0 - P_2 - P_3 - P_4 - P_5\\
C_5 &= -P_0 - P_1 + P_4 + P_5 - P_6 + \omega(P_2 - P_3 + P_4 - P_5)\\
C_6 &= P_6
\end{aligned}\right\} \tag{4.15}$$

inserting the equalities from (4.6), (4.14) into (4.15) we get,

$$\left.\begin{aligned}
C_0 &= P_0 \\
C_1 &= -P_0 - P_1 + P_{2,0} + P_{4,0} - P_6 - P_{2,1} \\
C_2 &= P_6 + P_{2,1} + P_{4,1} \\
C_3 &= P_{2,0} - P_{4,0} - P_{2,1} + P_{4,1} \\
C_4 &= P_0 + P_{2,0} + P_{4,0} \\
C_5 &= -P_0 - P_1 - P_{4,0} - P_6 + P_{2,1} + P_{4,1} \\
C_6 &= P_6
\end{aligned}\right\} \tag{4.16}$$

By using the cost of multi-evaluation and reconstruction values from Table 4.3 and Table 4.4, we calculate the complexities associated with N2 as follows,

$$\left.\begin{aligned}
M_9(4n) &\leq 7M_9(n) + 132n - 48, M_9(1) = 6 \\
M_{9,\otimes}(4n) &\leq 7M_{9,\otimes}(n), M_{9,\otimes}(1) = 4 \\
M_{9,\oplus}(4n) &\leq 7M_{9,\oplus}(n) + 132n - 48, M_{9,\oplus}(1) = 2 \\
M_3(4n) &\leq 3M_3(n) + 2M_9(n) + 50n - 20, M_3(1) = 1 \\
M_{3,\otimes}(4n) &\leq 3M_{3,\otimes}(n) + 2M_{9,\otimes}(n), M_{3,\otimes}(1) = 1 \\
M_{3,\oplus}(4n) &\leq 3M_{3,\oplus}(n) + 2M_{9,\oplus}(n) + 50n - 20, M_{3,\oplus}(1) = 0
\end{aligned}\right\} \tag{4.17}$$

And by Remark 1, we get the following explicit complexities:

$$\left.\begin{aligned}
M_9(n) &\leq 42n^{\log_4 7} - 44n - 8 \\
M_{9,\otimes}(n) &\leq 4n^{\log_4 7} \\
M_{9,\oplus}(n) &\leq 38n^{\log_4 7} - 44n - 8 \\
M_3(n) &\leq 21n^{\log_4 7} - 38n + 18 \\
M_{3,\otimes}(n) &\leq 2n^{\log_4 7} - n^{\log_4 3} \\
M_{3,\oplus}(n) &\leq 19n^{\log_4 7} + n^{\log_4 3} - 38n + 18
\end{aligned}\right\} \tag{4.18}$$

Moreover, assuming that $A(x)$ and $B(x)$ are degree $3n + k - 1$ polynomials where $1 \leq k \leq n$, $A_0, A_1, A_2, B_0, B_1, B_2$ are degree $n - 1$ polynomials and A_3, B_3 are degree $k - 1$ polynomials. Then, the cost analysis of the N2 4-way algorithm in

Table A.6 and Table A.7 (See Appendix A) yield,

$$\left.\begin{aligned} M_3(3n+k) &\leq 2M_3(n) + M_3(k) + 2M_9(n) + 38n + 12k - 20 \\ M_9(3n+k) &\leq 6M_9(n) + M_9(k) + 108n + 24k - 48 \end{aligned}\right\} \tag{4.19}$$

N2 becomes better than KA2 for $n \geq 60$ in $\mathbb{F}_3[x]$ and for $n \geq 20$ in $\mathbb{F}_9[x]$. N2 is more efficient than A3 beginning from $n \geq 180$ in $\mathbb{F}_3[x]$ and for $n \geq 72$ in $\mathbb{F}_9[x]$. Moreover, N2 is more efficient than the algorithm A2 for $n \geq 20$.

Table 4.3: Costs of multi-evaluation and reconstruction for N2 in $\mathbb{F}_3$

Computations	Cost for Multiplication in $\mathbb{F}_3$
$R_1 = A_1 + A_3$, $R'_1 = B_1 + B_3$	$2n$
$R_2 = R_1 + A_0$, $R'_2 = R'_1 + B_0$	$2n$
$R_3 = R_2 + A_2$, $R'_3 = R'_2 + B_2$	$2n$
$R_4 = A_0 - R_1$, $R'_4 = B_0 - R'_1$	$2n$
$R_5 = A_1 - A_2$, $R'_5 = B_1 - B_2$	$2n$
$R_6 = R_5 - A_3$, $R'_6 = R'_5 - B_3$	$2n$
$R_7 = -A_1 - A_2$, $R'_7 = -B_1 - B_2$	$2n$
$R_8 = R_7 + A_3$, $R'_8 = R'_7 + B_3$	$2n$
$R_9 = R_2 + \omega R_6$, $R'_9 = R'_2 + \omega R'_6$	0
$R_{10} = R_4 + \omega R_8$, $R'_{10} = R'_4 + \omega R'_8$	0
$R_{11} = R_2 - \omega R_6$, $R'_{11} = R'_2 - \omega R'_6$	0
$R_{12} = R_4 - \omega R_8$, $R'_{12} = R'_4 - \omega R'_8$	0
$P_0 = A_0 B_0$	$M_3(n)$
$P_1 = R_3 R'_3$	$M_3(n)$
$P_2 = R_9 R'_9$	$M_9(n)$
$P_3 = R_{11} R'_{11}$	0
$P_4 = R_{10} R'_{10}$	$M_9(n)$
$P_5 = R_{12} R'_{12}$	0
$P_6 = A_3 B_3$	$M_3(n)$
$U_1 = P_0 + P_1$	$(2n-1)$
$U_2 = P_{2,0} + P_{4,0}$	$(2n-1)$
$U_3 = P_{2,1} + P_{4,1}$	$(2n-1)$
$U_4 = P_{2,0} - P_{4,0}$	$(2n-1)$
$U_5 = P_{2,1} - P_{4,1}$	$(2n-1)$
$U_6 = U_2 - U_1$	$(2n-1)$
$U_7 = U_6 - P_6$	$(2n-1)$
$U_8 = U_3 - U_1$	$(2n-1)$
$U_9 = U_8 - P_{4,0}$	$(2n-1)$
$C_1 = U_7 - P_{2,1}$	$(2n-1)$
$C_2 = U_3 + P_6$	$(2n-1)$
$C_3 = U_4 - U_5$	$(2n-1)$
$C_4 = U_2 + P_0$	$(2n-1)$
$C_5 = U_9 - P_6$	$(2n-1)$
$C = C_0 + C_1 x^n + C_2 x^{2n} + ... + C_6 x^{6n}$	$6(n-1)$
TOTAL:	$M_3(4n) = 3M_3(n) + 2M_9(n) + 50n - 20$

Table 4.4: Costs of multi-evaluation and reconstruction for N2 in $\mathbb{F}_9$

Computations	Cost for Multiplication in $\mathbb{F}_9$
$R_1 = A_1 + A_3$, $R'_1 = B_1 + B_3$	$4n$
$R_2 = R_1 + A_0$, $R'_2 = R'_1 + B_0$	$4n$
$R_3 = R_2 + A_2$, $R'_3 = R'_2 + B_2$	$4n$
$R_4 = A_0 - R_1$, $R'_4 = B_0 - R'_1$	$4n$
$R_5 = A_1 - A_2$, $R'_5 = B_1 - B_2$	$4n$
$R_6 = R_5 - A_3$, $R'_6 = R'_5 - B_3$	$4n$
$R_7 = -A_1 - A_2$, $R'_7 = -B_1 - B_2$	$4n$
$R_8 = R_7 + A_3$, $R'_8 = R'_7 + B_3$	$4n$
$R_9 = R_2 + \omega R_6$, $R'_9 = R'_2 + \omega R'_6$	$4n$
$R_{10} = R_4 + \omega R_8$, $R'_{10} = R'_4 + \omega R'_8$	$4n$
$R_{11} = R_2 - \omega R_6$, $R'_{11} = R'_2 - \omega R'_6$	$4n$
$R_{12} = R_4 - \omega R_8$, $R'_{12} = R'_4 - \omega R'_8$	$4n$
$P_0 = A_0 B_0$	$M_9(n)$
$P_1 = R_3 R'_3$	$M_9(n)$
$P_2 = R_9 R'_9$	$M_9(n)$
$P_3 = R_{11} R'_{11}$	$M_9(n)$
$P_4 = R_{10} R'_{10}$	$M_9(n)$
$P_5 = R_{12} R'_{12}$	$M_9(n)$
$P_6 = A_3 B_3$	$M_9(n)$
$U_1 = -P_0 - P_1$	$2(2n-1)$
$U_2 = P_4 + P_5$	$2(2n-1)$
$U_3 = P_2 + P_3$	$2(2n-1)$
$U_4 = P_2 - P_3$	$2(2n-1)$
$U_5 = P_4 - P_5$	$2(2n-1)$
$U_6 = -U_3 - U_2$	$2(2n-1)$
$U_7 = U_1 + U_6$	$2(2n-1)$
$U_8 = U_7 - P_6$	$2(2n-1)$
$U_9 = -U_4 + U_5$	$2(2n-1)$
$U_{10} = -U_3 + U_2$	$2(2n-1)$
$U_{11} = U_4 + U_5$	$2(2n-1)$
$U_{12} = U_1 + U_2$	$2(2n-1)$
$U_{13} = U_{12} - P_6$	$2(2n-1)$
$C_1 = U_8 - \omega U_4$	$2(2n-1)$
$C_2 = P_6 + \omega U_{11}$	$2(2n-1)$
$C_3 = U_{10} + \omega U_9$	$2(2n-1)$
$C_4 = P_0 + U_6$	$2(2n-1)$
$C_5 = U_{13} + U_{11}$	$2(2n-1)$
$C = C_0 + C_1 x^n + C_2 x^{2n} + ... + C_6 x^{6n}$	$12(n-1)$
TOTAL:	$M_9(4n) = 7M_9(n) + 132n - 48$

4.3 Another Improved 4-way Polynomial Multiplication Algorithm (N3)

In this section, we propose another improved 4-way multiplication algorithm N3, with seven $1/4$ sized multiplications, which is derived by using Lagrange interpolation in $\mathcal{R} = \mathbb{F}_9[x]$ with evaluation points $\{0, 1, -1, x, \omega, -\omega, \infty\}$ [4, 17]. Assume that, $A(x)$ and $B(x)$ are two polynomials of degree $4n - 1$ where $n = 4^k$ for some $k \geq 0$. Also let $y = x^n$, $C(x) = A(x)B(x)$. $A(x)$ and $B(x)$ are divided into four parts as follows: $A(x) = A_0 + yA_1 + y^2A_2 + y^3A_3$ and $B(x) = B_0 + yB_1 + y^2B_2 + y^3B_3$ where A_i and B_i are polynomials of degree less then n for all $i \in \{0, 1, 2, 3\}$. Using $\{0, 1, -1, x, \omega, -\omega, \infty\}$ as the points of evaluation for Lagrange interpolation we get;

$$\left.\begin{aligned}
P_0 &= A_0B_0 \\
P_1 &= (A_0 + A_1 + A_2 + A_3)(B_0 + B_1 + B_2 + B_3) \\
P_2 &= (A_0 - A_1 + A_2 - A_3)(B_0 - B_1 + B_2 - B_3) \\
P_3 &= (A_0 + A_1x + A_2x^2 + A_3x^3)(B_0 + B_1x + B_2x^2 + B_3x^3) \\
P_4 &= [(A_0 - A_2) + \omega(A_1 - A_3)][(B_0 - B_2) + \omega(B_1 - B_3)] \\
P_5 &= [(A_0 - A_2) - \omega(A_1 - A_3)][(B_0 - B_2) - \omega(B_1 - B_3)] \\
P_6 &= A_3B_3 = C_6
\end{aligned}\right\} \tag{4.20}$$

Let,

$$\left.\begin{aligned}
P_4 &= P_{4,0} + \omega P_{4,1} \\
P_5 &= P_{5,0} + \omega P_{5,1}
\end{aligned}\right\} \tag{4.21}$$

then one can observe that,

$$\left.\begin{aligned}
P_{4,0} &= P_{5,0} \\
P_{4,1} &= -P_{5,1}
\end{aligned}\right\} \tag{4.22}$$

By (4.21) and (4.22), the product P_4 can be calculated from the product P_5. Thus, one multiplication gets cost-free. We get the formula for $C(x)$ as follows:

$$
\begin{aligned}
C(x) =& P_0 + x^n.\left[x^2\left(\frac{(P_1-P_2)}{x^2-1} - \frac{\omega(P_4-P_5)}{x^2+1}\right) - U\right] \\
&+ x^{2n}.[(P_1+P_2) - (P_4+P_5) - P_6] \\
&+ x^{3n}.[(P_1-P_2) + \omega(P_4-P_5)] + x^{4n}.[-P_0 + (P_1+P_2) + (P_4+P_5)] \\
&+ x^{5n}.\left[\left(-\frac{(P_1-P_2)}{x^2-1} - \frac{\omega(P_4-P_5)}{x^2+1}\right) + U\right] + x^{6n}.P_6
\end{aligned}
$$

where,

$$
U = \frac{P_0}{x} + \frac{P_3/x}{x^4-1} - x\left(\frac{P_4+P_5}{x^2+1} + \frac{P_1+P_2}{x^2-1}\right) - P_6 x
$$

Note that, N3 is not fully recursive and can only be applied once at a time since the six products P_0, P_1, P_2, P_4, P_5, and P_6 involve polynomials of degree $n-1$, but P_3 involves polynomials of degree $n+2$. To get a fully recursive version of N3, we express the product of degree $n+2$ polynomials in terms of one product of degree $n-1$ polynomials plus some additional non-recursive terms and then we expand the multiplication using schoolbook method, compute each product of the expansion separately and add them up to get the final result. The result indicates the following equalities,

$$
\left.
\begin{aligned}
M_3(n+3) &= M_3(n) + 12n + 12 \\
M_{3,\otimes}(n+3) &= M_{3,\otimes}(n) + 6n + 9 \\
M_{3,\oplus}(n+3) &= M_{3,\oplus}(n) + 6n + 3 \\
M_9(n+3) &= M_9(n) + 48n + 60 \\
M_{9,\otimes}(n+3) &= M_{9,\otimes}(n) + 24n + 36 \\
M_{9,\oplus}(n+3) &= M_{9,\oplus}(n) + 24n + 24
\end{aligned}
\right\} \tag{4.23}
$$

The costs of multi-evaluation and reconstruction for N3 are given in Table 4.5 by

means of which we get the complexity of the N3 algorithm as follows:

$$\left.\begin{aligned} M_9(4n) &\leq 7M_9(n) + 196n - 40, M_9(1) = 6 \\ M_{9,\otimes}(4n) &\leq 7M_{9,\oplus}(n) + 24n + 36, M_{9,\otimes}(1) = 4 \\ M_{9,\oplus}(4n) &\leq 7M_{9,\otimes}(n) + 172n - 76, M_{9,\oplus}(1) = 2 \\ M_3(4n) &\leq 5M_3(n) + M_9(n) + 78n - 36, M_3(1) = 1 \\ M_{3,\otimes}(4n) &\leq 5M_{3,\otimes}(n) + M_{9,\otimes}(n) + 6n + 9, M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(4n) &\leq 5M_{3,\oplus}(n) + M_{9,\oplus}(n) + 72n - 45, M_{3,\oplus}(1) = 0 \end{aligned}\right\} \tag{4.24}$$

$$\left.\begin{aligned} M_9(n) &\leq 64.66n^{\log_4 7} - 65.33n - 6.66 \\ M_{9,\otimes}(n) &\leq 18n^{\log_4 7} - 8n + 6 \\ M_{9,\oplus}(n) &\leq 46.66n^{\log_4 7} - 57.33n - 12.66 \\ M_3(n) &\leq 32.33n^{\log_4 7} - 29.33n^{\log_4 5} - 12.66n + 10.66 \\ M_{3,\otimes}(n) &\leq 9n^{\log_4 7} - 6.25n^{\log_4 5} + 2n - 3.75 \\ M_{3,\oplus}(n) &\leq 23.33n^{\log_4 7} - 23.08n^{\log_4 5} - 14.66n + 14.41 \end{aligned}\right\} \tag{4.25}$$

Moreover, assuming that $A(x)$ and $B(x)$ are degree $3n + k - 1$ polynomials where $1 \leq k \leq (n-1)$, $A_0, A_1, A_2, B_0, B_1, B_2$ are degree $n - 1$ polynomials and A_3, B_3 are degree $k - 1$ polynomials with $(n+1)/2 \leq k$. Then, the cost analysis of the N3 4-way algorithm in Table A.8 (See Appendix A) yields,

$$\left.\begin{aligned} M_3(3n + k) &\leq 4M_3(n) + M_3(k) + M_9(n) + 68n + 10k - 38 \\ M_9(3n + k) &\leq 6M_9(n) + M_9(k) + 176n + 20k - 44 \end{aligned}\right\} \tag{4.26}$$

In terms of arithmetic cost, the N3 4-way algorithm is better than the N1 and N2 4-way algorithms for $n \geq 64$. N3 is also better than Bernstein's 3-way algorithm B1 [5] for $n \geq 228$. Observe that, even though the arithmetic cost of N3 is better than those of N1 and N2 when it comes to implementation speed, there is a delay complexity, results from adding and shifting array elements, which makes N3 slower than N1 and N2. Note that the implementation speed might vary depending on the hardware architecture, thus, there is still a chance that we may get better speed results for N3 than those for N1 and N2 by implementing it in different platforms. On the other hand, N3 is still faster than Bernstein's 3-way algorithm B1 in terms of the implementation cycle counts as one can observe from Table 6.2 and Table 6.3.

