# New Efficient Characteristic Three Polynomial Multiplication Algorithms and Their Applications to NTRU Prime

Dr. Esra YENIARAS
Middle East Technical University & Ministry of Education
Turkiye

NIST Crypto Reading Club Presentation

September 21, 2022

# Outline

1. Problem Definition
   - Quantum Computers and Post Quantum Cryptography
   - NTRU Prime Protocol
   - NTRU Prime Decapsulation & Char 3 Polynomial Multiplication
   - Importance of Polynomial Multiplication in Cryptography
   - Our Purpose
2. Well-Known Characteristic 3 Polynomial Multiplication Algorithms & Recent Improvements
3. Our Contribution: New Efficient 4-way and 5-way Char 3 Polynomial Multiplication Algorithms
   - New 4-way Multiplication Algorithm (N1)
   - An Improved 4-way Multiplication Algorithm (N2)
   - Another Improved 4-way Multiplication Algorithm (N3)
   - New 5-way Multiplication Algorithm (V1)
   - Unbalanced 5-way Multiplication Algorithm (U1)
4. Application to NTRU Prime Decapsulation and Implementation Results
   - Bernstein's Hybrid-1 Approach vs Our Hybrid Methods
   - Bernstein's Improved B1-Hybrid Approach vs Our Improved Hybrid Methods
5. Conclusion
6. References

## QUANTUM COMPUTERS

- **Shor's Quantum Factoring Algorithm:** IFP (RSA), DLP (DH, ECDH) are vulnerable to attacks by sufficiently strong quantum computers. Thus, we need quantum-resistant algorithms!

- **Grover's Search Algorithm:** Reduces the search space from $O(N) \rightarrow O(\sqrt[2]{N})$ for brute force attacks!

  AES - 128$\rightarrow 2^{64}$ not secure enough! $X$

  AES - 256$\rightarrow 2^{128}$ $\checkmark$

# NIST's Post-Quantum Cryptography Competition

**NIST started a PQC Standardization Process in 2016 and different types of quantum-resistant algorithms are submitted as follows:**

- **Lattice-Based:** Saber, CRYSTALS-Kyber, CRYSTALS-Dilithium, New Hope, Frodo KEM, NTRU, NTRU Prime.
- **Code-Based:** BIKE, Classic McEllice, HQC
- **Supersingular Isogeny-Based:** SIKE.
- **Hash-Based:** Picnic, SPHINCS+.
- **Multivariate-Based:** GeMSS, Rainbow.

https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions

# NIST 3rd Round Results:

- **Third Round Finalists:**

  **Public Key Encryption/KEMs:** Classic McEliece, CRYSTALS-Kyber, NTRU, Saber.

  **Digital Signatures:** CRYSTALS-Dilithium, FALCON, Rainbow

- **Third Round Alternate Candidates:**

  **Public Key Encryption/KEMs**: BIKE, FrodoKEM, HQC, NTRU Prime, SIKE

  **Digital Signatures:** GeMSS, Picnic, SPHINCS+

# NIST Algorithms to be Standardized and the 4th Round:

- **Algorithms to be Standardized:**

  **Public Key Encryption/KEMs:** CRYSTALS-Kyber

  **Digital Signatures:** CRYSTALS-Dilithium, FALCON, SPHINCS+

- **Candidates Advancing to the Fourth Round:**

  **Public Key Encryption/KEMs:** BIKE, Classic McEliece, HQC, SIKE

  **Digital Signatures:**

- **LATTICE-BASED HARD PROBLEMS:**

  1- Shortest Vector Problem **(SVP)**

  2- Closest Vector Problem **(CVP)**

  3- The Shortest Independent Vector Problem **(SIVP)**

  4- Learning with errors **(LWE)**

  5- Ring Learning with Errors **(R-LWE)**

  6- Module Learning with Errors **(M-LWE)**

# NTRU Prime KEM: A NIST PQC Candidate

**NTRU PRIME: A LATTICE-BASED PQC ALGORITHM**

- **NTRU Prime KEM:** A **Lattice-based** KEM by Bernstein *et al.*
- Advanced to Round 3 as alternative candidate.
- **NTRU Prime** Based on hard problem **(SVP)**, to solve for input size $n \geq 100 \rightarrow$ Quantum Secure✓