Table 4.5: Costs of multi-evaluation and reconstruction for N3 in $\mathbb{F}_3$ and $\mathbb{F}_9$

Computations	Cost in $\mathbb{F}_3[x]$	Cost in $\mathbb{F}_9[x]$
$R_0 = A_0 + A_2$, $R'_0 = B_0 + B_2$	$2n$	$4n$
$R_1 = A_1 + A_3$, $R'_1 = B_1 + B_3$	$2n$	$4n$
$R_2 = R_0 + R_1, R'_2 = R'_0 + R'_1$	$2n$	$4n$
$R_3 = R_0 - R_1, R'_3 = R'_0 - R'_1$	$2n$	$4n$
$R_4 = A_0 + A_1x, R'_4 = B_0 + B_1x$	$(2n-2)$	$(4n-4)$
$R_5 = R_4 + A_2x^2, R'_5 = R'_4 + B_2x^2$	$(2n-2)$	$(4n-4)$
$R_6 = R_5 + A_3x^3, R'_6 = R'_5 + B'_3x^3$	$(2n-2)$	$(4n-4)$
$R_7 = A_0 - A_2, R'_7 = B_0 - B_2$	$2n$	$4n$
$R_8 = A_1 - A_3, R'_8 = B_1 - B_3$	$2n$	$4n$
$R_9 = R_7 + \omega R_8$, $R'_9 = R'_7 + \omega R'_8$	0	$4n$
$R_{10} = R_7 - \omega R_8$, $R'_9 = R'_7 - \omega R'_8$	0	$4n$
$P_0 = A_0B_0$	$M_3(n)$	$M_9(n)$
$P_1 = R_2R'_2$	$M_3(n)$	$M_9(n)$
$P_2 = R_3R'_3$	$M_3(n)$	$M_9(n)$
$P_3 = R_6R'_6$	$M_3(n+3) = M_3(n) + 12 + 12$	$M_9(n+3) = M_9(n) + 48n + 60$
$P_4 = R_9R'_9$	$M_9(n)$	$M_9(n)$
$P_5 = R_{10}R'_{10}$	0	$M_9(n)$
$P_6 = A_3B_3$	$M_3(n)$	$M_9(n)$
$U_0 = P_0/x$	0	0
$U_1 = P_3/x$	0	0
$U_2 = U_1/(x^2-1)$	$(2n)$	$4n$
$U_3 = U_2/(x^2+1)$	$(2n-2)$	$(4n-4)$
$U_4 = P_1 + P_2$	$(2n-1)$	$(4n-2)$
$U_5 = (P_1 + P_2)x$	0	0
$U_6 = U_5/(x^2-1)$	$(2n-4)$	$(4n-8)$
$U_7 = -P_{5,0}$	0	$(4n-2)$
$U_8 = -xP_{5,0}$	0	0
$U_9 = U_8/(x^2+1)$	$(2n-4)$	$(4n-8)$
$U_{10} = (U_6 + U_9)$	$(2n-2)$	$(4n-4)$
$U_{11} = U_0 + U_3$	$(2n-2)$	$(4n-4)$
$U_{12} = U_{11} - U_{10}$	$(2n-2)$	$(4n-4)$
$U = U_{12} - xP_6$	$(2n-1)$	$(4n-2)$
$V_0 = P_1 - P_2$	$(2n-1)$	$(4n-2)$
$V_1 = (P_1 - P_2)x^2$	0	0
$V_2 = V_1/(x^2-1)$	$(2n-3)$	$(4n-6)$
$V_3 = -P_{5,1}$	0	$(4n-2)$
$V_4 = -x^2P_{5,1}$	0	0
$V_5 = V_4/(x^2+1)$	$(2n-3)$	$(4n-6)$
$V_6 = V_2 - V_5$	$(2n-1)$	$(4n-2)$
$V_7 = V_6 - U$	$(2n-1)$	$(4n-2)$
$V_8 = U_4 - U_7$	$(2n-1)$	$(4n-2)$
$V_9 = V_8 - P_6$	$(2n-1)$	$(4n-2)$
$V_{10} = V_0 + V_3$	$(2n-1)$	$(4n-2)$
$V_{11} = -P_0 + U_4$	$(2n-1)$	$(4n-2)$
$V_{12} = V_{11} + U_7$	$(2n-1)$	$(4n-2)$
$V_{13} = -(V_2/x^2 + V_5/x^2)$	$(2n-3)$	$(4n-6)$
$V_{14} = V_{13} + U$	$(2n-3)$	$(4n-6)$
$C_0 = P_0 + x^nV_7$	$(n-1)$	$(2n-2)$
$C_1 = C_0 + x^{2n}V_9$	n	$2n$
$C_2 = C_1 + x^{3n}V_{10}$	$(n-1)$	$(2n-2)$
$C_3 = C_2 + x^{4n}V_{12}$	$(n-1)$	$(2n-2)$
$C_4 = C_3 + x^{5n}V_{14}$	$(n-1)$	$(2n-2)$
$C_= C_4 + x^{6n}P_6$	n	$2n$
TOTAL:	$M_3(4n) = 5M_3(n) + M_9(n) + 78n - 36$	$M_9(4n) = 7M_9(n) + 196n - 40$

CHAPTER 5

NEW 5-WAY SPLIT POLYNOMIAL MULTIPLICATION ALGORITHMS

In this chapter we propose a new 5-way split polynomial multiplication over characteristic three fields assuming that the input sizes are $n = 5^k$ for some integer $k \geq 1$. We then define the unbalanced split version of the same algorithm that can be used not only for the input sizes that are multiplies of 5 but for all input sizes $n \geq 17$.

5.1 New 5-way Multiplication Algorithm (V1)

In this section we derive a new 5-way split multiplication algorithm V1 which performs nine $1/5$ sized multiplications. This algorithm is based on the interpolation technique. Assume that,

$$\left.\begin{aligned} A(x) &= a_0 + a_1 x + a_2 x^2 + ... + a_{5n-1} x^{5n-1} \\ B(x) &= b_0 + b_1 x + b_2 x^2 + ... + b_{5n-1} x^{5n-1} \end{aligned}\right\} \tag{5.1}$$

are two polynomials of degree $5n - 1$ and $n = 5^k$ for some $k \geq 1$. Let $y = x^n$ and

also we assume that $C(x) = A(x)B(x)$. Given,

$$\left.\begin{aligned}
A_0 &= a_0 + a_1 x + ... + a_{n-1}x^{n-1} \\
A_1 &= a_n + a_{n+1}x + ... + a_{2n-1}x^{n-1} \\
A_2 &= a_{2n} + a_{2n+1}x + ... + a_{3n-1}x^{n-1} \\
A_3 &= a_{3n} + a_{3n+1}x + ... + a_{4n-1}x^{n-1} \\
A_4 &= a_{4n} + a_{4n+1}x + ... + a_{5n-1}x^{n-1} \\
B_0 &= b_0 + b_1 x + ... + b_{n-1}x^{n-1} \\
B_1 &= b_n + b_{n+1}x + ... + b_{2n-1}x^{n-1} \\
B_2 &= b_{2n} + b_{2n+1}x + ... + b_{3n-1}x^{n-1} \\
B_3 &= b_{3n} + b_{3n+1}x + ... + b_{4n-1}x^{n-1} \\
B_4 &= b_{4n} + b_{4n+1}x + ... + b_{5n-1}x^{n-1}
\end{aligned}\right\} \tag{5.2}$$

then,

$$\left.\begin{aligned}
A(x) &= A_0 + yA_1 + y^2A_2 + y^3A_3 + y^4A_4 \\
B(x) &= B_0 + yB_1 + y^2B_2 + y^3B_3 + y^4B_4
\end{aligned}\right\} \tag{5.3}$$

and the multiplication has the following form,

$$C(x) = C_0 + C_1y + C_2y^2 + C_3y^3 + C_4y^4 + C_5y^5 + C_6y^6 + C_7y^7 + C_8y^8 \tag{5.4}$$

To get the least expensive 1/5 sized products, we try each possible combination of $\mathbb{F}_9$ points for the interpolation evaluation and observe that the following nine interpolation points $\{0, 1, \omega, -\omega, \omega + 1, -\omega + 1, -\omega - 1, \omega - 1, \infty\}$ yield the most efficient 5-way algorithm.

$$\left.\begin{aligned}
&P_0 = A_0.B_0 = C_0\\
&P_1 = (A_0 + A_1 + A_2 + A_3 + A_4).(B_0 + B_1 + B_2 + B_3 + B_4) = C_1\\
&P_2 = [(A_0 + A_4 - A_2) + \omega(A_1 - A_3)].[(B_0 + B_4 - B_2) + \omega(B_1 - B_3)] = C_2\\
&P_3 = [(A_0 + A_4 - A_2) - \omega(A_1 - A_3)].[(B_0 + B_4 - B_2) - \omega(B_1 - B_3)] = C_3\\
&P_4 = [(A_0 + A_1 + A_3 - A_4) + \omega(A_1 - A_2 - A_3)].[(B_0 + B_1 + B_3 - B_4) + \omega(B_1 - B_2 - B_3)] = C_4\\
&P_5 = [(A_0 + A_1 + A_3 - A_4) + \omega(-A_1 + A_2 + A_3)].[(B_0 + B_1 + B_3 - B_4) + \omega(-B_1 + B_2 + B_3)] = C_5\\
&P_6 = [(A_0 - A_1 - A_3 - A_4) + \omega(-A_1 - A_2 + A_3)].[(B_0 - B_1 - B_3 - B_4) + \omega(-B_1 - B_2 + B_3)] = C_6\\
&P_7 = [(A_0 - A_1 - A_3 - A_4) + \omega(A_1 + A_2 - A_3)].[(B_0 - B_1 - B_3 - B_4) + \omega(B_1 + B_2 - B_3)] = C_7\\
&P_8 = A_4.B_4 = C_8
\end{aligned}\right\} \tag{5.5}$$

Let,

$$\left.\begin{aligned}
P_2 &= P_{2,0} + \omega P_{2,1}\\
P_3 &= P_{3,0} + \omega P_{3,1}\\
P_4 &= P_{4,0} + \omega P_{4,1}\\
P_5 &= P_{5,0} + \omega P_{5,1}\\
P_6 &= P_{6,0} + \omega P_{6,1}\\
P_7 &= P_{7,0} + \omega P_{7,1}
\end{aligned}\right\} \tag{5.6}$$

then one can observe that,

$$\left.\begin{aligned}
P_{2,0} &= P_{3,0}\\
P_{2,1} &= -P_{3,1}\\
P_{4,0} &= P_{5,0}\\
P_{4,1} &= -P_{5,1}\\
P_{6,0} &= P_{7,0}\\
P_{6,1} &= -P_{7,1}
\end{aligned}\right\} \tag{5.7}$$

Note that (5.6) and (5.7) helps us saving three $\mathbb{F}_9[x]$ multiplications. Finally, interpo-

lation gives us the following results,

$$\left.\begin{aligned}
C_0 &= P_0 \\
C_1 &= -P_0 + P_1 - P_2 - P_3 + P_6 + P_7 - P_8 + \omega(P_2 - P_3 - P_4 + P_5 + P_6 - P_7) \\
C_2 &= P_0 - P_2 - P_3 + P_4 + P_5 + P_6 + P_7 + P_8 + \omega(-P_4 + P_5 - P_6 + P_7) \\
C_3 &= -P_0 + P_1 - P_2 - P_3 + P_6 + P_7 - P_8 + \omega(-P_2 + P_3 + P_4 - P_5 - P_6 + P_7) \\
C_4 &= P_0 - P_4 - P_5 - P_6 - P_7 + P_8 \\
C_5 &= -P_0 + P_1 - P_2 - P_3 + P_4 + P_5 - P_8 + \omega(P_2 - P_3 + P_4 - P_5 - P_6 + P_7) \\
C_6 &= P_0 - P_2 - P_3 + P_4 + P_5 + P_6 + P_7 + P_8 + \omega(P_4 - P_5 + P_6 - P_7) \\
C_7 &= -P_0 + P_1 - P_2 - P_3 + P_4 + P_5 - P_8 + \omega(-P_2 + P_3 - P_4 + P_5 + P_6 - P_7) \\
C_8 &= P_8
\end{aligned}\right\} \tag{5.8}$$

Inserting the equalities from (5.6), (5.7) into (5.8) yields,

$$\left.\begin{aligned}
C_0 &= P_0 \\
C_1 &= -P_0 + P_1 + P_{2,0} - P_{6,0} - P_8 + P_{2,1} - P_{4,1} + P_{6,1} \\
C_2 &= P_0 + P_{2,0} - P_{4,0} - P_{6,0} + P_8 - P_{4,1} - P_{6,1} \\
C_3 &= -P_0 + P_1 + P_{2,0} - P_{6,0} - P_8 - P_{2,1} + P_{4,1} - P_{6,1} \\
C_4 &= P_0 + P_{4,0} + P_{6,0} + P_8 \\
C_5 &= -P_0 + P_1 + P_{2,0} - P_{4,0} - P_8 + P_{2,1} + P_{4,1} - P_{6,1} \\
C_6 &= P_0 + P_{2,0} - P_{4,0} - P_{6,0} + P_8 + P_{4,1} + P_{6,1} \\
C_7 &= -P_0 + P_1 + P_{2,0} - P_{4,0} - P_8 - P_{2,1} - P_{4,1} + P_{6,1} \\
C_8 &= P_8
\end{aligned}\right\} \tag{5.9}$$

By using cost of multi-evaluation values in Table 5.1 and Table 5.2, the associated

complexities for V1 in $\mathbb{F}_3[x]$ and $\mathbb{F}_9[x]$ can be calculated as follows,

$$\left.\begin{aligned} M_9(5n) &\leq 9M_9(n) + 196n - 72, M_9(1) = 6 \\ M_{9,\otimes}(5n) &\leq 9M_{9,\otimes}(n), M_{9,\otimes}(1) = 4 \\ M_{9,\oplus}(5n) &\leq 9M_{9,\oplus}(n) + 196n - 72, M_{9,\oplus}(1) = 2 \\ M_3(5n) &\leq 3M_3(n) + 3M_9(n) + 72n - 29, M_3(1) = 1 \\ M_{3,\otimes}(5n) &\leq 3M_{3,\otimes}(n) + 3M_{9,\otimes}(n), M_{3,\otimes}(1) = 1 \\ M_{3,\oplus}(5n) &\leq 3M_{3,\oplus}(n) + 3M_{9,\oplus}(n) + 72n - 29, M_{3,\oplus}(1) = 0 \end{aligned}\right\} \tag{5.10}$$

by Remark 1 we get,

$$\left.\begin{aligned} M_9(n) &\leq 47n^{\log_5 9} - 49n - 9 \\ M_{9,\otimes}(n) &\leq 4n^{\log_5 9} \\ M_{9,\oplus}(n) &\leq 43n^{\log_5 9} - 49n - 9 \\ M_3(n) &\leq 23.5n^{\log_5 9} - 13n^{\log_5 3} - 37.5n + 28 \\ M_{3,\otimes}(n) &\leq 2n^{\log_5 9} - n^{\log_5 3} \\ M_{3,\oplus}(n) &\leq 21.5n^{\log_5 9} - 12.n^{\log_5 3} - 37.5n + 28 \end{aligned}\right\} \tag{5.11}$$

Moreover, assuming that $A(x)$ and $B(x)$ are degree $4n + k - 1$ polynomials where $1 \leq k \leq n$, $A_0, A_1, A_2, A_3, B_0, B_1, B_2, B_3$ are degree $n - 1$ polynomials and A_4, B_4 are degree $k-1$ polynomials. Then, the cost analysis of the V1 5-way algorithm from Table A.9 and Table A.10 (See Appendix A) yield,

$$\left.\begin{aligned} M_3(4n + k) &\leq 2M_3(n) + M_3(k) + 3M_9(n) + 66n + 6k - 29 \\ M_9(4n + k) &\leq 8M_9(n) + M_9(k) + 168n + 28k - 72 \end{aligned}\right\} \tag{5.12}$$

V1 becomes more efficient than KA2 for $n \geq 100$ in $\mathbb{F}_3[x]$ and for $n \geq 20$ in $\mathbb{F}_9[x]$. Note that, V1 is better than A3 for $n \geq 60$ in $\mathbb{F}_3[x]$ and for $n \geq 15$ in $\mathbb{F}_9[x]$. V1 outperforms A2 beginning from $n \geq 15$. Also, V1 is better than N3 for $n \geq 20$. Note that, the use of V1 5-way algorithm yields the fastest of results compared to all aforementioned algorithms, in terms of both implementation cycles and arithmetic cost.

Table 5.1: Costs of multi-evaluation and reconstruction for V1 in $\mathbb{F}_3$

Computations	Cost for Multiplication in $\mathbb{F}_3$
$R_1 = A_1 + A_3, R'_1 = B_1 + B_3$	$2n$
$R_2 = A_0 - A_4, R'_2 = B_0 - B_4$	$2n$
$R_3 = A_0 + A_4, R'_3 = B_0 + B_4$	$2n$
$R_4 = A_1 - A_3, R'_4 = B_1 - B_3$	$2n$
$R_5 = R_4 - A_2, R'_5 = R'_4 - B_2$	$2n$
$R_6 = -A_2 - R_4, R'_6 = -B_2 - R'_4$	$2n$
$R_7 = R_3 - A_2, R'_7 = R'_3 - B_2$	$2n$
$R_8 = R_1 + R_2, R'_8 = R'_1 + R'_2$	$2n$
$R_9 = R_2 - R_1, R'_8 = R'_2 - R'_1$	$2n$
$R_{10} = R_1 + R_3, R'_{10} = R'_1 + R'_3$	$2n$
$R_{11} = R_{10} + A_2, R'_{11} = R'_{10} + B_2$	$2n$
$R_{12} = R_7 + \omega R_4, R'_{12} = R'_7 + \omega R'_4$	0
$R_{13} = R_7 - \omega R_4, R'_{13} = R'_7 - \omega R'_4$	0
$R_{14} = R_8 + \omega R_5, R'_{14} = R'_8 + \omega R'_5$	0
$R_{15} = R_8 - \omega R_5, R'_{15} = R'_8 - \omega R'_5$	0
$R_{16} = R_9 + \omega R_6, R'_{16} = R'_9 + \omega R'_6$	0
$R_{17} = R_9 - \omega R_6, R'_{17} = R'_9 - \omega R'_6$	0
$P_0 = A_0 B_0$	$M_3(n)$
$P_1 = R_{11} R'_{11}$	$M_3(n)$
$P_2 = R_{12} R'_{12}$	$M_9(n)$
$P_3 = R_{13} R'_{13}$	0
$P_4 = R_{14} R'_{14}$	$M_9(n)$
$P_5 = R_{15} R'_{15}$	0
$P_6 = R_{16} R'_{16}$	$M_9(n)$
$P_7 = R_{17} R'_{17}$	0
$P_8 = A_4 B_4$	$M_3(n)$
$U_1 = P_0 + P_8$	$(2n-1)$
$U_2 = -U_1 + P_1$	$(2n-1)$
$U_3 = P_{2,0} - P_{6,0}$	$(2n-1)$
$U_4 = U_3 - P_{4,0}$	$(2n-1)$
$U_5 = P_{2,0} - P_{4,0}$	$(2n-1)$
$U_6 = P_{6,0} + P_{4,0}$	$(2n-1)$
$U_7 = P_{2,1} - P_{4,1}$	$(2n-1)$
$U_8 = U_7 + P_{6,1}$	$(2n-1)$
$U_9 = P_{4,1} + P_{6,1}$	$(2n-1)$
$U_{10} = P_{2,1} + P_{4,1}$	$(2n-1)$
$U_{11} = U_{10} - P_{6,1}$	$(2n-1)$
$U_{12} = U_2 + U_3$	$(2n-1)$
$U_{13} = U_1 + U_4$	$(2n-1)$
$U_{14} = U_2 + U_5$	$(2n-1)$
$C_1 = U_{12} + U_8$	$(2n-1)$
$C_2 = U_{13} - U_9$	$(2n-1)$
$C_3 = U_{12} - U_8$	$(2n-1)$
$C_4 = U_1 + U_6$	$(2n-1)$
$C_5 = U_{14} + U_{11}$	$(2n-1)$
$C_6 = U_{13} + U_9$	$(2n-1)$
$C_7 = U_{14} - U_{11}$	$(2n-1)$
$C = C_0 + C_1 x^n + ... + C_8 x^{8n}$	$8(n-1)$
TOTAL:	$M_3(5n) = 3M_3(n) + 3M_9(n) + 72n - 29$

Table 5.2: Costs of multi-evaluation and reconstruction for V1 in $\mathbb{F}_9$

Computations	Cost for Multiplication in $\mathbb{F}_9$
$R_1 = A_1 + A_3, R'_1 = B_1 + B_3$	$4n$
$R_2 = A_0 - A_4, R'_2 = B_0 - B_4$	$4n$
$R_3 = A_0 + A_4, R'_3 = B_0 + B_4$	$4n$
$R_4 = A_1 - A_3, R'_4 = B_1 - B_3$	$4n$
$R_5 = R_4 - A_2, R'_5 = R'_4 - B_2$	$4n$
$R_6 = -A_2 - R_4, R'_6 = -B_2 - R'_4$	$4n$
$R_7 = R_3 - A_2, R'_7 = R'_3 - B_2$	$4n$
$R_8 = R_1 + R_2, R'_8 = R'_1 + R'_2$	$4n$
$R_9 = R_2 - R_1, R'_9 = R'_2 - R'_1$	$4n$
$R_{10} = R_1 + R_3, R'_{10} = R'_1 + R'_3$	$4n$
$R_{11} = R_{10} + A_2, R'_{11} = R'_{10} + B_2$	$4n$
$R_{12} = R_7 + \omega R_4, R'_{12} = R'_7 + \omega R'_4$	$4n$
$R_{13} = R_7 - \omega R_4, R'_{13} = R'_7 - \omega R'_4$	$4n$
$R_{14} = R_8 + \omega R_5, R'_{14} = R'_8 + \omega R'_5$	$4n$
$R_{15} = R_8 - \omega R_5, R'_{15} = R'_8 - \omega R'_5$	$4n$
$R_{16} = R_9 + \omega R_6, R'_{16} = R'_9 + \omega R'_6$	$4n$
$R_{17} = R_9 - \omega R_6, R'_{17} = R'_9 - \omega R'_6$	$4n$
$P_0 = A_0 B_0$	$M_9(n)$
$P_1 = R_{11} R'_{11}$	$M_9(n)$
$P_2 = R_{12} R'_{12}$	$M_9(n)$
$P_3 = R_{13} R'_{13}$	$M_9(n)$
$P_4 = R_{14} R'_{14}$	$M_9(n)$
$P_5 = R_{15} R'_{15}$	$M_9(n)$
$P_6 = R_{16} R'_{16}$	$M_9(n)$
$P_7 = R_{17} R'_{17}$	$M_9(n)$
$P_8 = A_4 B_4$	$M_9(n)$
$U_1 = -P_0 + P_1$	$2(2n-1)$
$U_2 = P_2 + P_3$	$2(2n-1)$
$U_3 = P_2 - P_3$	$2(2n-1)$
$U_4 = P_6 + P_7$	$2(2n-1)$
$U_5 = P_6 - P_7$	$2(2n-1)$
$U_6 = P_4 + P_5$	$2(2n-1)$
$U_7 = P_4 - P_5$	$2(2n-1)$
$U_8 = U_4 + U_6$	$2(2n-1)$
$U_9 = U_1 - U_2$	$2(2n-1)$
$U_{10} = U_9 + U_4$	$2(2n-1)$
$U_{11} = U_{10} - P_8$	$2(2n-1)$
$U_{12} = -U_7 + U_5$	$2(2n-1)$
$U_{13} = U_3 + U_{12}$	$2(2n-1)$
$U_{14} = P_0 - U_2$	$2(2n-1)$
$U_{15} = U_{14} + U_8$	$2(2n-1)$
$U_{16} = U_{15} + P_8$	$2(2n-1)$
$U_{17} = -U_7 - U_5$	$2(2n-1)$
$U_{18} = U_9 + U_6$	$2(2n-1)$
$U_{19} = U_{18} - P_8$	$2(2n-1)$
$U_{20} = U_3 - U_{12}$	$2(2n-1)$
$U_{21} = P_0 - U_8$	$2(2n-1)$
$C_1 = U_{11} + \omega U_{13}$	$2(2n-1)$
$C_2 = U_{16} + \omega U_{17}$	$2(2n-1)$
$C_3 = U_{11} - \omega U_{13}$	$2(2n-1)$
$C_4 = U_{21} + P_8$	$2(2n-1)$
$C_5 = U_{19} + \omega U_{20}$	$2(2n-1)$
$C_6 = U_{16} - \omega U_{17}$	$2(2n-1)$
$C_7 = U_{19} - \omega U_{20}$	$2(2n-1)$
$C = C_0 + C_1 x^n + ... + C_8 x^{8n}$	$16(n-1)$
TOTAL:	$M_9(5n) = 9M_9(n) + 196n - 72$

5.2 Unbalanced Split 5-way Polynomial Multiplication Algorithm (U1)

In this section, we define the unbalanced split version of the 5-way polynomial multiplication algorithm V1 and we call this version U1. The main advantage of U1 is that it can be used not only for input sizes that are multiples of 5 but also for all input sizes $n \geq 17$. Assume that $A(x)$ and $B(x)$ are two degree $n-1$ polynomials with $n \in \mathbb{Z}^+$ and $n \geq 17$. Also let $k \equiv 5 - (n \pmod 5)$, i.e., $k \in \{0,1,2,3,4\}$. If n is not a multiple of 5 then we divide A and B into five smaller size polynomials so that the first four of them have $(n+k)/5$ elements and the last one has $(n-4k)/5$ elements. By this means, we get an *unbalanced* 5-way division method for any polynomial with size $n \geq 17$. Let $y = x^{(n+k)/5}$ and $C(x) = A(x)B(x)$. $A(x)$ and $B(x)$ are divided into five parts as follows:

$$\left.\begin{aligned}
A_0 &= a_0 + a_1 x + ... + a_{\frac{(n+k)}{5}-1} x^{\frac{(n+k)}{5}-1} \\
A_1 &= a_{\frac{(n+k)}{5}} + a_{\frac{(n+k)}{5}+1} x + ... + a_{\frac{2(n+k)}{5}-1} x^{\frac{(n+k)}{5}-1} \\
A_2 &= a_{\frac{2(n+k)}{5}} + a_{\frac{2(n+k)}{5}+1} x + ... + a_{\frac{3(n+k)}{5}-1} x^{\frac{(n+k)}{5}-1} \\
A_3 &= a_{\frac{3(n+k)}{5}} + a_{\frac{3(n+k)}{5}+1} x + ... + a_{\frac{4(n+k)}{5}-1} x^{\frac{(n+k)}{5}-1} \\
A_4 &= a_{\frac{4(n+k)}{5}} + a_{\frac{4(n+k)}{5}+1} x + ... + a_{n-1} x^{\frac{(n-4k)}{5}-1}
\end{aligned}\right\} \tag{5.13}$$

Similarly, we divide $B(x)$ into five pieces just as we do to $A(x)$ above and then we get,

$$\left.\begin{aligned}
A(x) &= A_0 + yA_1 + y^2A_2 + y^3A_3 + y^4A_4 \\
B(x) &= B_0 + yB_1 + y^2B_2 + y^3B_3 + y^4B_4
\end{aligned}\right\} \tag{5.14}$$

thus $C(x)$ becomes,

$$\begin{aligned}
C(x) &= (A_0 + yA_1 + y^2A_2 + y^3A_3 + y^4A_4)(B_0 + yB_1 + y^2B_2 + y^3B_3 + y^4B_4) \\
&= C_0 + C_1y + C_2y^2 + C_3y^3 + C_4y^4 + C_5y^5 + C_6y^6 + C_7y^7 + C_8y^8
\end{aligned}$$

We use the same nine interpolation points $\{0, 1, \omega, -\omega, \omega+1, -\omega+1, -\omega-1, \omega-1, \infty\}$

as in the derivation of V1 algorithm:

$$\left.\begin{aligned}
P_0 &= A_0B_0 \\
P_1 &= (A_0 + A_1 + A_2 + A_3 + A_4)(B_0 + B_1 + B_2 + B_3 + B_4) \\
P_2 &= [(A_0 + A_4 - A_2) + \omega(A_1 - A_3)][(B_0 + B_4 - B_2) + \omega(B_1 - B_3)] \\
P_3 &= [(A_0 + A_4 - A_2) - \omega(A_1 - A_3)][(B_0 + B_4 - B_2) - \omega(B_1 - B_3)] \\
P_4 &= [(A_0 + A_1 + A_3 - A_4) + \omega(A_1 - A_2 - A_3)][(B_0 + B_1 + B_3 - B_4) + \omega(B_1 - B_2 - B_3)] \\
P_5 &= [(A_0 + A_1 + A_3 - A_4) + \omega(-A_1 + A_2 + A_3)][(B_0 + B_1 + B_3 - B_4) + \omega(-B_1 + B_2 + B_3)] \\
P_6 &= [(A_0 - A_1 - A_3 - A_4) + \omega(-A_1 - A_2 + A_3)][(B_0 - B_1 - B_3 - B_4) + \omega(-B_1 - B_2 + B_3)] \\
P_7 &= [(A_0 - A_1 - A_3 - A_4) + \omega(A_1 + A_2 - A_3)][(B_0 - B_1 - B_3 - B_4) + \omega(B_1 + B_2 - B_3)] \\
P_8 &= A_4B_4
\end{aligned}\right\} \tag{5.15}$$

then the formulas for $C(x)$ becomes,

$$\left.\begin{aligned}
C_0 &= P_0 \\
C_1 &= -P_0 + P_1 + P_{2,0} - P_{6,0} - P_8 + P_{2,1} - P_{4,1} + P_{6,1} \\
C_2 &= P_0 + P_{2,0} - P_{4,0} - P_{6,0} + P_8 - P_{4,1} - P_{6,1} \\
C_3 &= -P_0 + P_1 + P_{2,0} - P_{6,0} - P_8 - P_{2,1} + P_{4,1} - P_{6,1} \\
C_4 &= P_0 + P_{4,0} + P_{6,0} + P_8 \\
C_5 &= -P_0 + P_1 + P_{2,0} - P_{4,0} - P_8 + P_{2,1} + P_{4,1} - P_{6,1} \\
C_6 &= P_0 + P_{2,0} - P_{4,0} - P_{6,0} + P_8 + P_{4,1} + P_{6,1} \\
C_7 &= -P_0 + P_1 + P_{2,0} - P_{4,0} - P_8 - P_{2,1} - P_{4,1} + P_{6,1} \\
C_8 &= P_8
\end{aligned}\right\} \tag{5.16}$$

where,

$$\left.\begin{aligned}
P_2 &= P_{2,0} + \omega P_{2,1} \\
P_3 &= P_{3,0} + \omega P_{3,1} \\
P_4 &= P_{4,0} + \omega P_{4,1} \\
P_5 &= P_{5,0} + \omega P_{5,1} \\
P_6 &= P_{6,0} + \omega P_{6,1} \\
P_7 &= P_{7,0} + \omega P_{7,1}
\end{aligned}\right\} \tag{5.17}$$

By using the cost of multi-evaluation and reconstruction values from Table 5.3 and

Table 5.4, the complexity of U1 can be calculated as below:

$$\left.\begin{aligned} M_9(n) &= 8M_9(\frac{n+k}{5}) + M_9(\frac{n-4k}{5}) + \frac{196n}{5} + \frac{76k}{5} - 72, M_9(1) = 6 \\ M_3(n) &= 2M_3(\frac{n+k}{5}) + M_3(\frac{n-4k}{5}) + 3M_9(\frac{n+k}{5}) + \frac{72n}{5} + \frac{42k}{5} - 29, M_3(1) = 1 \end{aligned}\right\} \tag{5.18}$$

Observe that, for $k = 5$, the U1 algorithm yields the V1 algorithm, so we can think of V1 as a special case of the U1 algorithm. According to Table 6.1, U1 is the best algorithm of all other algorithms in terms of arithmetic cost. Furthermore, one can convey from Table 6.2 and Table 6.3 that, the U1 algorithm has the fastest implementation run-time of all of the aforementioned 3-way and 4-way algorithms including Bernstein's B1 method [5].

Table 5.3: Costs of multi-evaluation and reconstruction for U1 in $\mathbb{F}_3$

Computations	Cost for Multiplication in $\mathbb{F}_3$
$R_1 = A_1 + A_3, R'_1 = B_1 + B_3$	$2(n+k)/5$
$R_2 = A_0 - A_4, R'_2 = B_0 - B_4$	$2(n-4k)/5$
$R_3 = A_0 + A_4, R'_3 = B_0 + B_4$	$2(n-4k)/5$
$R_4 = A_1 - A_3, R'_4 = B_1 - B_3$	$2(n+k)/5$
$R_5 = R_4 - A_2, R'_5 = R'_4 - B_2$	$2(n+k)/5$
$R_6 = -A_2 - R_4, R'_6 = -B_2 - R'_4$	$2(n+k)/5$
$R_7 = R_3 - A_2, R'_7 = R'_3 - B_2$	$2(n+k)/5$
$R_8 = R_1 + R_2, R'_8 = R'_1 + R'_2$	$2(n+k)/5$
$R_9 = R_2 - R_1, R'_8 = R'_2 - R'_1$	$2(n+k)/5$
$R_{10} = R_1 + R_3, R'_{10} = R'_1 + R'_3$	$2(n+k)/5$
$R_{11} = R_{10} + A_2, R'_{11} = R'_{10} + B_2$	$2(n+k)/5$
$R_{12} = R_7 + \omega R_4, R'_{12} = R'_7 + \omega R'_4$	0
$R_{13} = R_7 - \omega R_4, R'_{13} = R'_7 - \omega R'_4$	0
$R_{14} = R_8 + \omega R_5, R'_{14} = R'_8 + \omega R'_5$	0
$R_{15} = R_8 - \omega R_5, R'_{15} = R'_8 - \omega R'_5$	0
$R_{16} = R_9 + \omega R_6, R'_{16} = R'_9 + \omega R'_6$	0
$R_{17} = R_9 - \omega R_6, R'_{17} = R'_9 - \omega R'_6$	0
$P_0 = A_0 B_0$	$M_3((n+k)/5)$
$P_1 = R_{11} R'_{11}$	$M_3((n+k)/5)$
$P_2 = R_{12} R'_{12}$	$M_9((n+k)/5)$
$P_3 = R_{13} R'_{13}$	0
$P_4 = R_{14} R'_{14}$	$M_9((n+k)/5)$
$P_5 = R_{15} R'_{15}$	0
$P_6 = R_{16} R'_{16}$	$M_9((n+k)/5)$
$P_7 = R_{17} R'_{17}$	0
$P_8 = A_4 B_4$	$M_3((n-4k)/5)$
$U_1 = P_0 + P_8$	$(2(n-4k)/5 - 1)$
$U_2 = -U_1 + P_1$	$(2(n+k)/5 - 1)$
$U_3 = P_{2,0} - P_{6,0}$	$(2(n+k)/5 - 1)$
$U_4 = U_3 - P_{4,0}$	$(2(n+k)/5 - 1)$
$U_5 = P_{2,0} - P_{4,0}$	$(2(n+k)/5 - 1)$
$U_6 = P_{6,0} + P_{4,0}$	$(2(n+k)/5 - 1)$
$U_7 = P_{2,1} - P_{4,1}$	$(2(n+k)/5 - 1)$
$U_8 = U_7 + P_{6,1}$	$(2(n+k)/5 - 1)$
$U_9 = P_{4,1} + P_{6,1}$	$(2(n+k)/5 - 1)$
$U_{10} = P_{2,1} + P_{4,1}$	$(2(n+k)/5 - 1)$
$U_{11} = U_{10} - P_{6,1}$	$(2(n+k)/5 - 1)$
$U_{12} = U_2 + U_3$	$(2(n+k)/5 - 1)$
$U_{13} = U_1 + U_4$	$(2(n+k)/5 - 1)$
$U_{14} = U_2 + U_5$	$(2(n+k)/5 - 1)$
$C_1 = U_{12} + U_8$	$(2(n+k)/5 - 1)$
$C_2 = U_{13} - U_9$	$(2(n+k)/5 - 1)$
$C_3 = U_{12} - U_8$	$(2(n+k)/5 - 1)$
$C_4 = U_1 + U_6$	$(2(n+k)/5 - 1)$
$C_5 = U_{14} + U_{11}$	$(2(n+k)/5 - 1)$
$C_6 = U_{13} + U_9$	$(2(n+k)/5 - 1)$
$C_7 = U_{14} - U_{11}$	$(2(n+k)/5 - 1)$
$C = C_0 + C_1 x^{n/5} + ... + C_8 x^{8n/5}$	$8((n+k)/5 - 1)$
TOTAL:	$M_3(n) = 2M_3(\frac{n+k}{5}) + M_3(\frac{n-4k}{5}) + 3M_9(\frac{n+k}{5}) + \frac{72n}{5} + \frac{42k}{5} - 29$