# Why NTRU Prime: Char 3 Polynomial Multiplication

- **Google-Couldfire Experiment:** NTRU Prime KEM with batch key generation feature is considered as a faster and a more secure alternative to `ntruhrss701` for **TLS 1.3** in 2021 [6].
- NTRU Prime uses **characteristic-3 polynomial multiplication** in its decapsulation phase. (**Possible Improvement Here!**)

# NTRU Prime Decapsulation

### NTRU PRIME DECAPSULATION & CHAR 3 POLYNOMIAL MULTIPLICATION

- Decapsulation includes poly. mult. in $\mathbb{Z}_3[x]/(x^p - x - 1)$ where $p$ is prime [1].

**Algorithm 1** 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**

- $p$ and $q$ are prime numbers, $q \geq 17$, $0 < \omega \leq p$, $2p \geq 3\omega$, $q \geq 16\omega + 1$ and $x^p - x - 1$ is an irreducible polynomial in the polynomial ring $\mathbb{Z}_q[x]$.
- $\mathcal{R} = \mathbb{Z}[x]/(x^p - x - 1)$ ring
  $\mathcal{R}/3 = \mathbb{Z}_3[x]/(x^p - x - 1)$ ring
  $\mathcal{R}/q = \mathbb{Z}_q[x]/(x^p - x - 1)$ field
- If the parameters are $p = 761$, $q = 4591$ and $\omega = 286$ then the cryptosystem is represented as **sntrup761**.

# Another NIST PQC Candidate: NTRU KEM

**MORE PQC ALGORITHMS & CHAR 3 MULTIPLICATION**

- **NTRU KEM:**

  - Hüsling *et al.*, Advanced to Round 3 as a main candidate.

  - Merger of two earlier submissions **NTRU HRSS-KEM** and **NTRUEncrypt**.

  - Polynomial multiplication in $\mathbb{Z}_3[x]/(x^{n-1} + x^{n-2} + ... + 1)$ may improve the efficiency of the decapsulation phase.

- **Thus**, it is worth to improve the arithmetical completixy of the polynomial multiplication over $\mathbb{F}_3$.

# Importance of Polynomial Multiplication in Cryptography

**POLYNOMIAL MULTIPLICATION&CHARACTERISTIC 3 FIELDS:**

- **Polynomial multiplication** is a very commonly used, important tool in most cryptographic protocols that effects the efficiency.
- Polynomial multiplication in **char 3 fields** is used in many cryptographic applications such as **pairing-based cryptography** and/or **post-quantum cryptography: NTRU Prime, NTRU exc.**
- There exist efficient 2-way, 3-way, 4-way, 5-way (or above) split type poly. mult. algorithms in **binary fields**. However, in char 3, we have up to 3-way split algorithms so far. This can be improved!

**OUR PURPOSE IN THIS STUDY:**

- **Primary Purpose:** To develop **new** & **more efficient** polynomial multiplication algorithms that are specific to characteristic 3 fields in general.

  **And improve arithmetical complexities for multiplying polynomials in char 3.** ✓

- **Secondary Purpose:** To apply these new char 3 algorithms on the **NTRU Prime Decapsulation!**

  **And improve the implementation run-time of NTRU Prime Decapsulation**✓

# Well-Known Characteristic 3 Polynomial Multiplication Algorithms

- ✓ **SB**: Schoolbook polynomial multiplication algorithm.
- ✓ **LT**: Schoolbook recursion method [8]. We refer to it as the last term method.
- ✓ **KA2**: Improved (Refined) Karatsuba 2-way polynomial multiplication algorithm [8].
- ✓ **UB**: Unbalanced Refined Karatsuba 2-way polynomial multiplication algorithm [8].
- ✓ **KA3**: (Improved) Karatsuba like 3-way polynomial multiplication algorithm [10].