Table 5.4: Costs of multi-evaluation and reconstruction for U1 in $\mathbb{F}_9$

Computations	Cost for Multiplication in $\mathbb{F}_9$
$R_1 = A_1 + A_3, R'_1 = B_1 + B_3$	$4(n+k)/5$
$R_2 = A_0 - A_4, R'_2 = B_0 - B_4$	$4(n-4k)/5$
$R_3 = A_0 + A_4, R'_3 = B_0 + B_4$	$4(n-4k)/5$
$R_4 = A_1 - A_3, R'_4 = B_1 - B_3$	$4(n+k)/5$
$R_5 = R_4 - A_2, R'_5 = R'_4 - B_2$	$4(n+k)/5$
$R_6 = -A_2 - R_4, R'_6 = -B_2 - R'_4$	$4(n+k)/5$
$R_7 = R_3 - A_2, R'_7 = R'_3 - B_2$	$4(n+k)/5$
$R_8 = R_1 + R_2, R'_8 = R'_1 + R'_2$	$4(n+k)/5$
$R_9 = R_2 - R_1, R'_9 = R'_2 - R'_1$	$4(n+k)/5$
$P_3 = R_{13}R'_{13}$	$M_9((n+k)/5)$
$P_4 = R_{14}R'_{14}$	$M_9((n+k)/5)$
$P_5 = R_{15}R'_{15}$	$M_9((n+k)/5)$
$P_6 = R_{16}R'_{16}$	$M_9((n+k)/5)$
$P_7 = R_{17}R'_{17}$	$M_9((n+k)/5)$
$P_8 = A_4B_4$	$M_9((n-4k)/5)$
$R_{10} = R_1 + R_3, R'_{10} = R'_1 + R'_3$	$4(n+k)/5$
$R_{11} = R_{10} + A_2, R'_{11} = R'_{10} + B_2$	$4(n+k)/5$
$R_{12} = R_7 + \omega R_4, R'_{12} = R'_7 + \omega R'_4$	$4(n+k)/5$
$R_{13} = R_7 - \omega R_4, R'_{13} = R'_7 - \omega R'_4$	$4(n+k)/5$
$R_{14} = R_8 + \omega R_5, R'_{14} = R'_8 + \omega R'_5$	$4(n+k)/5$
$R_{15} = R_8 - \omega R_5, R'_{15} = R'_8 - \omega R'_5$	$4(n+k)/5$
$R_{16} = R_9 + \omega R_6, R'_{16} = R'_9 + \omega R'_6$	$4(n+k)/5$
$R_{17} = R_9 - \omega R_6, R'_{17} = R'_9 - \omega R'_6$	$4(n+k)/5$
$P_0 = A_0B_0$	$M_9((n+k)/5)$
$P_1 = R_{11}R'_{11}$	$M_9((n+k)/5)$
$P_2 = R_{12}R'_{12}$	$M_9((n+k)/5)$
$U_1 = -P_0 + P_1$	$4(n+k)/5 - 2$
$U_2 = P_2 + P_3$	$4(n+k)/5 - 2$
$U_3 = P_2 - P_3$	$4(n+k)/5 - 2$
$U_4 = P_6 + P_7$	$4(n+k)/5 - 2$
$U_5 = P_6 - P_7$	$4(n+k)/5 - 2$
$U_6 = P_4 + P_5$	$4(n+k)/5 - 2$
$U_7 = P_4 - P_5$	$4(n+k)/5 - 2$
$U_8 = U_4 + U_6$	$4(n+k)/5 - 2$
$U_9 = U_1 - U_2$	$4(n+k)/5 - 2$
$U_{10} = U_9 + U_4$	$4(n+k)/5 - 2$
$U_{11} = U_{10} - P_8$	$4(n-4k)/5 - 2$
$U_{12} = -U_7 + U_5$	$4(n+k)/5 - 2$
$U_{13} = U_3 + U_{12}$	$4(n+k)/5 - 2$
$U_{14} = P_0 - U_2$	$4(n+k)/5 - 2$
$U_{15} = U_{14} + U_8$	$4(n+k)/5 - 2$
$U_{16} = U_{15} + P_8$	$4(n-4k)/5 - 2$
$U_{17} = -U_7 - U_5$	$4(n+k)/5 - 2$
$U_{18} = U_9 + U_6$	$4(n+k)/5 - 2$
$U_{19} = U_{18} - P_8$	$4(n-4k)/5 - 2$
$U_{20} = U_3 - U_{12}$	$4(n+k)/5 - 2$
$U_{21} = P_0 - U_8$	$4(n+k)/5 - 2$
$C_1 = U_{11} + \omega U_{13}$	$4(n+k)/5 - 2$
$C_2 = U_{16} + \omega U_{17}$	$4(n+k)/5 - 2$
$C_3 = U_{11} - \omega U_{13}$	$4(n+k)/5 - 2$
$C_4 = U_{21} + P_8$	$4(n-4k)/5 - 2$
$C_5 = U_{19} + \omega U_{20}$	$4(n+k)/5 - 2$
$C_6 = U_{16} - \omega U_{17}$	$4(n+k)/5 - 2$
$C_7 = U_{19} - \omega U_{20}$	$4(n+k)/5 - 2$
$C = C_0 + C_1x^{n/5} + ... + C_8x^{8n/5}$	$16((n+k)/5 - 1)$
TOTAL:	$M_9(n) = 8M_9(\frac{n+k}{5}) + M_9(\frac{n-4k}{5}) + \frac{196n}{5} + \frac{76k}{5} - 72$

CHAPTER 6

IMPROVEMENTS IN THE ARITHMETIC COMPLEXITIES AND THE IMPLEMENTATION RUN-TIMES BY THE NEW ALGORITHMS

6.1 Comparison of the Arithmetic Complexities for the Multiplication Algorithms

In this chapter, we examine the improvements in the arithmetic complexities for polynomial multiplication that comes with the new algorithms N1, N2, N3, V1, and U1. Table 6.1 displays the arithmetic complexities before and after the existence of N1, N2, N3, V1, and U1. Each algorithm is indicated as a number; 1 refers to the schoolbook method SB [4], 2 refers to the refined Karatsuba 2-way algorithm KA2 [4], 3 refers to the 3-way algorithm A3 in [17], 4 refers to the unbalanced refined Karatsuba algorithm UB [4, 23], 5 refers to schoolbook recursion LT [4], 6 refers to A2 [17], 7 refers to Bernstein's 3-way algorithm B1 [5], 8 refers to N1 [40], 9 refers to N2 [40], 10 refers to N3 [39], 11 refers to V1 [40], 12 refers to U11, 13 refers to U12, 14 refers to U13 and finally, 15 refers to U14. Note that U1 [39] refers to U11 for $k = 4$, U12 for $k = 3$, U13 for $k = 2$ and U14 for $k = 1$ in Table 6.1. By using the new algorithms N1, N2, N3, V1, and U1 combined with the other known ones, we get a 48.6% reduction in the arithmetic complexity for polynomial multiplication in $\mathbb{F}_9[x]$ and a 26.8% reduction for polynomial multiplication in $\mathbb{F}_3[x]$ for the input size $n = 1280$. Note that the N3 algorithm is better than N1 and N2 for all input sizes over $\mathbb{F}_3$. After the input size $n \geq 168$, both of N3 and U1 become better than the B1 algorithm over $\mathbb{F}_3$. Also, U1 is more efficient than N3 for $n \geq 772$ over $\mathbb{F}_3$.

Table 6.1: Improvements in the arithmetic complexities before and after the N1, N2, N3, V1, and U1 algorithms

	Previous $\mathbb{F}_3$ complexity			New $\mathbb{F}_3$ complexity			Previous $\mathbb{F}_9$ complexity			New $\mathbb{F}_9$ complexity		
n	$M_3(n)$	*Alg.	Split	$M_3(n)$	*Alg.	Split	$M_9(n)$	*Alg.	Split	$M_9(n)$	*Alg.	Split
1	1	1	-	1	1	-	6	1	-	6	1	-
2	5	1	-	5	1	-	26	2	1	26	2	1
3	13	1	-	13	1	-	60	6	3	60	6	3
4	25	1	-	25	1	-	100	2	2	100	2	2
5	41	1	-	41	1	-	160	6	5	160	6	5
6	57	2	3	57	2	3	216	6	6	216	6	6
7	81	5	6	81	5	6	296	6	7	296	6	7
8	100	2	4	100	2	4	350	2	4	350	2	4
9	132	5	8	132	5	8	456	3	3	456	3	3
10	155	2	5	155	2	5	542	6	10	542	6	10
11	189	4	5-6	189	4	5-6	652	6	11	652	6	11
12	210	2	6	210	2	6	716	3	4	716	3	4
13	258	5	12	258	5	12	875	6	13	875	6	13
14	289	2	7	289	2	7	976	6	14	976	6	14
15	329	4	7-8	329	4	7-8	1076	3	5	**1056**	11	3
25	807	5	24	807	5	24	2594	5	24	**2348**	11	5
28	962	2	14	962	2	14	3107	6	28	**2884**	13	4-6
30	1089	2	15	1089	2	15	3286	3	10	**3048**	11	6
31	1139	4	15-16	1139	4	15-16	3592	4	15-16	**3532**	5	30
40	1733	2	20	1733	2	20	5516	6	40	**4646**	11	8
60	3474	2	30	3474	2	30	9941	3	20	**8724**	11	12
64	3725	2	32	3725	2	32	11500	2	32	**10156**	9	16
98	7810	2	49	**7755**	13	18-20	23144	2	49	**18112**	13	18-20
99	7919	7	33	**7809**	12	19-20	21436	3	33	**18268**	12	19-20
100	8126	2	50	**7843**	11	20	25072	2	50	**18356**	11	20
125	11446	4	62-63	**11236**	11	25	33298	4	62-63	**25960**	11	25
128	11620	2	64	11620	2	64	35390	2	64	**28382**	9	32
256	35753	2	128	**33737**	10	64	107956	2	128	**77173**	15	48-52
360	57327	7	120	**54829**	11	72	156956	3	120	**121680**	11	72
509	107890	4	254-255	**93376**	12	101-102	299852	4	254-255	**203024**	12	101-102
510	104972	7	170	**93457**	11	102	294181	3	170	**203484**	11	102
512	109048	2	256	**94050**	14	100-103	327446	2	256	**202816**	14	100-103
625	145100	5	624	**120559**	11	125	431144	5	624	**258068**	11	125
653	160648	4	326-327	**135827**	13	129-131	411822	4	326-327	**292520**	13	129-131
655	157250	5	654	**136477**	11	131	409844	5	654	**292823**	11	131
677	168617	4	338-339	**140855**	14	133-136	476832	4	338-339	**301176**	14	133-136
701	175656	4	350-351	**145051**	15	137-141	475082	4	350-351	**311087**	15	137-141
704	187894	2	352	**145243**	12	140-141	569311	6	704	**311768**	12	140-141
761	197651	4	380-381	**168505**	15	149-153	546488	4	380-381	**360395**	15	149-153
765	187417	7	255	**169795**	11	153	504131	3	255	**364536**	11	153
768	190016	7	256	**170040**	10	192	555116	3	256	**363536**	9	192
821	229346	4	410-411	**185017**	15	161-165	643692	4	410-411	**395006**	15	161-165
825	215932	7	275	**185824**	11	165	600876	3	275	**396012**	11	165
857	245673	4	428-429	**192872**	14	169-172	655785	4	428-429	**409884**	14	169-172
860	259450	2	430	**193294**	11	172	683483	2	430	**410614**	11	172
953	286446	4	476-477	**221236**	13	189-191	781534	4	476-477	**468628**	13	189-191
955	267375	5	954	**221587**	11	191	739979	5	954	**469211**	11	191
1000	325628	2	500	**229081**	11	200	932218	2	500	**485366**	11	200
1013	309843	4	506-507	**250237**	13	201-203	860247	4	506-507	**530886**	13	201-203
1015	302585	5	1014	**251692**	11	203	864854	5	1014	**532952**	11	203
1024	330725	2	512	**254553**	12	204-205	989500	2	512	**539582**	12	204-205
1277	443737	4	638-639	**351406**	14	253-256	1193327	4	638-639	**743940**	14	253-256
1280	479836	2	640	**351133**	11	256	1449745	6	1280	**744661**	11	256
1284	451169	7	428	**353092**	12	256-257	1221691	3	428	**746073**	12	256-257
1300	471848	2	650	**357889**	11	260	1185742	2	650	**755786**	11	260
1320	465857	7	440	**363961**	11	264	1314676	3	440	**767478**	11	264

*Algorithms: 1=SB, 2=KA2, 3=A3, 4=UB, 5=LT, 6=A2, 7=B1, 8=N1, 9=N2, 10=N3, 11=V1, 12-15=U11-U14

6.2 Comparison of Implementation Results for the Multiplication Algorithms

In this section, we report the implementation results for all of the aforementioned characteristic three polynomial multiplication algorithms for relatively small-sized inputs and compare them through Table 6.2 and Table 6.3. Note that while implementing the polynomial multiplication algorithms, we recursively call the fastest method at each input size level beginning from the small sizes to larger input sizes. Therefore, in Table 6.2, we report the implementation run-times for polynomial multiplications with relatively small-sized inputs over $\mathbb{F}_3$ and in Table 6.3 we report the implementation run-times for the polynomial multiplications over $\mathbb{F}_9$ with the same small-sized inputs. These small values are to be used in the recursive implementation of the larger-sized inputs.

Note that U1 becomes faster than B1, N1, N2, and N3 after $n \geq 96$ over $\mathbb{F}_3$ in terms of implementation cycle counts. Also, N1, N2, and N3 is faster than the B1 3-way algorithm for $n \geq 192$ over $\mathbb{F}_3$. In general, B1 3-way algorithm is faster than A3 3-way method and N3 4-way algorithm is slower than N1 and N2 4-way methods over $\mathbb{F}_3$. In $\mathbb{F}_9[x]$, the A3 3-way algorithm is faster than B1 and N2 4-way algorithm is faster than N1 and N3 in general. We observe from Table 6.1, Table 6.2, and Table 6.3 that U1 yields the most efficient results of all algorithms in terms of both arithmetic complexity and implementation run-time.

Benchmark tests for all implementations are performed on an Intel (R) Core (TM) i7-10510U processor running at 1.80GHz. The operating system is Ubuntu 20.04.3 LTS and Linux Kernel 5.11.0. All software is compiled with gcc-9.3.0. We report the median of 1000,000 executions of the corresponding algorithm for the cycle counts.

Table 6.2: Implementation speeds (cycles) for the multiplication algorithms over $\mathbb{F}_3$

Size	**Polynomial Multiplication Algorithms in $\mathbb{F}_3$**													
n	**SB**	**KA2**	**A3**	**B1**	**N1**	**N2**	**N3**	**V1**	**U11**	**U12**	**U13**	**U14**	**UB**	**LT**
1	**64**	-	-	-	-	-	-	-	-	-	-	-	-	-
2	**105**	225	-	-	-	-	-	-	-	-	-	-	-	-
3	**150**	-	1065	526	-	-	-	-	-	-		-	270	163
4	**188**	307	-	-	820	834	-	-	-	-	-	-	-	-
5	**247**	-	-	-	-	1141	-	-	-	-	-	-	502	291
6	**328**	549	1387	824	-	-	-	-	-	-	-	-	-	-
7	**359**	-	-	-	-	-	-	-	-	-	-	-	708	453
8	**499**	734	-	-	1456	1429	1593	-	-	-	-	-	-	-
9	**609**	-	1664	1303	-	-	-	-	-	-	-	-	965	702
10	**716**	989	-	-	-	-	-	1913	-	-	-	-	-	-
11	**854**	-	-	-	-	-	-	-	-	-	-	-	1115	953
12	**933**	1301	2005	1678	2078	2053	2292	-	-	-	-	-	-	-
13	**1020**	-	-	-	-	-	-	-	-	-	-	-	1470	1131
14	**1233**	1499	-	-	-	-	-	-	-	-	-	-	-	-
15	**1384**	-	2499	2089	-	-	-	2644	-	-	-	-	1745	1522
16	**1564**	1688	-	-	2610	2070	3057	-	-	-	-	-	-	-
17	**1694**	-	-	-	-	-	-	-	-	-	3579	-	1979	1828
18	**1873**	2024	3045	2560	-	-	-	-	-	3609	-	-	-	-
19	**1984**	-	-	-	-	-	-	-	3613	-	-	-	2380	2185
20	**2192**	2408	-	-	3531	3506	3912	3582	-	-	-	-	-	-
21	**2496**	-	3504	3002	-	-	-	-	-	-	-	4548	2743	2599
22	**2691**	2746	-	-	-	-	-	-	-	-	4584	-	-	-
23	**2802**	-	-	-	-	-	-	-	-	4672	-	-	3121	3078
24	3323	**3132**	4163	3609	4427	4303	4837	-	4752	-	-	-	-	-
25	3536	-	-	-	-	-	-	4617	-	-	-	-	**3496**	3681
26	3792	**3642**	-	-	-	-	-	-	-	-	-	4544	-	-
27	**3985**	-	4752	4312	-	-	-	-	-	-	5669	-	3994	4162
28	4260	**4156**	-	-	5323	5349	5902	-	-	5795	-	-	-	-
29	4739	-	-	-	-	-	-	-	5834	-	-	-	**4490**	4656
30	4984	**4584**	5513	4989	-	-	-	5845	-	-	-	-	-	-
31	5310	-	-	-	-	-	-	-	-	-	-	6825	**5096**	5197
32	5676	**5032**	-	-	6333	6260	6893	-	-	-	6978	-	-	-
33	5953	-	6184	5689	-	-	-	-	-	6922	-	-	**5582**	5682
34	6299	**5610**	-	-	-	-	-	-	6995	-	-	-	-	-
35	6655	-	-	-	-	-	-	7061	-	-	-	-	**6007**	6295
36	7114	**6197**	7036	6535	7347	7280	7975	-	-	-	-	8076	-	-
37	7343	-	-	-	-	-	-	-	-	-	8301	-	**6769**	6804
38	7671	**6753**	-	-	-	-	-	-	-	8195	-	-	-	-
39	8068	-	7892	7273	-	-	-	-	8299	-	-	-	**7229**	7413
40	8566	**7366**	-	-	8583	8393	9115	8294	-	-	-	-	-	-
41	8707	-	-	-	-	-	-	-	-	-	-	9416	**7961**	8050
42	9458	**8070**	8703	8288	-	-	-	-	-	-	9454	-	-	-
43	10008	-	-	-	-	-	-	-	-	9543	-	-	**8724**	8868
44	10149	**8821**	-	-	9630	9532	10440	-	9682	-	-	-	-	-
45	10724	-	9549	**9245**	-	-	-	9791	-	-	-	-	9345	9458
46	11266	**9679**	-	-	-	-	-	-	-	-	-	10898	-	-
47	11825	-	-	-	-	-	-	-	-	-	11028	-	10651	**10314**
48	12152	10832	10647	**10070**	11014	10794	12022	-	-	11006	-	-	-	-
49	12789	-	-	-	-	-	-	-	11256	-	-	-	11737	**11069**
50	13332	11862	-	-	-	-	-	**11277**	-	-	-	-	-	-
51	13689	-	11802	**11015**	-	-	-	-	-	-	-	12601	12284	12246
52	14433	**12065**	-	-	12374	12259	13114	-	-	-	12394	-	-	-
53	14887	-	-	-	-	-	-	-	-	**12700**	-	-	12757	12932
54	15474	12688	12877	**12071**	-	-	-	-	12869	-	-	-	-	-
55	15950	-	-	-	-	-	-	**12975**	-	-	-	-	13466	13182
56	16703	13628	-	-	13692	**13475**	14899	-	-	-	-	14096	-	-
57	17313	-	13776	**13235**	-	-	-	-	-	-	14377	-	14642	14956
58	17760	14944	-	-	-	-	-	-	-	**14456**	-	-	-	-
59	18380	-	-	-	-	-	-	-	**14458**	-	-	-	15390	15647
60	19156	15162	15530	**14242**	15530	14810	16166	14648	-	-	-	-	-	-
96	47552	32518	31429	31426	31961	31441	34140	-	-	-	-	**30945**	-	-
192	184396	100339	89060	94226	85243	86121	92322	-	-	-	**83856**	-	-	-

Table 6.3: Implementation results (cycles) for the multiplication algorithms over $\mathbb{F}_9$

Size	**Polynomial Multiplication Algorithms in $\mathbb{F}_9$**													
n	**SB**	**KA2**	**A3**	**B1**	**N1**	**N2**	**N3**	**V1**	**U11**	**U12**	**U13**	**U14**	**UB**	**LT**
1	**45**	-	-	-	-	-	-	-	-	-	-	-	-	-
2	**104**	288	-	-	-	-	-	-	-	-	-	-	-	-
3	**233**		1158	727	-	-	-	-	-	-		-	432	276
4	**341**	564	-	-	1088	1034	-	-						
5	**483**	-	-	-	-	-	-	1251	-	-	-	-	731	552
6	**634**	896	1664	1325										
7	**805**	-	-	-	-	-	-	-	-	-	-	-	1114	924
8	**1035**	1265	-	-	2014	1873	2237	-	-	-	-	-	-	-
9	**1253**	-	2104	1954	-	-	-	-	-	-	-	-	1497	1288
10	**1454**	1672	-	-	-	-	-	2459	-	-	-	-	-	-
11	**1627**	-	-	-	-	-	-	-	-	-	-	-	1896	1842
12	**1937**	2074	2870	2641	3031	2835	3319							
13	**2182**	-	-	-	-	-	-	-	-	-	-	-	2427	2373
14	**2458**	2604	-	-	-	-	-	-	-	-	-	-	-	-
15	**2816**	-	3572	3480	-	-	-	3750	-	-	-	-	3010	3021
16	**3140**	3257	-	-	4261	4040	4583	-	-	-	-	-	-	-
17	**3584**	-	-	-	-	-	-	-	-	-	5661	-	3627	3821
18	**3948**	3978	4535	4585	-	-	-	-	-	5259	-	-	-	-
19	**4356**	-	-	-	-	-	-	-	5335	-	-	-	4377	4591
20	4756	**4679**	-	-	5523	5278	6197	5290	-	-	-	-	-	-
21	5278	-	5504	5491	-	-	-	-	-	-	-	7068	**5087**	5503
22	5759	**5483**	-	-	-	-	-	-	-	-	7172	-	-	-
23	6209	-	-	-	-	-	-	-	-	6876	-	-	**6000**	6270
24	6787	**6172**	6621	6706	6945	6612	7454	-	6870	-	-	-	-	-
25	7257	-	-	-	-	-	-	7090	**6779**	7115				
26	7824	**7113**	-	-	-	-	-	-	-	-	-	8843	-	-
27	8256	-	**7723**	7864	-	-	-	-	-	-	8552	-	7779	8075
28	8975	**8070**	-	-	8413	8109	9235	-	-	8609	-	-	-	-
29	9642	-	-	-	-	-	-	-	8786	-	-	-	**8609**	9122
30	10412	9103	9032	9302	-	-	-	**8820**	-	-	-	-	-	-
31	10904	-	-	-	-	-	-	-	-	-	-	10545	**9829**	9943
32	11690	10119	-	-	10082	**9813**	11046	-	-	-	10494	-	-	-
33	12298	-	**10276**	10559	-	-	-	-	-	10743	-	-	10697	10862
34	13091	11342	-	-	-	-	-	-	**10921**	-	-	-	-	-
35	13797	-	-	-	-	-	-	**10916**	-	-	-	-	12266	12679
36	14604	12282	11740	12167	12096	**11682**	13129	-	-	-	-	12647	-	-
37	15491	-	-	-	-	-	-	-	-	-	**12790**	-	13206	13203
38	16038	13059	-	-	-	-	-	-	-	**12897**	-	-	-	-
39	17026	-	13543	14181	-	-	-	-	**13200**	-	-	-	14510	15282
40	18082	15150	-	-	14191	13708	15000	**13140**	-	-	-	-	-	-
41	18891	-	-	-	-	-	-	-	-	-	-	14997	16003	**14563**
42	19562	16160	15297	15520	-	-	-	-	-	-	**14964**	-	-	-
43	20691	-	-	-	-	-	-	-	-	**15076**	-	-	17128	16523
44	21520	17450	-	-	16365	1608	17333	-	**15471**	-	-	-	-	-
45	22418	-	17063	17684	-	-	-	**15544**	-	-	-	-	18553	17173
46	23681	18755	-	-	-	-	-	-	-	-	-	**17550**	-	-
47	24510	-	-	-	-	-	-	-	-	-	**17519**	-	19696	19284
48	25679	20148	18969	19469	18561	18051	19815	-	-	**17798**	-	-	-	-
49	26609	-	-	-	-	-	-	-	**18017**	-	-	-	21451	19762
50	27885	21910	-	-	-	-	-	**18384**	-	-	-	-	-	-
51	28987	-	21096	21876	-	-	-	-	-	-	-	20667	22851	**20427**
52	30435	23326	-	-	21321	**20637**	22677	-	-	-	20747	-	-	-
53	31870	-	-	-	-	-	-	-	-	**20785**	-	-	24611	22712
54	33197	25397	23499	24021	-	-	-	-	**20959**	-	-	-	-	-
55	33563	-	-	-	-	-	-	**21124**	-	-	-	-	25992	22821
56	34857	26330	-	-	23790	23449	25361	-	-	-	-	**23312**	-	-
57	36465	-	25827	26358	-	-	-	-	-	-	**23881**	-	27223	25155
58	37750	28320	-	-	-	-	-	-	-	**24116**	-	-	-	-
59	39145	-	-	-	-	-	-	-	**23808**	-	-	-	29061	26338
60	40722	28478	28381	28787	26582	26154	28352	**24470**	-	-	-	-	-	-

CHAPTER 7

APPLICATION OF THE NEW ALGORITHMS TO NTRU PRIME DECAPSULATION AND THE IMPLEMENTATION RESULTS

The decapsulation stage of the Streamlined NTRU Prime Key Encapsulation Mechanism (KEM) conducts a characteristic three polynomial multiplication operation to multiply the elements of $\mathbb{Z}_3[x]/(x^p - x - 1)$ for the parameters $p = 653, 761, 857, 953, 1013, 1277$. Thus, we can apply the proposed 4-way and 5-way polynomial multiplication algorithms N1, N2, N3, V1, and U1 into this step. (Step 5 of Algorithm 3 in Chapter 2). Bernstein *et al.* propose a hybrid characteristic three polynomial multiplication algorithm for the NTRU Prime decapsulation in [7] and later they modified the protocol with an improved hybrid characteristic three polynomial multiplication method in [5]. In their recent submissions [8, 9] to NIST, they also modify the input sizes and added new parameters to the NTRU Prime protocol. One can refer to [5–9] for the full explanation of the NTRU Prime protocol and the recent modifications on it. In this chapter, we analyze these two different hybrid methods that are used in the characteristic three polynomial multiplication step of NTRU Prime decapsulation by Bernstein *et al.*, then we propose new hybrid methods to be used in the same multiplication step and report the comparison results for all of these methods.

7.1 Bernstein's Hybrid-1 Multiplication Algorithm: 5 KA2 then SB (n=768)

In [7], Bernstein *et al.* use a combination of five layers of KA2 and then SB for multiplying the input size $n = 768$ polynomials for the Streamlined NTRU Prime

decapsulation phase. We call this algorithm as Hybrid-1 and our first purpose is to construct a faster alternative to this one. We show that the new hybrid algorithms containing N1, N2, and V1 provide better results than that of Hybrid-1.

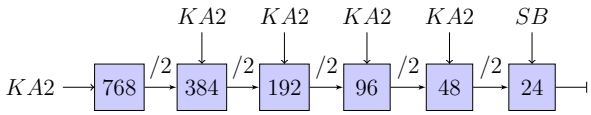

Figure 7.1: Hybrid-1 Algorithm requires a total # of $303,600$ arithmetic operations

Figure 7.1 is a visual representation of Hybrid-1 after each recursive call for the combining functions of it. It ends up with the non-recursive SB algorithm. By SB, we refer to the schoolbook-comba multiplication algorithm [29]. Hybrid-1 costs a total of $303,600$ arithmetic operations including both multiplications and summations.

7.2 Constructing New Hybrid Algorithms Using N1, N2, and V1 to Replace Bernstein's Hybrid-1 Algorithm

In this section, we construct different combined versions of the new algorithms N1, N2, and V1 along with KA2, A2, A3, UB, LT, and SB so that they provide efficiency gain compared to the characteristic three polynomial multiplication method that is described in [7] for the Streamlined NTRU Prime Decapsulation.

7.2.1 Hybrid-2 Multiplication Algorithm: 8 KA2 then SB (n=768)

Hybrid-2 uses eight layers of KA2 and then SB, for $n = 768$ as represented in Figure 7.2. In the absence of N1, N2, and V1, Hybrid-2 is the most arithmetically efficient combination of all other algorithms. It costs $207,858$ total number of operations which is better than Hybrid-1 and all other possible combinations. We aim to construct arithmetically cheaper hybrid algorithms than Hybrid-2 by using new N1, N2, and V1 algorithms.

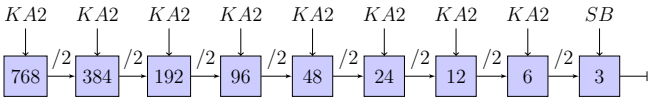

Figure 7.2: Hybrid-2 Algorithm requires a total # of $207,858$ arithmetic operations

It's important to point out that, the arithmetic complexity and the implementation

complexity may not always be compatible. Hybrid-1 and Hybrid-2 can be an example of this situation. Even though Hybrid-2 is arithmetically more efficient than Hybrid-1, when it comes to the software implementation, the cycle count and the runtime of Hybrid-2 is much worse than that of Hybrid-1. This is the reason why Bernstein *et al.* [6–8] selected Hybrid-1 over Hybrid-2 for the multiplication in NTRU Prime.

7.2.2 N1-Hybrid Multiplication Algorithm (n=768)

N1-Hybrid is the first hybrid algorithm that our new method N1 is used in. It takes a total number of $187,152$ arithmetic operations to multiply the polynomials of input sizes $n = 768$. According to Table 7.1 the reduction percentage in the arithmetic cost compared with the reference Hybrid-1 method is 38.35%.

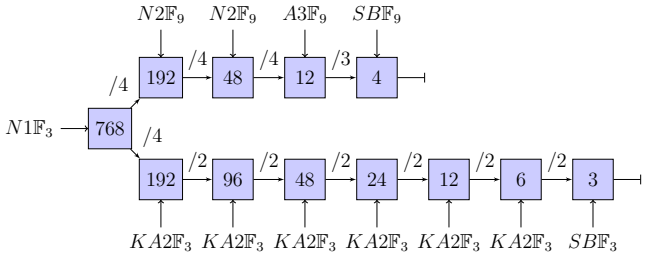

Figure 7.3: N1-Hybrid Algorithm requires a total # of $187,152$ arithmetic operations

Figure 7.3 is a visual representation of N1-Hybrid. Note that the branches represent the multiplications that are performed in $\mathbb{F}_3[x]$ and $\mathbb{F}_9[x]$, respectively.

Remark 3 To obtain a reduced cycle count in the implementation, we modify the N1-Hybrid algorithm in a way that it calls $SB\mathbb{F}_9$ at $n = 12$ instead of $A3\mathbb{F}_9$ in $\mathbb{F}_9[x]$ (See the upper branch of the Figure 7.3). We also make it call $SB\mathbb{F}_3$ at $n = 48$ instead of $KA2\mathbb{F}_3$ in $\mathbb{F}_3[x]$ (See the lower branch of the Figure 7.3). We call this new hybrid method **N1-Hybrid2**.

7.2.3 N2-Hybrid Multiplication Algorithm (n=768)

Recall that the N2 Algorithm is an improved version of N1. Multiplying polynomials in $\mathbb{Z}_3[x]$ for $n = 768$ takes $180,878$ total number of operations with N2-Hybrid which is visually represented in the Figure 7.4. Table 7.1 indicates that the reduction

percentage that comes with N2-Hybrid in arithmetic cost is 40.42% compared to the reference Hybrid-1 algorithm.

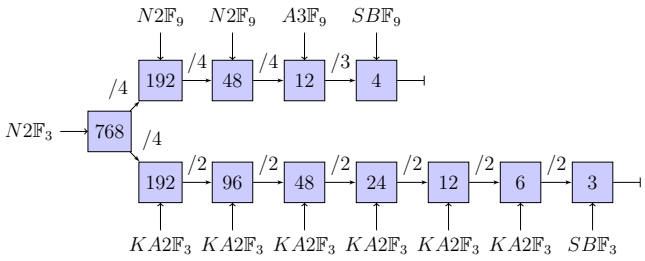

Figure 7.4: N2-Hybrid Algorithm requires a total # of $180,878$ arithmetic operations

7.2.4 V1-Hybrid Multiplication Algorithm (n=765)

This hybrid algorithm contains the new 5-way split multiplication algorithm V1 in it.

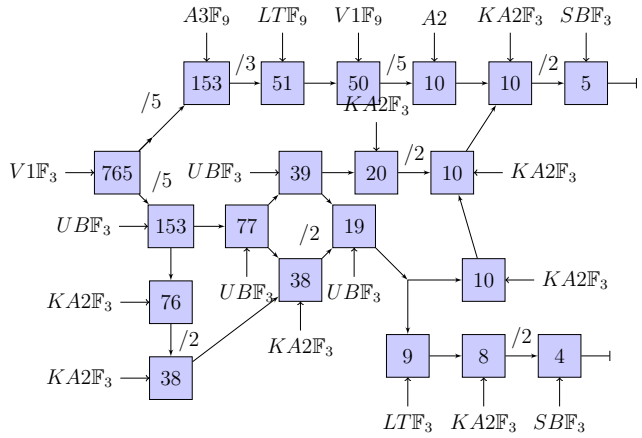

Figure 7.5: V1-Hybrid Algorithm requires a total # of $182,647$ arithmetic operations

Notice that 761 is not divisible by 5 that's why we pick $n = 765$ (zero-padded from the 761-coefficient inputs) as for the input sizes of the polynomials. Figure 7.5 displays the flowchart representation of V1-Hybrid for $n = 765$. The total arithmetic operation cost for this algorithm is $182,647$. One can convey from Table 7.1 that the reduction percentage that comes with V1-Hybrid in the arithmetic cost is 39.83% compared to the reference Hybrid-1 method.

7.2.5 A3-Hybrid Multiplication Algorithm (n=768)

As represented in the Figure 7.6, A3-Hybrid algorithm has a total arithmetic cost of $189,115$. One can convey from Table 7.1 that the reduction percentage that comes

with A3-Hybrid in the arithmetic cost is 37.70% compared to the reference Hybrid-1 algorithm.

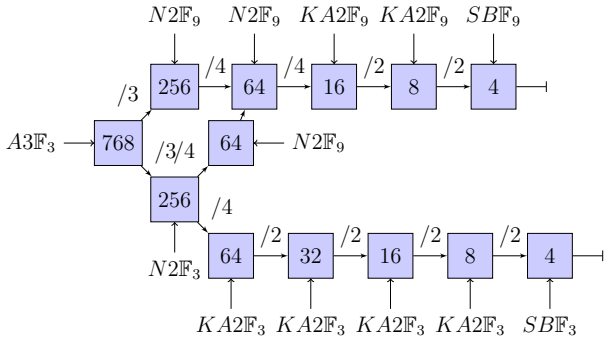

Figure 7.6: A3-Hybrid Algorithm requires a total # of $189,115$ arithmetic operations

Remark 4 To achieve more efficiency in the implementation rather than the arithmetic operations, we modify the A3-Hybrid algorithm so that it calls $SB\mathbb{F}_9$ at $n = 16$ instead of $KA2\mathbb{F}_9$ in $\mathbb{F}_9[x]$ (See the upper branch of the Figure 7.6). Yet again we make it call $SB\mathbb{F}_3$ at $n = 32$ instead of $KA2\mathbb{F}_3$ in $\mathbb{F}_3[x]$ (See the lower branch of the Figure 7.6). We call this new hybrid method **A3-Hybrid2**.

7.2.6 LT-Hybrid Multiplication Algorithm (n=761)

For the LT algorithm we just pick $n = 761$ so that no padding would be necessary for the input polynomials. Observe from Table 7.1 that LT-Hybrid has an arithmetic complexity of $186,914$ which provides a reduction percentage of 38.43% compared with the reference Hybrid-1 method. Figure 7.7 shows the steps of LT-Hybrid for $n = 761$.

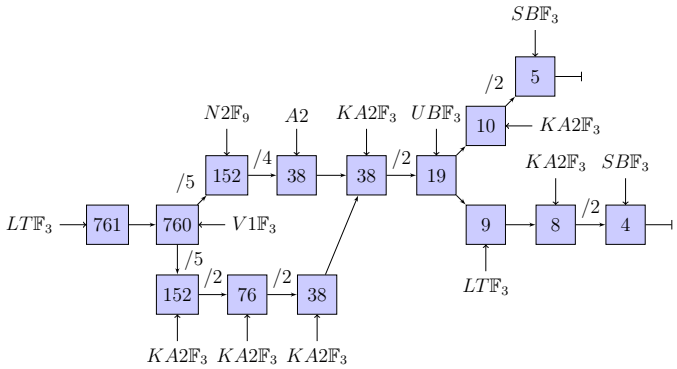

Figure 7.7: LT-Hybrid Algorithm requires a total # of $186,914$ arithmetic operations

Benchmark tests for all implementations are performed on an Intel (R) Core (TM)

i7-9750H processor running at 2600 MHz. The operating system is Ubuntu 20.04.1 LTS and Linux Kernel 5.4.0. All software is compiled with gcc-9.3.0. Bernstein *et al.* implemented the Hybrid-1 algorithm in Haswell x86 architecture using AVX2 vector instructions in the NTRU Prime Protocol. For the comparison purposes, we implement all of the new hybrid methods along with Hybrid-1 and Hybrid-2 in C without using AVX/AVX2 instructions. We report the median of 1000,000 executions of the corresponding algorithm for the cycle counts.