# Well-Known Characteristic 3 Polynomial Multiplication Algorithms

**RECENT IMPROVEMENTS:**

- ✓ **A3**: **In 2018**, Cenk, Hasan, and Zadeh [7] introduced a 3-way split polynomial multiplication algorithm using interpolation method which is similar to Toom-Cook's formula [7].

  → **A3** is more efficient than **SB**, **Refined Karatsuba 2-way**, **(Improved) Karatsuba like 3-way** algorithms.

- ✓ **B1**: **In 2021**, Bernstein *et al.* proposed a 3-way algorithm in [6, 8].

  → **B1** is more efficient than **A3** over $\mathbb{F}_3$ but slower than it over $\mathbb{F}_9$.

# Our Contributions: New Efficient Multiplication Algorithms in Char 3

- We develop new efficient 4-way split algorithms **N1, N2, and N3**.
- Furthermore, we develop new efficient 5-way split algorithm **V1** and the unbalanced 5-way version **U1**.
- We reduce the arithmetical complexities for Char 3 polynomial multiplication **in general**, by the help of the proposed N1, N2, N3, V1, and U1 algorithms.
- Finally, we apply the **hybrid use of N1, N2, N3, V1 and U1** combined with the others on **NTRU Prime Decapsulation**. We obtain speedups in the C implementation run-times (cycle counts) of the multiplication step compared to Bernstein's methods.

N1 is a multiplication algorithm in Char 3, with seven $1/4$ sized multiplications, which is derived by using the interpolation method in $\mathbb{F}_9$.

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

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,

# N1 4-way Split Algorithm

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

then,

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

# N1 4-way Split Algorithm

thus, the result of the multiplication becomes,

$$\begin{aligned} C(x) &= (A_0 + yA_1 + y^2 A_2 + y^3 A_3)(B_0 + yB_1 + y^2 B_2 + y^3 B_3) \\ &= C_0 + C_1 y + C_2 y^2 + C_3 y^3 + C_4 y^4 + C_5 y^5 + C_6 y^6 \end{aligned}$$

For interpolation we need 7 point but since $\mathbb{F}_3$ does not have enough points we use points from $\mathbb{F}_9$.

# N1 4-way Split Algorithm

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: 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 |
| $\omega\cdot(a+b\omega)=-b+a\cdot\omega$ | 0 | 0 |

Multiplying an element of $\mathbb{F}_9$ by $\omega$, 1, or $-1$ is cost-free.

# N1 4-way Split Algorithm

We choose $\{\omega, -\omega, \omega+1, -\omega+1, -\omega-1, \omega-1, \infty\}$ as the points of evaluation for interpolation and we get the following system of equations,

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

# N1 4-way Split Algorithm

Solving the matrix representation of the system of equation,

$$[V_{ij}]_{7x7} \cdot [C_i]_{7x1} = [P_j]_{7x1}$$

then,

$$\Rightarrow [C_i]_{7x1} = [V_{ij}]_{7x7}^{-1} \cdot [P_j]_{7x1}$$

# N1 4-way Split Algorithm

yields,

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

# N1 4-way Split Algorithm

where,

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

and observe that,

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

which helps us avoiding the cost of three 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.

# Complexity of N1 Algorithm

By using the cost of multi-evaluation and reconstruction tables we get,

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

then we get the explicit complexities as follows,

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

# Complexity of N1 4-way Algorithm - Unbalanced Split Version

Moreover, assuming that $A(x)$ and $B(x)$ are degree $3n+k-1$ polynomials where $1 \leq k \leq n$, i.e the size of the polynomials to be multiplied are not multiples of 4, $A_0, A_1, A_2, B_0, B_1, B_2$ 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 yields,

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

# Comparison of N1 Algorithm to Others

- **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 $\mathbb{F}_9[x]$.
- **N1** is faster than than **B1** for $n \geq 192$.