Table 7.1: Comparison of the arithmetic complexities of the new hybrid algorithms including N1, N2, and V1 / candidates for the Streamlined NTRU Prime KEM

n	Algorithm	Arithmetic Cost	Improvement	Source
768	Hybrid-1	$303,600$	**reference**	method used in `sntrup761` [7]
768	Hybrid-2	$207,858$	31.53%	min. cost: **before this thesis**
768	A3-Hybrid	$189,115$	37.70%	this thesis [40]
768	N1-Hybrid	$187,152$	38.35%	this thesis [40]
761	LT-Hybrid	$186,914$	38.43%	this thesis [40]
765	V1-Hybrid	$182,647$	39.83%	this thesis [40]
768	N2-Hybrid	$180,878$	40.42%	min. cost: **after this thesis [40]**

As Table 7.1 indicates all of the proposed hybrid methods are arithmetically more efficient than Bernstein's Hybrid-1 method. For instance, N2-Hybrid provides up to a 40.42% reduction in the arithmetic complexity compared to Hybrid-1 approach.

Table 7.2: Implementation results of the new hybrid algorithms including N1, N2, and V1 / candidates for the Streamlined NTRU Prime KEM (without AVX/AVX2)

Algorithm	n	Cycle Count	Time (s)	Improvement
Hybrid-1	768	$481,688$	0.000186	method used in `sntrup761` [7]
Hybrid-2	768	$2,028,918$	0.000783	-
LT-Hybrid	761	$1,312,231$	0.000506	-
N2-Hybrid	768	$758,611$	0.000293	-
V1-Hybrid	765	$561,386$	0.000217	-
A3-Hybrid	768	$469,257$	0.000181	2.58%
N1-Hybrid	768	$456,071$	0.000176	5.31%
A3-Hybrid2	768	$317,692$	0.000123	34.04%
N1-Hybrid2	768	$301,571$	0.000116	37.39%

It's concluded from Table 7.2 that the N1-Hybrid, A3-Hybrid, N1-Hybrid2, and the A3-Hybrid2 algorithms are all faster than the Hybrid-1 algorithm which is the one used for NTRU Prime protocol in [7]. Although it is not the most arithmetically efficient one, the N1-Hybrid2 algorithm is the fastest of all, with a significant reduction percentage of 37.39% in the cycle count. The A3-Hybrid2 algorithm is again much faster than the Hybrid-1 method with a reduction percentage of 34.04%.

7.3 Bernstein's B1-Hybrid1 Multiplication Algorithm (n=653)

In [5], using the B1 3-way algorithm, Bernstein *et al.* proposed a different hybrid method than the Hybrid-1 algorithm. This new method suggests using B1 3-way algorithm [5] once then some variants of KA2 and SB algorithms. We call this new method B1-Hybrid1 for the input size $n = 653$. To construct this method, we need to use the B1 3-way algorithm at the first level with the zero-padded parameter 654 (since it is divisible by 3). Then, at the lower levels, we use Karatsuba 2-way algorithm KA2 for the parameters which are divisible by 2 whereas unbalanced refined Karatsuba 2-way algorithm UB [4,23] for the odd parameters. We use the schoolbook method SB for the parameters which are less than or equal to 16.

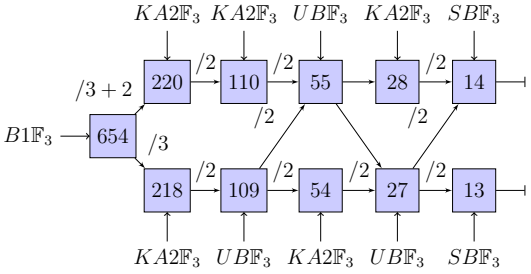

Figure 7.8: B1-Hybrid1 Algorithm cycles/time is $758,027/0.000329$

Figure 7.8 displays the flowchart representation of B1-Hybrid1 and the cycle count.

7.4 Bernstein's B1-Hybrid2 Multiplication Algorithm (n=761)

This algorithm is also introduced in [5] by Bernstein *et al.* and it applies the B1 3-way approach as well, but this time the input parameter is $n = 761$. Since 761 is not divisible by 3, we pick the input parameter 768 (zero-padded from the 761 coefficient

inputs). Figure 7.9 gives the visual representation of B1-Hybrid2 and the cycle count.

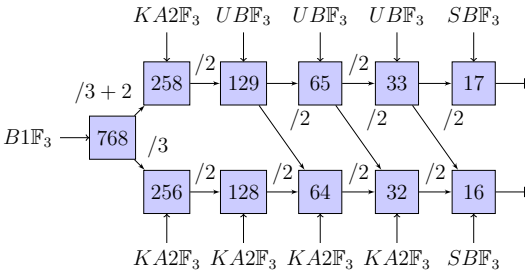

Figure 7.9: B1-Hybrid2 Algorithm cycles/time is $944,139/0.000410$

7.5 Constructing New Hybrid Algorithms Using N1, N2, N3, V1, and U1 to Replace Bernstein's B1-Hybrid1 and B1-Hybrid2 Algorithms

7.5.1 U1-Hybrid1 Multiplication Method (n=653)

In this section we propose a new hybrid multiplication algorithm U1-Hybrid1 which uses the unbalanced 5-way method U1 in combination with other methods and it is designed for the input parameter $n = 653$. Figure 7.10 displays a visual scheme for the U1-Hybrid1 and indicates the cycle count. Note that the branches represent the multiplications that are performed in $\mathbb{F}_3[x]$ and $\mathbb{F}_9[x]$ respectively. Table 7.3 indicates that, with the usage of U1-Hybrid1, the reduction percentage in the cycle count is 29.85% compared to Bernstein's reference B1-Hybrid1 method.

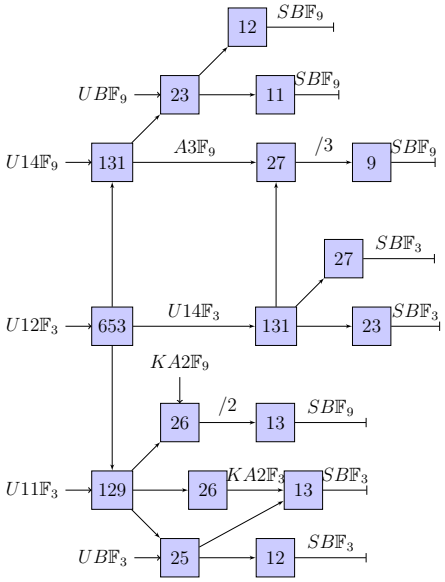

Figure 7.10: U1-Hybrid1 Algorithm cycle/time is $531,692/0.000231$

7.5.2 U1-Hybrid2 Multiplication Algorithm (n=761)

As a second application of the unbalanced 5-way algorithm U1, we introduce and implement another hybrid multiplication algorithm U1-Hybrid2 for the input size $n = 761$.

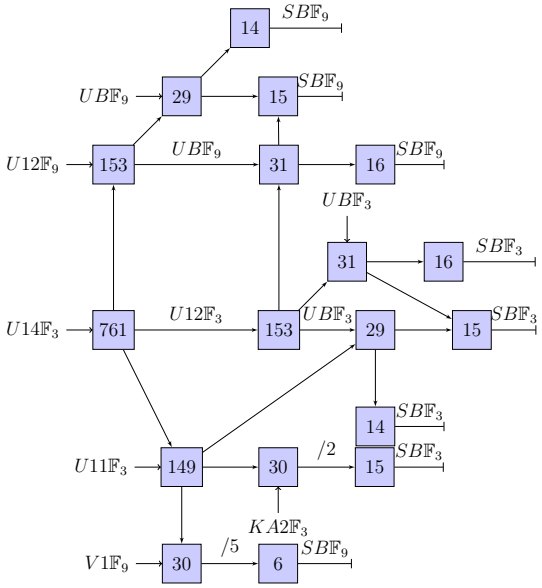

Figure 7.11: U1-Hybrid2 Algorithm cycle/time is $608,694/0.0000265$

Table 7.3 indicates that using U1-Hybrid2, the reduction percentage in the cycle count is 35.52% compared with Bernstein's reference B1-Hybrid2 algorithm. See Figure 7.11 for the flowchart and the implementation cycle count of U1-Hybrid2.

Below we give the implementation results for the new algorithms U1-Hybrid1 and U1-Hybrid2 which are constructed by using a hybrid recursive call technique [4, 11, 12, 17, 23], in C language, without using AVX instructions. Thus, for comparison purposes, we implement Bernstein's B1-Hybrid1 and B1-Hybrid2 approaches in C without using AVX instructions as well. Benchmark tests for all implementations are performed on an Intel (R) Core (TM) i7-10510U processor running at 1.80GHz. The operating system is Ubuntu 20.04.3 LTS and Linux Kernel 5.11.0. All software is compiled with gcc-9.3.0. We report the median of 1000,000 executions of the corresponding algorithm for the cycle counts.

Table 7.3: Implementation results for polynomial multiplication over $\mathbb{F}_3$ in Streamlined NTRU Prime Decapsulation

Parameter	Algorithm	Cycles/Time	Improvement
sntrup653	**B1-Hybrid1** (Bernstein's latest [5,9])	$758,027/0.000329$	**reference value**
	U1-Hybrid1 (this thesis [39])	$531,692/0.000231$	29.85%
sntrup761	**Hybrid-1** (Bernstein's previous [6–8])	$1,054,828/0.000458$	-10.49%
	B1-Hybrid2 (Bernstein's latest [5,9]	$944,139/0.000410$	**reference value**
	N1-Hybrid2 (this thesis [40])	$665,729/0.0000289$	29.48%
	U1-Hybrid2 (this thesis [39])	$608,694/0.0000265$	35.52%

As Table 7.3 indicates the new hybrid multiplication algorithm U1-Hybrid1 that we suggest for the input size $n = 653$ is 29.85% faster than B1-Hybrid1 and the new hybrid algorithm U1-Hybrid2 that we suggest for the input size $n = 761$ is 35.52% faster than the B1-Hybrid2 in terms of the implementation cycle count. Also, Table 7.3 shows that U1-Hybrid2 is 8.56% faster than N1-Hybrid2 method for the input size $n = 768$ (zero padded to 761 coefficient inputs).

CHAPTER 8

CONCLUSION

We show that the new 4-way and 5-way split polynomial multiplication algorithms N1, N2, N3, V1 and the unbalanced 5-way algorithm U1 introduced in this thesis are more efficient than the current state-of-the-art algorithms over the characteristic three fields $\mathbb{F}_3$ and $\mathbb{F}_9$ (the finite field with nine elements). The proposed algorithms outperform the current ones in both arithmetical and implementational perspectives. We present the improvements in the arithmetic complexities before and after the proposed algorithms in Table 6.1 and observe that with the presence of the new N1, N2, N3, V1, and U1 algorithms the reduction percentage regarding the arithmetic complexity for polynomial multiplication over $\mathbb{F}_9$ reaches up to 48.6%, it becomes 26.8% for polynomial multiplication over $\mathbb{F}_3$ with the input size $n = 1280$. Results indicate that the new 4-way algorithm N3 has better acquisition in the arithmetic complexity than the other two 4-way algorithms N1, N2 [40], and Bernstein's 3-way algorithm B1 [5]. According to Table 6.1, Table 6.2, and Table 6.3, the application of the U1 algorithm yields the fastest polynomial multiplication results so far that are specific to characteristic three fields. As a use case of these new characteristic three polynomial multiplication algorithms, we pick the post-quantum key encapsulation mechanism NTRU Prime [5–9]. We report that with the proposed N1-Hybrid algorithm in this thesis [40], the arithmetic complexity for the polynomial multiplication over $\mathbb{F}_3$ is reduced by 38.35% compared to Bernstein's Hybrid-1 method which is described in [7]. As Table 7.2 indicates the reduction percentage regarding the implementation performance is 37.39% for N1-Hybrid2 compared to the Hybrid-1 algorithm. Therefore, N1-Hybrid2 could be a better alternative for the polynomial multiplication in the Streamlined NTRU Prime Key Encapsulation Mechanism described in [7]. As we

know, Bernstein *et al.* introduce a different hybrid approach in [5], which we call B1-Hybrid1 for the input size $n = 653$ and B1-Hybrid2 for the input size $n = 761$ for the characteristic three polynomial multiplication in the NTRU Prime decapsulation. We show that the proposed hybrid methods in this thesis [39,40] such as U1-Hybrid1 and U1-Hybrid2 surpass the implementation performances of Bernstein's B1-Hybrid1 and B1-Hybrid2 approaches [5] as well. As Table 7.3 indicates, the reduction percentage regarding the implementation performance is 29.85% for the U1-Hybrid1 algorithm whereas it is 35.52% for U1-Hybrid2. Therefore, U1-Hybrid1 and U1-Hybrid2 can be better alternatives than the B1-Hybrid1 and B1-Hybrid2 for the characteristic three polynomial multiplication in NTRU Prime decapsulation. It should also be noted that the implementation improvement for the multiplication step that comes with the proposed U1-Hybrid2 method is 8.56% faster than the previous improvement for the same multiplication step that comes with the proposed N1-Hybrid2 method for the input size 761. In the conclusion, implementing the new hybrid methods using Haswell x86 architecture with AVX/AVX2 instructions has the potential of improving the performance of the Streamlined NTRU Prime Key Encapsulation Protocol.

REFERENCES

[1] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, Springall, N. Heninger, D., E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann, Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice, Association for Computing Machinery, pp. 5–17, 2015.

[2] U. S. N. S. Agency, Commercial national security algorithm suite and quantum computing faq, 2016, https://cryptome.org/2016/01/CNSA-Suite-and-Quantum-Computing-FAQ.pdf.

[3] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tonel, Status report on the second round of the nist post-quantum cryptography standardization process, NIST, july 2020, https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf.

[4] D. Bernstein, Batch Binary Edwards, Advances in Cryptology - CRYPTO 2009, 5677, pp. 317–336, 2009.

[5] D. J. Bernstein, B. B. Brumley, M. Chen, and N. Tuveri, OpenSSLNTRU: Faster postquantum TLS key exchange, IACR Cryptol. ePrint Arch., 826, 2021, https://eprint.iacr.org/2021/826.pdf.

[6] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. V. Vredendaal, Ntru prime, NIST Post-Quantum Cryptography Standardization Process-Round-1, 2017, https://ntruprime.cr.yp.to/nist/ntruprime-20171130.pdf.

[7] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. V. Vredendaal, NTRU Prime: reducing attack surface at low cost, International Conference on Selected Areas in Cryptography, 24, pp. 235–260, august 2017.

[8] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. V. Vredendaal, Ntru prime, NIST Post-Quantum Cryptography Standardization Process-Round-3, 2019, https://ntruprime.cr.yp.to/nist/ntruprime-20201007.pdf.

[9] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. V. Vredendaal, Ntru prime, NIST Post-Quantum Cryptography Standardiza-

tion Process-Round-2, 2019, https://ntruprime.cr.yp.to/nist/ntruprime-20190330.pdf.

[10] F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé, and P. Zimmermann, 2019, https://lists.gforge.inria.fr/pipermail/cado-nfs-discuss/2019-December/001139.html.

[11] M. Cenk, Karatsuba-like formulae and their associated techniques, J. Cryptogr. Eng., 8(3), pp. 259–269, 2018.

[12] M. Cenk and M. A. Hasan, Some new results on binary polynomial multiplication, J. Cryptogr. Eng., 5(4), pp. 289–303, 2015.

[13] M. Cenk, c. K. Koç, and F. Özbudak, Polynomial multiplication over finite fields using field extensions and interpolation, IEEE symposium on computer arithmetic, pp. 84–91, 2009.

[14] M. Cenk, C. Negre, and M. A. Hasan, Improved three-way split formulas for binary polynomial multiplication, Selected areas in cryptography, pp. 384–398, 2011.

[15] M. Cenk, C. Negre, and M. A. Hasan, Improved 3-way split formulas for binary polynomial and toeplitz matrix vector products, IEEE Transactions on Computers, 62(7), pp. 1345–1361, 2013.

[16] M. Cenk and F. Özbudak, Efficient Multiplication in $F_{3^{lm}}$, $m \geq 1$ and $5 \leq l \leq 18$, AFRICACRYPT, pp. 406–414, 2008.

[17] M. Cenk, F. H. Zadeh, and M. A. Hasan, New Efficient Algorithms for Multiplication Over Fields of Characteristic Three, J. Sign. Process Syst., 90(3), pp. 285–294, march 2018.

[18] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Pres, U.S., 2nd edition, 2001.

[19] N. S. B. Cryptography, Suit b implementers' guide to nist sp 800-56a, 2009, https://web.archive.org/web/20160306033416/http://www.nsa.gov/ia/_files/SuiteB_Implementer_G-113808.pdf.

[20] W. Diffie and M. Hellman, New directions in cryptography, IEEE Transactions on Information Theory, 22(6), pp. 644–654, 1976.

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

[22] L. K. Grover, A Fast Quantum Mechanical Algorithm for Database Search, Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, pp. 212–219, 1996.

[23] M. B. Ilter and M. Cenk, Efficient Big Integer Multiplication in Cryptography, International Journal of Information Security Science, 6(4), pp. 70–78, december 2017.

[24] J. Martinis and S. Boixo, Quantum supremacy using a programmable superconducting processor, October 2019, https://ai.googleblog.com/2019/10/quantum-supremacy-using-programmable.html.

[25] R. C. Merkle, Secure Communications over Insecure Channels, Association for Computing Machinery, 21(4), pp. 294–299, 1978.

[26] P. L. Montgomery, Five, six and seven-term karatsuba-like formulae , IEEE Transactions on Computers, 54(3), pp. 362–369, 2005.

[27] NIST, Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography, 2006, https://csrc.nist.gov/publications/detail/sp/800-56a/revised/archive/2007-03-14.

[28] NIST, Post quantum cryptography pqc standardization, 2016-2020, https://csrc.nist.gov/projects/post-quantum-cryptography.

[29] M. Oneill and C. Rafferty, Evaluation of Large Integer Multiplication Methods on Hardware, IEEE Transactions on Computers, 66(8), pp. 1369–1382, august 2017.

[30] E. Pednault, J. Gunnels, D. Maslov, and J. Gambetta, On Quantum Supremacy, IBM Research Blog, 2019.

[31] C. Research, Standards for efficient cryptography, sec 1: Elliptic curve cryptography, version 2.0, 2009, https://www.secg.org/sec1-v2.pdf.

[32] R. L. Rivest, A. Shamir, and L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Association for Computing Machinery, 21(2), pp. 120–126, 1978.

[33] K. H. Rosen, *Elementary Number Theory and Its Application 6th ed*, Addison-Wesley, 2011.

[34] P. Shor, Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM Journal on Computing, 26(5), pp. 1484–1509, 1997.

[35] P. W. Shor, Refined analysis and improvements on some factoring algorithms, Journal of Algorithms, 3(2), pp. 101–127, 1982.

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

[37] A. Weimerskirch and C. Paar, Generalizations of the Karatsuba Algorithm for Efficient Implementations, IACR Cryptol. ePrint Arch., p. 224, 2006.

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

[39] E. Yeniaras and M. Cenk, Improved polynomial multiplication algorithms over characteristic three fields and applications to NTRU Prime, The 14th International Conference on Security for Information Technology and Communications - SECITC'21.

[40] E. Yeniaras and M. Cenk, Faster characteristic three polynomial multiplication and its application to NTRU Prime decapsulation, Journal of Cryptographic Engineering, 2022, https://doi.org/10.1007/s13389-021-00282-7.

[41] G. Zhou and H. Michalik, Comments on a new architecture for a parallel finite field multiplier with low complexity based on composite field, IEEE Trans. Computers, 59(7), pp. 1007–1008, 2010.

APPENDIX A

COST OF MULTI-EVALUATION AND RESCONSTRUCTION TABLES FOR THE A1, A3, B1, N1, N2, N3, AND V1 ALGORITHMS

Table A.1: Costs of multi-evaluation and reconstruction for A1 3-way algorithm $\mathbb{F}_3$ and $\mathbb{F}_9$ - unbalanced split version

Computations	Cost in $\mathbb{F}_3[x]$	Cost in $\mathbb{F}_9[x]$
$R_1 = A_0 + A_2$, $R_12 = B_0 + B_2$, $k \leq n$	$2k$	$4k$
$R_2 = R_1 + A_1$, $R'_2 = R'_1 + B_1$	$2n$	$4n$
$R_3 = R_1 - A_1$, $R_3 = R'_1 - B_1$	$2n$	$4n$
$R_4 = \xi A1$, $R'_4 = \xi B1$	0	0
$R_5 = A_0 - A_2$, $R'_5 = B_0 - B_2$, $k \leq n$	$2k$	$4k$
$R_6 = R_4 + R_5$, $R'_6 = R'_4 + R'_5$	0	$4n$
$P_0 = A_0 B_0$	$M_3(n)$	$M_9(n)$
$P_1 = R_2 R'_2$	$M_3(n)$	$M_9(n)$
$P_2 = R_3 R'_3$	$M_3(n)$	$M_9(n)$
$P_3 = R_6 R'_6$	$M_9(n)$	$M_9(n)$
$P_4 = A_2 B_2$	$M_3(k)$	$M_9(k)$
$U_1 = P_1 - P_2$	$(2n-1)$	$(4n-2)$
$U_2 = P_1 + P_2$	$(2n-1)$	$(4n-2)$
$U_3 = P_0 + P_4$, $k \leq n$	$(2k-1)$	$(4k-2)$
$C_2 = -U_2 - U_3$	$(2n-1)$	$(4n-2)$
$U_4 = \omega(U_2 - U_3)$	0	$(4n-2)$
$U_5 = U_4 - \omega P_3$	0	$(4n-2)$
$C_3 = U_1 + U_5$	$(2n-1)$	$(4n-2)$
$C_1 = U_1 - U_5$	$(2n-1)$	$(4n-2)$
$C = P_0 + C_1 x^n + C_2 x^{2n} + C_3 x^{3n} + C_4 x^{4n}$	$(4n-4)$	$(8n-8)$
TOTAL:	$M_3(2n+k) = 3M_3(n) + M_3(k) + M_9(n) + 18n + 6k - 10$	$M_9(2n+k) = 4M_9(n) + M_9(k) + 48n + 12k - 24$

Table A.2: Costs of multi-evaluation and reconstruction for A3 3-way algorithm $\mathbb{F}_3$ - unbalanced split version

Computations	Cost in $\mathbb{F}_3[x]$
$R_1 = A_0 - A_2$, $R'_1 = B_0 - B_2$, $k \leq n$	$2k$
$R_2 = A_0 + A_2$, $R'_2 = B_0 + B_2$, $k \leq n$	$2k$
$R_3 = R_2 + A_1$, $R_3 = R'_2 + B_1$	$2n$
$R_4 = R_1 + \xi A1$, $R'_4 = R'_1 + \xi B1$	0
$R_5 = R_1 - \xi A1$, $R'_5 = R'_1 xi B1$	0
$P_0 = A_0 B_0$	$M_3(n)$
$P_1 = R_3 R'_3$	$M_3(n)$
$P_2 = R_4 R'_4$	$M_9(n)$
$P_3 = R_5 R'_5$	0
$P_4 = A_2 B_2$	$M_3(k)$
$U_1 = P_0 + P_4$, $k \leq n$	$(2k-1)$
$U_2 = P_1 + P_{2,0}$	$(2n-1)$
$U_3 = -U_1 - U_2$	$(2n-1)$
$C_1 = U_3 - P_{2,1}$	$(2n-1)$
$C_2 = U_1 - P_{2,0}$	$(2n-1)$
$C_3 = U_3 + P_{2,1}$	$(2n-1)$
$C = C_0 + C_1 x^n + C_2 x^{2n} + C_3 x^{3n} + C_4 x^{4n}$	$(4n-4)$
TOTAL:	$M_3(2n+k) = 2M_3(n) + M_3(k) + M_9(n) + 16n + 6k - 10$

Table A.3: Costs of multi-evaluation and reconstruction for Bernstein's 3-way algorithm B1 in $\mathbb{F}_3$ and $\mathbb{F}_9$ - unbalanced split version

Computations	Cost in $\mathbb{F}_3[x]$	Cost in $\mathbb{F}_9[x]$
$R_0 = A_0 + A_2, R'_0 = B_0 + B_2$, $k \leq n$	$2k$	$4k$
$R_1 = R_0 + A_1, R'_1 = R'_0 + B_1$	$2n$	$4n$
$R_2 = R_0 - A_1, R'_2 = R'_0 - B_1$	$2n$	$4n$
$R_3 = A_1 x + A_2 x^2, R'_3 = B_1 x + B_2 x^2$, $k \leq (n-1)$	$2k$	$4k$
$R_4 = A_0 + R_3, R'_4 = B_0 + R'_3$	$(2n-2)$	$(4n-4)$
$P_0 = A_0 B_0$	$M_3(n)$	$M_9(n)$
$P_1 = R_1 R'_1$	$M_3(n)$	$M_9(n)$
$P_2 = R_2 R'_2$	$M_3(n)$	$M_9(n)$
$P_3 = R_4 R'_4$	$M_3(n+2) = M_3(n) + 8n + 4$	$M_9(n+2) = M_9(n) + 32n + 24$
$P_4 = A_2 B'_2$	$M_3(k)$	$M_9(k)$
$V_0 = P_1 + P_2$	$(2n-1)$	$(4n-2)$
$V_1 = P_1 - P_2$	$(2n-1)$	$(4n-2)$
$V_2 = V_0 x + V_1$	$(2n-2)$	$(4n-4)$
$V_3 = V_2 + P_3/x$	$2n$	$4n$
$V_4 = V_3/(x-1)$	$(2n-2)$	$(4n-4)$
$V = V_4/(x+1)$	$(2n-3)$	$(4n-6)$
$U_0 = V - P_4 x$, $k \leq (n-2)$	$(2k-1)$	$(4k-2)$
$U = U_0 + P_0/x$	$(2n-2)$	$(4n-4)$
$C_0 = U + V_1$	$(2n-2)$	$(4n-4)$
$C_1 = P_0 - C_0 x^n$	$(n-1)$	$(2n-2)$
$C_2 = P_0 + V_0$	$(2n-1)$	$(4n-2)$
$C_3 = C_2 + P_4$, $k \leq n$	$(2k-1)$	$(4k-2)$
$C_4 = C_1 - C_3 x^{2n}$	$(n-1)$	$(2n-2)$
$C_5 = C_4 + U x^{3n}$	$(n-1)$	$(2n-2)$
$C = C_5 + P_4 x^{4n}$, $k \geq n/2$	$(n-1)$	$(2n-2)$
TOTAL:	$M_3(2n+k) = 4M_3(n) + M_3(k) + 36n + 8k - 18$	$M_9(2n+k) = 4M_9(n) + M_9(k) + 88n + 16k - 20$

Table A.4: Costs of multi-evaluation and reconstruction for N1 in $\mathbb{F}_3[x]$ - unbalanced split version

Computations	Cost for multiplication in $\mathbb{F}_3[x]$
$R_0 = A_0 - A_2, R'_0 = B_0 - B_2$	$2n$
$R_1 = A_1 - A_3, R'_1 = B_1 - B_3$, $k \leq n$	$2k$
$R_2 = A_1 + A_3, R'_2 = B_1 + B_3$, $k \leq n$	$2k$
$R_3 = A_0 + R_2, R'_3 = B_0 + R'_2$	$2n$
$R_4 = R_1 - A_2, R'_4 = R'_1 - B_2$	$2n$
$R_5 = A_0 - R_2, R'_5 = B_0 - R'_2$	$2n$
$R_6 = -A_2 - R_1, R'_6 = -B_2 - R'_1$	$2n$
$R_7 = R_0 + \omega R_1, R'_7 = R'_0 + \omega R'_1$	0
$R_8 = R_3 + \omega R_4, R'_8 = R'_3 + \omega R'_4$	0
$R_9 = R_5 + \omega R_6, R'_9 = R'_5 + \omega R'_6$	0
$R_{10} = R_0 - \omega R_1, R'_{10} = R'_0 - \omega R'_1$	0
$R_{11} = R_3 - \omega R_4, R'_{11} = R'_3 - \omega R'_4$	0
$R_{12} = R_5 - \omega R_6, R'_{12} = R'_5 - \omega R'_6$	0
$P_0 = R_7 R'_7$	$M_9(n)$
$P_1 = R_{10} R'_{10}$	0
$P_2 = R_8 R'_8$	$M_9(n)$
$P_3 = R_{11} R'_{11}$	0
$P_4 = R_9 R'_9$	$M_9(n)$
$P_5 = R_{12} R'_{12}$	0
$P_6 = A_3 B_3$	$M_3(k)$
$U_0 = P_{2,1} + P_{4,1}$	$(2n-1)$
$U_1 = P_{4,1} - P_{2,1}$	$(2n-1)$
$U_2 = P_{2,0} + P_{4,0}$	$(2n-1)$
$U_3 = P_{2,0} - P_{4,0}$	$(2n-1)$
$U_4 = P_6 - P_{0,0}$, $k \leq n$	$(2k-1)$
$U_5 = U_4 - U_0$	$(2n-1)$
$C_0 = U_5 + U_2$	$(2n-1)$
$C_1 = U_3 - P_{0,1}$	$(2n-1)$
$C_2 = U_0 + P_6$, $k \leq n$	$(2k-1)$
$C_3 = U_1 + U_3$	$(2n-1)$
$C_4 = U_5 - U_2$	$(2n-1)$
$C_5 = U_1 - P_{0,1}$	$(2n-1)$
$C = C_0 + C_1 x^n \quad + C_2 x^{2n} + ... + C_6 x^{6n}$	$6(n-1)$
TOTAL:	$M_3(3n+k) = M_3(k) + 3M_9(n) + 36n + 8k - 18$

Table A.5: Costs of multi-evaluation and reconstruction for N1 in $\mathbb{F}_9[x]$ - unbalanced split version

Computations	Cost for multiplication in $\mathbb{F}_9[x]$
$R_0 = A_0 - A_2$, $R'_0 = B_0 - B_2$	$4n$
$R_1 = A_1 - A_3$, $R'_1 = B_1 - B_3$, $k \leq n$	$4k$
$R_2 = A_1 + A_3$, $R'_2 = B_1 + B_3$, $k \leq n$	$4k$
$R_3 = A_0 + R_2$, $R'_3 = B_0 + R'_2$	$4n$
$R_4 = R_1 - A_2$, $R'_4 = R'_1 - B_2$	$4n$
$R_5 = A_0 - R_2$, $R'_5 = B_0 - R'_2$	$4n$
$R_6 = -A_2 - R_1$, $R'_6 = -B_2 - R'_1$	$4n$
$R_7 = R_0 + \omega R_1$, $R'_7 = R'_0 + \omega R'_1$	$4n$
$R_8 = R_3 + \omega R_4$, $R'_8 = R'_3 + \omega R'_4$	$4n$
$R_9 = R_5 + \omega R_6$, $R'_9 = R'_5 + \omega R'_6$	$4n$
$R_{10} = R_0 - \omega R_1$, $R'_{10} = R'_0 - \omega R'_1$	$4n$
$R_{11} = R_3 - \omega R_4$, $R'_{11} = R'_3 - \omega R'_4$	$4n$
$R_{12} = R_5 - \omega R_6$, $R'_{12} = R'_5 - \omega R'_6$	$4n$
$P_0 = R_7 R'_7$	$M_9(n)$
$P_1 = R_{10} R'_{10}$	$M_9(n)$
$P_2 = R_8 R'_8$	$M_9(n)$
$P_3 = R_{11} R'_{11}$	$M_9(n)$
$P_4 = R_9 R'_9$	$M_9(n)$
$P_5 = R_{12} R'_{12}$	$M_9(n)$
$P_6 = A_3 B_3$	$M_9(k)$
$U_1 = P_0 + P_1$	$2(2n-1)$
$U_2 = -P_0 + P_1$	$2(2n-1)$
$U_3 = P_2 + P_3$	$2(2n-1)$
$U_4 = -P_2 + P_3$	$2(2n-1)$
$U_5 = P_4 + P_5$	$2(2n-1)$
$U_6 = -P_4 + P_5$	$2(2n-1)$
$U_7 = U_3 + U_5$	$2(2n-1)$
$U_8 = -U_3 + U_5$	$2(2n-1)$
$U_9 = U_4 + U_6$	$2(2n-1)$
$U_{10} = U_4 - U_6$	$2(2n-1)$
$U_{11} = U_1 - U_7$	$2(2n-1)$
$U_{12} = U_{11} + P_6$, $k \leq n$	$2(2k-1)$
$U_{13} = U_1 + U_7$	$2(2n-1)$
$U_{14} = U_{13} + P_6$, $k \leq n$	$2(2k-1)$
$C_0 = U_{12} + \omega U_9$	$2(2n-1)$
$C_1 = U_8 + \omega U_2$	$2(2n-1)$
$C_2 = P_6 - \omega U_9$, $k \leq n$	$2(2k-1)$
$C_3 = U_8 + \omega U_{10}$	$2(2n-1)$
$C_4 = U_{14} + \omega U_9$	$2(2n-1)$
$C_5 = \omega(U_2 + U_{10})$	$2(2n-1)$
$C = C_0 + C_1 x^n \quad + C_2 x^{2n} + ... + C_6 x^{6n}$	$12(n-1)$
TOTAL:	$M_9(3n+k) = 6M_9(n) + M_9(k) + 124n + 20k - 52$

Table A.6: Costs of multi-evaluation and reconstruction for N2 in $\mathbb{F}_3$ - unbalanced split version

Computations	Cost for Multiplication in $\mathbb{F}_3$
$R_1 = A_1 + A_3$, $R'_1 = B_1 + B_3$, $k \leq n$	$2k$
$R_2 = R_1 + A_0$, $R'_2 = R'_1 + B_0$	$2n$
$R_3 = R_2 + A_2$, $R'_3 = R'_2 + B_2$	$2n$
$R_4 = A_0 - R_1$, $R'_4 = B_0 - R'_1$	$2n$
$R_5 = A_1 - A_2$, $R'_5 = B_1 - B_2$	$2n$
$R_6 = R_5 - A_3$, $R'_6 = R'_5 - B_3$, $k \leq n$	$2k$
$R_7 = -A_1 - A_2$, $R'_7 = -B_1 - B_2$	$2n$
$R_8 = R_7 + A_3$, $R'_8 = R'_7 + B_3$, $k \leq n$	$2k$
$R_9 = R_2 + \omega R_6$, $R'_9 = R'_2 + \omega R'_6$	0
$R_{10} = R_4 + \omega R_8$, $R'_{10} = R'_4 + \omega R'_8$	0
$R_{11} = R_2 - \omega R_6$, $R'_{11} = R'_2 - \omega R'_6$	0
$R_{12} = R_4 - \omega R_8$, $R'_{12} = R'_4 - \omega R'_8$	0
$P_0 = A_0 B_0$	$M_3(n)$
$P_1 = R_3 R'_3$	$M_3(n)$
$P_2 = R_9 R'_9$	$M_9(n)$
$P_3 = R_{11} R'_{11}$	0
$P_4 = R_{10} R'_{10}$	$M_9(n)$
$P_5 = R_{12} R'_{12}$	0
$P_6 = A_3 B_3$	$M_3(k)$
$U_1 = P_0 + P_1$	$(2n-1)$
$U_2 = P_{2,0} + P_{4,0}$	$(2n-1)$
$U_3 = P_{2,1} + P_{4,1}$	$(2n-1)$
$U_4 = P_{2,0} - P_{4,0}$	$(2n-1)$
$U_5 = P_{2,1} - P_{4,1}$	$(2n-1)$
$U_6 = U_2 - U_1$	$(2n-1)$
$U_7 = U_6 - P_6$, $k \leq n$	$(2k-1)$
$U_8 = U_3 - U_1$	$(2n-1)$
$U_9 = U_8 - P_{4,0}$	$(2n-1)$
$C_1 = U_7 - P_{2,1}$	$(2n-1)$
$C_2 = U_3 + P_6$, $k \leq n$	$(2k-1)$
$C_3 = U_4 - U_5$	$(2n-1)$
$C_4 = U_2 + P_0$	$(2n-1)$
$C_5 = U_9 - P_6$, $k \leq n$	$(2k-1)$
$C = C_0 + C_1 x^n + C_2 x^{2n} + ... + C_6 x^{6n}$	$6(n-1)$
TOTAL:	$M_3(3n+k) = 2M_3(n) + M_3(k) + 2M_9(n) + 38n + 12k - 20$