# An Improved 4-way Multiplication Algorithm (N2)

**N2 is an improved version of N1:**

-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.

# N2 4-way Split Algorithm

More simplifications than N1 case as follows:

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

$P_3$ 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.

# N2 4-way Split Algorithm

This time the coefficients of the multiplication polynomial becomes:

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

# Complexity of N2 Algorithm

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

And we get the following asymptotic complexities:

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

# Complexity of N2 4-way Algorithm - Unbalanced Split Version

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 yields,

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

# Comparison of N2 with Others

- **N2** becomes faster 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 $\mathbb{F}_9[x]$.
- **N2** is faster than than **B1** for $n \geq 192$
- In general, **N2** is more efficient than **N1** for all input sizes.

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

**N3** is another improved 4-way split multiplication algorithm in **Char 3**, with **seven** $1/4$ sized multiplications, using Lagrange interpolation in $\mathcal{R} = \mathbb{F}_9[x]$ with evaluation points $\{0, 1, -1, x, \omega, -\omega, \infty\}$ we get,

$$\left.\begin{array}{l}
P_0 = A_0 B_0 = C(0) \\
P_1 = (A_0 + A_1 + A_2 + A_3)(B_0 + B_1 + B_2 + B_3) = C(1) \\
P_2 = (A_0 - A_1 + A_2 - A_3)(B_0 - B_1 + B_2 - B_3) = C(-1) \\
P_3 = (A_0 + A_1 x + A_2 x^2 + A_3 x^3)(B_0 + B_1 x + B_2 x^2 + B_3 x^3) = C(x) \\
P_4 = [(A_0 - A_2) + \omega(A_1 - A_3)][(B_0 - B_2) + \omega(B_1 - B_3)] = C(\omega) \\
P_5 = [(A_0 - A_2) - \omega(A_1 - A_3)][(B_0 - B_2) - \omega(B_1 - B_3)] = C(-\omega) \\
P_6 = A_3 B_3 = C_6
\end{array}\right\}$$

# New 4-way Split Algorithm (N3)

Let,

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

then one can observe that,

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

By the above two equations, 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:

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

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

# New 4-way Split Algorithm (N3)

N3 is not 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 $n+2$.

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

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.

Then we get the complexity of the N3 algorithm as follows:

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

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

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 for $(n+1)/2 \leq k$. Then, the cost analysis of the N3 4-way algorithm yields,

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

# Comparison of N3 Algorithm to Others

In terms of arithmetic complexity,

- **N3** 4-way algorithm is better than the **N1** and **N2** 4-way algorithms [5] for $n \geq 64$.
- Also note that, all of **N1, N2, and N3** 4-way methods are faster than Bernstein's 3-way algorithm **B1** [6] for $n \geq 192$ in the implementation run-times.

Let,

$$\left.\begin{aligned} A(x) &= a_0 + a_1x + a_2x^2 + \dots + a_{5n-1}x^{5n-1} \\ B(x) &= b_0 + b_1x + b_2x^2 + \dots + b_{5n-1}x^{5n-1} \end{aligned}\right\}$$

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,

$$\left.\begin{aligned} A_0 &= a_0 + a_1x + \dots + a_{n-1}x^{n-1} \\ A_1 &= a_n + a_{n+1}x + \dots + a_{2n-1}x^{n-1} \\ A_2 &= a_{2n} + a_{2n+1}x + \dots + a_{3n-1}x^{n-1} \\ A_3 &= a_{3n} + a_{3n+1}x + \dots + a_{4n-1}x^{n-1} \\ A_4 &= a_{4n} + a_{4n+1}x + \dots + a_{5n-1}x^{n-1} \end{aligned}\right\}$$

# V1 5-way Split Algorithm

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

and the multiplication has the following form,

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

# V1 5-way Split Algorithm

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

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

# Complexity of V1 Algorithm

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

by Remark 1 we get,

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

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 yields,

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

# Comparison of V1 with Others

- **V1** becomes more efficient than **KA2** for $n \geq 100$ in $\mathbb{F}_3[x]$ and for $n \geq 20$ in $\mathbb{F}_9[x]$..
- **V1** is better than **A3** for $n \geq 60$ in $\mathbb{F}_3[x]$ and for $n \geq 15$ in $\mathbb{F}_9[x]$.
- In general, **V1** is the most efficient among all algorithms including **B1**, **N1**, **N2**, and **N3** for all input sizes.