Table A.7: Costs of multi-evaluation and reconstruction for N2 in $\mathbb{F}_9$ - unbalanced split version

Computations	Cost for Multiplication in $\mathbb{F}_9$
$R_1 = A_1 + A_3$, $R'_1 = B_1 + B_3$, $k \leq n$	$4k$
$R_2 = R_1 + A_0$, $R'_2 = R'_1 + B_0$	$4n$
$R_3 = R_2 + A_2$, $R'_3 = R'_2 + B_2$	$4n$
$R_4 = A_0 - R_1$, $R'_4 = B_0 - R'_1$	$4n$
$R_5 = A_1 - A_2$, $R'_5 = B_1 - B_2$	$4n$
$R_6 = R_5 - A_3$, $R'_6 = R'_5 - B_3$, $k \leq n$	$4k$
$R_7 = -A_1 - A_2$, $R'_7 = -B_1 - B_2$	$4n$
$R_8 = R_7 + A_3$, $R'_8 = R'_7 + B_3$, $k \leq n$	$4k$
$R_9 = R_2 + \omega R_6$, $R'_9 = R'_2 + \omega R'_6$	$4n$
$R_{10} = R_4 + \omega R_8$, $R'_{10} = R'_4 + \omega R'_8$	$4n$
$R_{11} = R_2 - \omega R_6$, $R'_{11} = R'_2 - \omega R'_6$	$4n$
$R_{12} = R_4 - \omega R_8$, $R'_{12} = R'_4 - \omega R'_8$	$4n$
$P_0 = A_0 B_0$	$M_9(n)$
$P_1 = R_3 R'_3$	$M_9(n)$
$P_2 = R_9 R'_9$	$M_9(n)$
$P_3 = R_{11} R'_{11}$	$M_9(n)$
$P_4 = R_{10} R'_{10}$	$M_9(n)$
$P_5 = R_{12} R'_{12}$	$M_9(n)$
$P_6 = A_3 B_3$	$M_9(k)$
$U_1 = -P_0 - P_1$	$2(2n-1)$
$U_2 = P_4 + P_5$	$2(2n-1)$
$U_3 = P_2 + P_3$	$2(2n-1)$
$U_4 = P_2 - P_3$	$2(2n-1)$
$U_5 = P_4 - P_5$	$2(2n-1)$
$U_6 = -U_3 - U_2$	$2(2n-1)$
$U_7 = U_1 + U_6$	$2(2n-1)$
$U_8 = U_7 - P_6$, $k \leq n$	$2(2k-1)$
$U_9 = -U_4 + U_5$	$2(2n-1)$
$U_{10} = -U_3 + U_2$	$2(2n-1)$
$U_{11} = U_4 + U_5$	$2(2n-1)$
$U_{12} = U_1 + U_2$	$2(2n-1)$
$U_{13} = U_{12} - P_6$, $k \leq n$	$2(2k-1)$
$C_1 = U_8 - \omega U_4$	$2(2n-1)$
$C_2 = P_6 + \omega U_{11}$, $k \leq n$	$2(2k-1)$
$C_3 = U_{10} + \omega U_9$	$2(2n-1)$
$C_4 = P_0 + U_6$	$2(2n-1)$
$C_5 = U_{13} + U_{11}$	$2(2n-1)$
$C = C_0 + C_1 x^n + C_2 x^{2n} + ... + C_6 x^{6n}$	$12(n-1)$
TOTAL:	$M_9(3n+k) = 6M_9(n) + M_9(k) + 108n + 24k - 48$

Table A.8: Costs of multi-evaluation and reconstruction for N3 in $\mathbb{F}_3$ and $\mathbb{F}_9$ - unbalanced split version

Computations	Cost in $\mathbb{F}_3[x]$	Cost in $\mathbb{F}_9[x]$
$R_0 = A_0 + A_2, R'_0 = B_0 + B_2$	$2n$	$4n$
$R_1 = A_1 + A_3, R'_1 = B_1 + B_3, k \leq n$	$2k$	$4k$
$R_2 = R_0 + R_1, R'_2 = R'_0 + R'_1$	$2n$	$4n$
$R_3 = R_0 - R_1, R'_3 = R'_0 - R'_1$	$2n$	$4n$
$R_4 = A_0 + A_1 x, R'_4 = B_0 + B_1 x$	$(2n-2)$	$(4n-4)$
$R_5 = R_4 + A_2 x^2, R'_5 = R'_4 + B_2 x^2$	$(2n-2)$	$(4n-4)$
$R_6 = R_5 + A_3 x^3, R'_6 = R'_5 + B'_3 x^3, k \leq (n-1)$	$2k$	$4k$
$R_7 = A_0 - A_2, R'_7 = B_0 - B_2$	$2n$	$4n$
$R_8 = A_1 - A_3, R'_8 = B_1 - B_3$, $k \leq n$	$2k$	$4k$
$R_9 = R_7 + \omega R_8, R'_9 = R'_7 + \omega R'_8$	0	$4n$
$R_{10} = R_7 - \omega R_8, R'_9 = R'_7 - \omega R'_8$	0	$4n$
$P_0 = A_0 B_0$	$M_3(n)$	$M_9(n)$
$P_1 = R_2 R'_2$	$M_3(n)$	$M_9(n)$
$P_2 = R_3 R'_3$	$M_3(n)$	$M_9(n)$
$P_3 = R_6 R'_6$	$M_3(n+3) = M_3(n) + 12n + 12$	$M_9(n+3) = M_9(n) + 48n + 60$
$P_4 = R_9 R'_9$	$M_9(n)$	$M_9(n)$
$P_5 = R_{10} R'_{10}$	0	$M_9(n)$
$P_6 = A_3 B_3$	$M_3(k)$	$M_9(k)$
$U_0 = P_0/x$	0	0
$U_1 = P_3/x$	0	0
$U_2 = U_1/(x^2-1)$	$(2n-2)$	$(4n-4)$
$U_3 = U_2/(x^2+1)$	$(2n-4)$	$(4n-8)$
$U_4 = P_1 + P_2$	$(2n-1)$	$(4n-2)$
$U_5 = (P_1 + P_2)x$	0	0
$U_6 = U_5/(x^2-1)$	$(2n-4)$	$(4n-8)$
$U_7 = -P_{5,0} = P_4 + P_5$	0	$(4n-2)$
$U_8 = -xP_{5,0}$	0	0
$U_9 = U_8/(x^2+1)$	$(2n-4)$	$(4n-8)$
$U_{10} = (U_6 + U_9)$	$(2n-2)$	$(4n-4)$
$U_{11} = U_0 + U_3$	$(2n-2)$	$(4n-4)$
$U_{12} = U_{11} - U_{10}$	$(2n-2)$	$(4n-4)$
$U = U_{12} - xP_6$, $k \leq n$	$(2k-1)$	$(4k-2)$
$V_0 = P_1 - P_2$	$(2n-1)$	$(4n-2)$
$V_1 = (P_1 - P_2)x^2$	0	0
$V_2 = V_1/(x^2-1)$	$(2n-3)$	$(4n-6)$
$V_3 = -P_{5,1} = \omega(P_4 - P_5)$	0	$(4n-2)$
$V_4 = -x^2 P_{5,1}$	0	0
$V_5 = V_4/(x^2+1)$	$(2n-3)$	$(4n-6)$
$V_6 = V_2 - V_5$	$(2n-1)$	$(4n-2)$
$V_7 = V_6 - U$	$(2n-1)$	$(4n-2)$
$V_8 = U_4 - U_7$	$(2n-1)$	$(4n-2)$
$V_9 = V_8 - P_6$, $k \leq n$	$(2k-1)$	$(4k-2)$
$V_{10} = V_0 + V_3$	$(2n-1)$	$(4n-2)$
$V_{11} = -P_0 + U_4$	$(2n-1)$	$(4n-2)$
$V_{12} = V_{11} + U_7$	$(2n-1)$	$(4n-2)$
$V_{13} = -(V_2/x^2 + V_5/x^2)$	$(2n-3)$	$(4n-6)$
$V_{14} = V_{13} + U$	$(2n-3)$	$(4n-6)$
$C_0 = P_0 + x^n V_7$	$(n-1)$	$(2n-2)$
$C_1 = C_0 + x^{2n} V_9$	n	$2n$
$C_2 = C_1 + x^{3n} V_{10}$	$(n-1)$	$(2n-2)$
$C_3 = C_2 + x^{4n} V_{12}$	$(n-1)$	$(2n-2)$
$C_4 = C_3 + x^{5n} V_{14}$	$(n-1)$	$(2n-2)$
$C_= C_4 + x^{6n} P_6$, $k \geq (n+1)/2$	n	$2n$
TOTAL:	$M_3(3n+k) = 4M_3(n) + M_3(k) +$ $M_9(n) + 68n + 10k - 38$	$M_9(3n+k) = 6M_9(n) + M_9(k)$ $+ 176n + 20k - 44$

Table A.9: Costs of multi-evaluation and reconstruction for V1 in $\mathbb{F}_3$ - unblanced split version

Computations	Cost for Multiplication in $\mathbb{F}_3$
$R_1 = A_1 + A_3, R'_1 = B_1 + B_3$	$2n$
$R_2 = A_0 - A_4, R'_2 = B_0 - B_4$, $k \leq n$	$2k$
$R_3 = A_0 + A_4, R'_3 = B_0 + B_4$, $k \leq n$	$2k$
$R_4 = A_1 - A_3, R'_4 = B_1 - B_3$	$2n$
$R_5 = R_4 - A_2, R'_5 = R'_4 - B_2$	$2n$
$R_6 = -A_2 - R_4, R'_6 = -B_2 - R'_4$	$2n$
$R_7 = R_3 - A_2, R'_7 = R'_3 - B_2$	$2n$
$R_8 = R_1 + R_2, R'_8 = R'_1 + R'_2$	$2n$
$R_9 = R_2 - R_1, R'_8 = R'_2 - R'_1$	$2n$
$R_{10} = R_1 + R_3, R'_{10} = R'_1 + R'_3$	$2n$
$R_{11} = R_{10} + A_2, R'_{11} = R'_{10} + B_2$	$2n$
$R_{12} = R_7 + \omega R_4, R'_{12} = R'_7 + \omega R'_4$	0
$R_{13} = R_7 - \omega R_4, R'_{13} = R'_7 - \omega R'_4$	0
$R_{14} = R_8 + \omega R_5, R'_{14} = R'_8 + \omega R'_5$	0
$R_{15} = R_8 - \omega R_5, R'_{15} = R'_8 - \omega R'_5$	0
$R_{16} = R_9 + \omega R_6, R'_{16} = R'_9 + \omega R'_6$	0
$R_{17} = R_9 - \omega R_6, R'_{17} = R'_9 - \omega R'_6$	0
$P_0 = A_0 B_0$	$M_3(n)$
$P_1 = R_{11} R'_{11}$	$M_3(n)$
$P_2 = R_{12} R'_{12}$	$M_9(n)$
$P_3 = R_{13} R'_{13}$	0
$P_4 = R_{14} R'_{14}$	$M_9(n)$
$P_5 = R_{15} R'_{15}$	0
$P_6 = R_{16} R'_{16}$	$M_9(n)$
$P_7 = R_{17} R'_{17}$	0
$P_8 = A_4 B_4$	$M_3(k)$
$U_1 = P_0 + P_8$, $k \leq n$	$(2k-1)$
$U_2 = -U_1 + P_1$	$(2n-1)$
$U_3 = P_{2,0} - P_{6,0}$	$(2n-1)$
$U_4 = U_3 - P_{4,0}$	$(2n-1)$
$U_5 = P_{2,0} - P_{4,0}$	$(2n-1)$
$U_6 = P_{6,0} + P_{4,0}$	$(2n-1)$
$U_7 = P_{2,1} - P_{4,1}$	$(2n-1)$
$U_8 = U_7 + P_{6,1}$	$(2n-1)$
$U_9 = P_{4,1} + P_{6,1}$	$(2n-1)$
$U_{10} = P_{2,1} + P_{4,1}$	$(2n-1)$
$U_{11} = U_{10} - P_{6,1}$	$(2n-1)$
$U_{12} = U_2 + U_3$	$(2n-1)$
$U_{13} = U_1 + U_4$	$(2n-1)$
$U_{14} = U_2 + U_5$	$(2n-1)$
$C_1 = U_{12} + U_8$	$(2n-1)$
$C_2 = U_{13} - U_9$	$(2n-1)$
$C_3 = U_{12} - U_8$	$(2n-1)$
$C_4 = U_1 + U_6$	$(2n-1)$
$C_5 = U_{14} + U_{11}$	$(2n-1)$
$C_6 = U_{13} + U_9$	$(2n-1)$
$C_7 = U_{14} - U_{11}$	$(2n-1)$
$C = C_0 + C_1 x^n + ... + C_8 x^{8n}$	$8(n-1)$
TOTAL:	$M_3(4n+k) = 2M_3(n) + M_3(k) + 3M_9(n) + 66n + 6k - 29$

Table A.10: Costs of multi-evaluation and reconstruction for V1 in $\mathbb{F}_9$ - unbalanced split version

Computations	Cost for Multiplication in $\mathbb{F}_9$
$R_1 = A_1 + A_3, R'_1 = B_1 + B_3$	$4n$
$R_2 = A_0 - A_4, R'_2 = B_0 - B_4$, $k \leq n$	$4k$
$R_3 = A_0 + A_4, R'_3 = B_0 + B_4$, $k \leq n$	$4k$
$R_4 = A_1 - A_3, R'_4 = B_1 - B_3$	$4n$
$R_5 = R_4 - A_2, R'_5 = R'_4 - B_2$	$4n$
$R_6 = -A_2 - R_4, R'_6 = -B_2 - R'_4$	$4n$
$R_7 = R_3 - A_2, R'_7 = R'_3 - B_2$	$4n$
$R_8 = R_1 + R_2, R'_8 = R'_1 + R'_2$	$4n$
$R_9 = R_2 - R_1, R'_9 = R'_2 - R'_1$	$4n$
$R_{10} = R_1 + R_3, R'_{10} = R'_1 + R'_3$	$4n$
$R_{11} = R_{10} + A_2, R'_{11} = R'_{10} + B_2$	$4n$
$R_{12} = R_7 + \omega R_4, R'_{12} = R'_7 + \omega R'_4$	$4n$
$R_{13} = R_7 - \omega R_4, R'_{13} = R'_7 - \omega R'_4$	$4n$
$R_{14} = R_8 + \omega R_5, R'_{14} = R'_8 + \omega R'_5$	$4n$
$R_{15} = R_8 - \omega R_5, R'_{15} = R'_8 - \omega R'_5$	$4n$
$R_{16} = R_9 + \omega R_6, R'_{16} = R'_9 + \omega R'_6$	$4n$
$R_{17} = R_9 - \omega R_6, R'_{17} = R'_9 - \omega R'_6$	$4n$
$P_0 = A_0 B_0$	$M_9(n)$
$P_1 = R_{11} R'_{11}$	$M_9(n)$
$P_2 = R_{12} R'_{12}$	$M_9(n)$
$P_3 = R_{13} R'_{13}$	$M_9(n)$
$P_4 = R_{14} R'_{14}$	$M_9(n)$
$P_5 = R_{15} R'_{15}$	$M_9(n)$
$P_6 = R_{16} R'_{16}$	$M_9(n)$
$P_7 = R_{17} R'_{17}$	$M_9(n)$
$P_8 = A_4 B_4$	$M_9(k)$
$U_1 = -P_0 + P_1$	$2(2n-1)$
$U_2 = P_2 + P_3$	$2(2n-1)$
$U_3 = P_2 - P_3$	$2(2n-1)$
$U_4 = P_6 + P_7$	$2(2n-1)$
$U_5 = P_6 - P_7$	$2(2n-1)$
$U_6 = P_4 + P_5$	$2(2n-1)$
$U_7 = P_4 - P_5$	$2(2n-1)$
$U_8 = U_4 + U_6$	$2(2n-1)$
$U_9 = U_1 - U_2$	$2(2n-1)$
$U_{10} = U_9 + U_4$	$2(2n-1)$
$U_{11} = U_{10} - P_8$, $k \leq n$	$2(2k-1)$
$U_{12} = -U_7 + U_5$	$2(2n-1)$
$U_{13} = U_3 + U_{12}$	$2(2n-1)$
$U_{14} = P_0 - U_2$	$2(2n-1)$
$U_{15} = U_{14} + U_8$	$2(2n-1)$
$U_{16} = U_{15} + P_8$, $k \leq n$	$2(2k-1)$
$U_{17} = -U_7 - U_5$	$2(2n-1)$
$U_{18} = U_9 + U_6$	$2(2n-1)$
$U_{19} = U_{18} - P_8$, $k \leq n$	$2(2k-1)$
$U_{20} = U_3 - U_{12}$, $k \leq n$	$2(2k-1)$
$U_{21} = P_0 - U_8$	$2(2n-1)$
$C_1 = U_{11} + \omega U_{13}$	$2(2n-1)$
$C_2 = U_{16} + \omega U_{17}$	$2(2n-1)$
$C_3 = U_{11} - \omega U_{13}$	$2(2n-1)$
$C_4 = U_{21} + P_8$, $k \leq n$	$2(2k-1)$
$C_5 = U_{19} + \omega U_{20}$	$2(2n-1)$
$C_6 = U_{16} - \omega U_{17}$	$2(2n-1)$
$C_7 = U_{19} - \omega U_{20}$	$2(2n-1)$
$C = C_0 + C_1 x^n + ... + C_8 x^{8n}$	$16(n-1)$
TOTAL:	$M_9(4n+k) = 8M_9(n) + M_9(k) + 168n + 28k - 72$

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Yeniaras, Esra
Nationality: Turkish (TC)
Date and Place of Birth: 16.03.1979, İstanbul
Marital Status: Single
e-mail: esramath@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
M.S.	University of Wisconsin-Madison	2010
M.S.	Ankara University	2007
M.Ed.	Gazi University	2006
B.S.	Ankara University	2004

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2011-present	Ministry of Education (MEB)	IT/Technician
2010-2011	University of California-Riverside	Teaching Assistant
2008-2010	University of Wisconsin-Madison	Teaching Assistant

PUBLICATIONS

International Journal Publications

- E. Yeniaras, M. Cenk. Faster characteristic three polynomial multiplication and its application to NTRU Prime decapsulation. J Cryptogr Eng (2022). https://doi.org/10.1007/s13389-021-00282-7

International Conference Publications

- E. Yeniaras, M. Cenk. Improved polynomial multiplication algorithms over characteristic three fields and applications to NTRU Prime, SECITC'21 - 14th International Conference on Security for Information Technology and Communications, (2021).