# Unbalanced 5-way Split Multiplication Algorithm (U1)

- Assume that $A(x)$ and $B(x)$ degree $5n - k - 1$ polynomials, $n \in \mathbb{Z}^+$ and $n \geq 5$.
- Let $k \in \{0, 1, 2, 3, 4\}$. If $5n - k$ 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 $(5n - k + k)/5 = n$ elements and the **last one** has $(5n - k - 4k)/5 = n - k$ elements.
- By this means, we get an **unbalanced** 5-way division method for any polynomial with size $n \geq 5$.

Let $y = x^n$ and $C(x) = A(x)B(x)$ then $A(x)$ and $B(x)$ are divided into five parts as follows:

$$\left.\begin{aligned}
A_0 &= a_0 + a_1x + \ldots + a_{n-1}x^{n-1} \\
A_1 &= a_n + a_{n+1}x + \ldots + a_{2n-1}x^{n-1} \\
A_2 &= a_{2n} + a_{2n+1+1}x + \ldots + a_{3n-1}x^{n-1} \\
A_3 &= a_{3n} + a_{3n+1}x + \ldots + a_{4n-1}x^{n-1} \\
A_4 &= a_{4n} + a_{4n+1}x + \ldots + a_{5n-k-1}x^{n-k-1}
\end{aligned}\right\}$$

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

# Complexity of U1 Algorithm

**COMPLEXITY:**

- By using the cost of multi-evaluation and reconstruction tables, the **complexity of U1** can be calculated as below:

$$\left.\begin{array}{l} M_9(5n-k) \leq 8M_9(n) + M_9(n-k) + 196n - 24k - 72, M_9(1) = 6 \\ M_3(5n-k) \leq 2M_3(n) + M_3(n-k) + 3M_9(n) + 72n - 6k - 29, M_3(1) = 1 \end{array}\right\}$$

- Observe that, for $k = 0$, the **U1** algorithm yields the **V1** algorithm, so we can think of **V1** as a special case of the **U1** algorithm.

**COMPARISON TO OTHER ALGORITHMS:**

- According to the arithmetic costs and the implementation run-times, the use of **U1** algorithm yields **fastest run-time of all** algorithms.

- The decapsulation phase of the Streamlined NTRU Prime Key Encapsulation Mechanism (KEM) conducts a polynomial multiplication operation for multiplying the elements of $\mathbb{Z}_3[x]/(x^p - x - 1)$ for parameters $p = 653, 761$.
- Thus, we can apply the proposed 4-way and 5-way polynomial multiplication algorithms N1, N2, N3, and V1 to it.
- Bernstein uses 2 different methods. First method is **Hybrid-1** [2] and the second method is **B1-Hyrid** [6].

# Bernstein's Hybrid-1 Algorithm for NTRU Prime Decapsulation

**Hybrid-1 Multiplication Algorithm: 5 KA2 then SB (n=768)**:

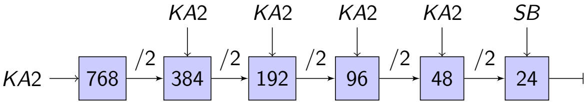


Figure: Hybrid-1 Algorithm requires a total # of 303600 arithmetic operations

Bernstein *et al.* use a combination of five layers of KA2 and then SB for multiplying the input size $n = 768$ (zero-padded from 761 coefficient inputs) polynomials in the Streamlined NTRU Prime decapsulation phase.

# Our Alternative Approaches for Hybrid-1: Hybrid-2 Multiplication Algorithm

**First Alternative Method for Hybrid-1:**

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

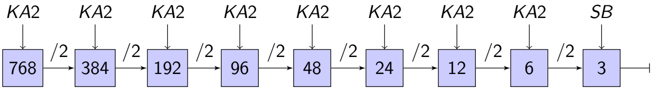


Figure: Hybrid-2 Algorithm requires a total # of 207858 arithmetic operations

**Second Alternative Method for Hybrid-1:**

**(ii) N1-Hybrid Algorithm: The new N1 algorithm is used in:**

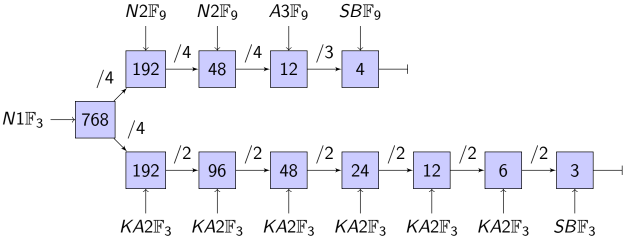


Figure: N1-Hybrid Algorithm requires a total # of 187152 arithmetic operations

**Third Alternative Method for Hybrid-1:**

**(iii) N2-Hybrid Multiplication Algorithm (n=768)**

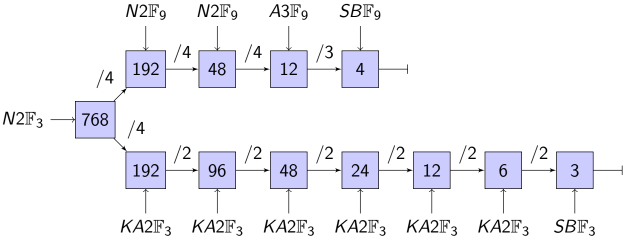


Figure: N2-Hybrid Algorithm requires a total # of 180878 arithmetic operations

## (iv) V1-Hybrid Multiplication Algorithm (n=765):

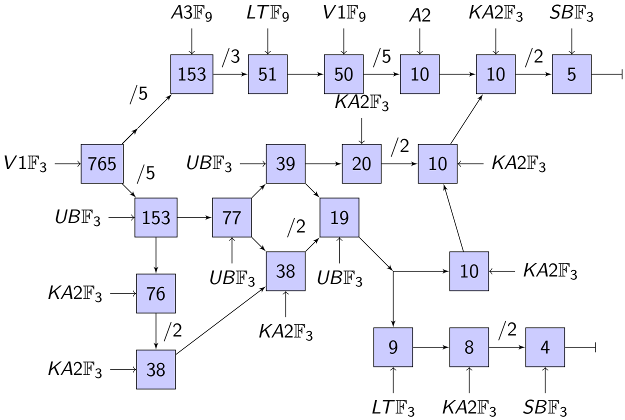


Figure: V1-Hybrid Algorithm requires a total # of 182647 arithmetic operations

## (v) A3-Hybrid Multiplication Algorithm (n=768):

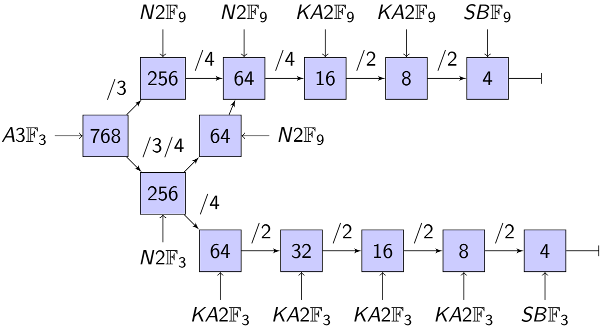


Figure: A3-Hybrid Algorithm requires a total # of 189115 arithmetic operations

**(vi) LT-Hybrid Multiplication Algorithm (n=761):**

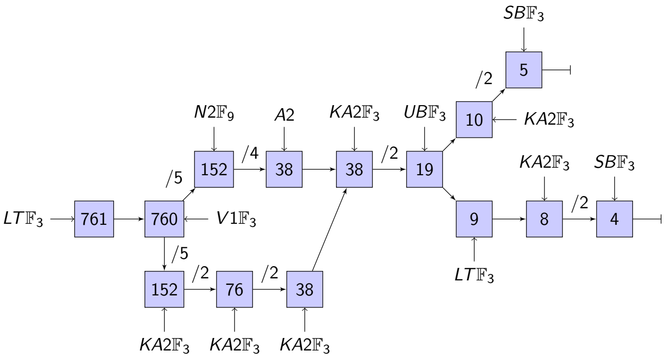


Figure: LT-Hybrid Algorithm requires a total # of 186914 arithmetic operations

# Comparison of the Arithmetical Complexities of the New Hybrid Algorithms Including N1, N2, and V1

Table: Comparison of the Arithmetical 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 | 303600 | **Reference** | method used in [2] for `sntrup761` |
| 768 | Hybrid-2 | 207858 | 31.53% | min. cost: **before this study [5]** |
| 768 | A3-Hybrid | 189115 | 37.70% | this work [5] |
| 768 | N1-Hybrid | 187152 | 38.35% | this work [5] |
| 761 | LT-Hybrid | 186914 | 38.43% | this work [5] |
| 765 | V1-Hybrid | 182647 | 39.83% | this work [5] |
| 768 | N2-Hybrid | 180878 | 40.42% | min. cost: **after this study [5]** |

# Implementation Results of the New Hybrid Algorithms Including N1, N2, and V1

Table: 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 [2] for `sntrup761` |
| 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% |

# Comparison Results to Bernstein's Hybrid-1 Method

- ✓ **N1-Hybrid2** 37.39% faster than **Hybrid-1**.
- ✓ **A3-Hybrid2** 34.04% faster than **Hybrid-1**.
- ✓ Therefore **N1-Hybrid2** can be a better alternative for Char 3 polynomial multiplication in NTRU Prime decapsulation.

**B1-Hybrid1 Algorithm for** $n = 653$

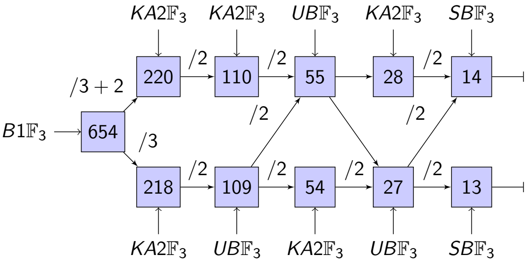


Figure: B1-Hybrid1 Algorithm cycles/time is 758.027/0.000329

**B1-Hybrid2 Algorithm for** $n = 761$

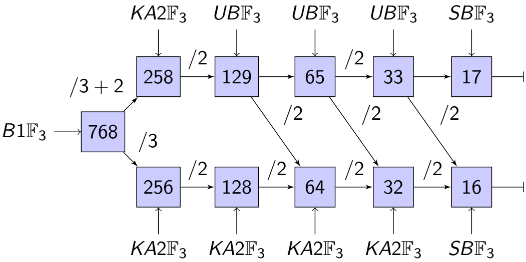


Figure: B1-Hybrid2 Algorithm cycles/time is 944.139/0.000410

## U1-Hybrid1 Algorithm for $n = 653$

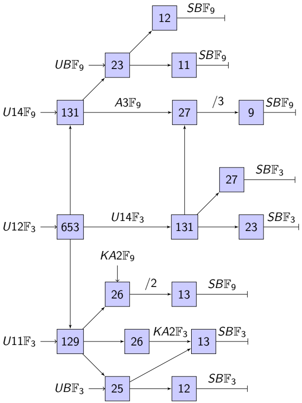


Figure: U1-Hybrid1 Algorithm cycle/time is 531.692/0.000231

# Our Alternative Approach to B1-Hybrid2: U1-Hybrid2 for $n = 761$

## U1-Hybrid2 Algorithm for $n = 761$

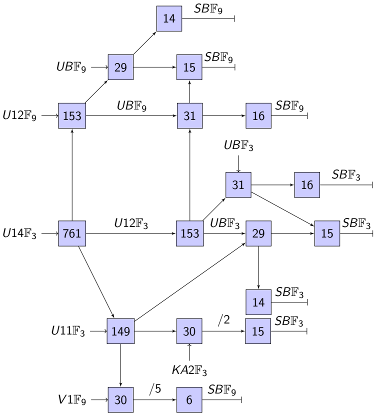


Figure: U1-Hybrid2 Algorithm cycle/time is 608.694/0.0000265

# Implementation Results for New Hybrid Algorithms

Table: Implementation Results for Polynomial Multiplication over $\mathbb{F}_3$ in Streamlined NTRU Prime Decapsulation

| Parameter | Algorithm | Cycles/Time | Improvement |
|---|---|---|---|
| **sntrup653** | **B1-Hybrid1** (Bernstein's B1 [6]) | $758,027/0.000329$ | Ref. Value |
| | **U1-Hybrid1** (this work [9]) | $531,692/0.000231$ | 29.85% |
| **sntrup761** | **B1-Hybrid2** (Bernstein's B1 [6]) | $944,139/0.000410$ | Ref. Value |
| | **Hybrid-1** (Bernstein's prev. [2]) | $1,054,828/0.000458$ | -10.49% |
| | **U1-Hybrid2** (this work [9]) | $608,694/0.0000265$ | 35.52% |
| | **N1-Hybrid2** (this work [5]) | $665,729/0.0000289$ | 29.48% |

`https://github.com/cryptoarith/F3Mul`
`https://github.com/cryptoarith/NTRUPrimePolyMultF3`

# Conclusion

- The use of new algorithms N1, N2, N3, V1, and U1 provides improved in arithmetical complexities in **Char 3** polynomial multiplication:

  \- For instance, 48.6% **reduction in arithmetic complexity** for polynomial multiplication in $\mathbb{F}_9[x]$ and a 26.8% **reduction** for polynomial multiplication in $\mathbb{F}_3[x]$ for $n = 1280$.

- The proposed **U1-Hybrid1** is 29.85% faster than the Bernstein's **B1-Hybrid1** algorithm for $n = 653$.
- The proposed **U1-Hybrid2** is 35.52% faster than the Bernstein's **B1-Hybrid2** algorithm for $n = 761$.
- Therefore **U1-Hybrid1** for **U1-Hybrid2** can be better alternatives for Char 3 polynomial multiplication in NTRU Prime decapsulation.

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

[2] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. van Vredendaal. Ntru prime. Reducing the attack surface at low cost, International Conference on Selected Areas in Cryptography, 24, pp, 235-260, August 2017. 2019. https://ntruprime.cr.yp.to/nist/ntruprime-20201007.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, july 2020. https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf.

[4] NIST Post Quantum Cryptography *PQC* Standardization 2016-2020. https://csrc.nist.gov/projects/post-quantum-cryptography.

[5] E. Yeniaras and M. Cenk Faster Characteristic Three Polynomial Multiplication and Its Application to NTRU Prime Decapsulation,Journal of Cryptographic Engineering, 1-20, 2022. https://eprint.iacr.org/2020/1336.pdf

[6] D. Bernstein, B.B. Brumley, M. Chen, N. Tuveri OpenSSLNTRU: Faster post-quantum TLS key exchange, IACR Cryptol. ePrint Arch., 826, 2021. https://eprint.iacr.org/2021/826.pdf

[7] 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):285–294, march 2018.

[8] D.J. Bernstein. Batch Binary Edwards. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*, pages 317–336, 2009.

[9] E. Yeniaras. and M. Cenk Improved polynomial multiplication algorithms over characteristic three fields and applications to NTRU Prime. *Innovative Security Solutions for Information Technology and Communications, SecITC 2021, LNCS 13195.*

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

*Thank You For Listening!*