TMVP-based Multiplication for Polynomial Quotient Rings and Application to Saber on ARM Cortex-M4*

İrem Keskinkurt Paksoy †
Murat Cenk ‡
Institute of Applied Mathematics, Middle East Technical University

Abstract

Lattice-based NIST PQC finalists need efficient multiplication in $\mathbb{Z}_q[x]/\langle f(x)\rangle$. Multiplication in this ring can be performed very efficiently via number theoretic transform (NTT) as done in CRYSTALS-Kyber if the parameters of the scheme allow it. If NTT is not supported, other multiplication algorithms must be employed. For example, if the modulus q of the scheme is a power of two as in Saber and NTRU, then NTT can not be used directly. In this case, Karatsuba and Toom-Cook methods together with modular reduction are commonly used for multiplication in this ring. In this paper, we show that the Toeplitz matrix-vector product (TMVP) representation of modular polynomial multiplication yields better results than Karatsuba and Toom-Cook methods. We present three- and four-way TMVP formulas that we derive from three- and four-way Toom-Cook algorithms, respectively. We use the four-way TMVP formula to develop an algorithm for multiplication in the ring $\mathbb{Z}_{2^m}[x]/\langle x^{256}+1\rangle$. We implement the proposed algorithm on the ARM Cortex-M4 microcontroller and apply it to Saber, which is one of the lattice-based finalists of the NIST PQC competition. We compare the results to previous implementations. The TMVP-based multiplication algorithm we propose is 20.83% faster than the previous algorithm that uses a combination of Toom-Cook, Karatsuba, and schoolbook methods. Our algorithm also speeds up key generation, encapsulation, and decapsulation algorithms of all variants of Saber. The speedups vary between $4.3-39.8\%$. Moreover, our algorithm requires less memory than the others, except for the memory-optimized implementation of Saber.

Keywords: Lattice-based cryptography, Post-quantum cryptography, ARM Cortex-M4, MLWR, Saber, Toeplitz, TMVP

*The first author is partially supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) 2211-c graduate scholarship program and Council of Higher Education (YÖK) 100/2000 PhD scholarship program.

†irem@metu.edu.tr

‡mcenk@metu.edu.tr

1 Introduction

Since the beginning of the NIST post-quantum standardization competition [29], lattice-based cryptographic schemes have been compelling candidates. In July 2020 NIST announced the third round finalists [28], and three out of four PKE/KEM finalists are lattice-based schemes: CRYSTALS-Kyber [5], NTRU [18, 9], and Saber [12, 13]. These schemes are defined on polynomial rings of the form $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$ where $f(x) \in \mathbb{Z}[x]$ is a degree n polynomial. Multiplication in these rings has a major effect on the efficiency of the schemes. The most efficient polynomial multiplication algorithm used in implementations varies depending on the values of the parameters n and q. The parameters of the schemes such as NewHope [2] and CRYSTALS-Kyber [5] are convenient for using the 'Number Theoretic Transform (NTT)' [11], which is the most efficient polynomial multiplication algorithm known. The downside of NTT is the constraint on the parameters, which also limits the security levels of the schemes. The polynomial rings that Saber [12, 13] and NTRU [9, 17, 18] operate on are not NTT friendly. In such polynomial rings, Toom-Cook [10, 32] and Karatsuba [22] are the most preferred algorithms for multiplication. These algorithms or combinations of these algorithms are more efficient than the naive method, but they are not as efficient as NTT. In [19], the use of Karatsuba and Toom-Cook methods in the optimized ARM Cortex-M4 implementations of some schemes can be seen. Albeit not commonly used, Toeplitz matrix-vector product (TMVP) based algorithms are good alternatives for residue polynomial multiplication [33]. Toeplitz matrices appear in various cryptographic applications. For detailed information about the use of Toeplitz matrices in cryptography, we refer to the reader to [24, 25, 14, 7, 8, 1, 31]. TMVP-based algorithms include the polynomial reduction step inside the multiplication process, whereas Karatsuba and Toom-Cook methods require additional reduction. This property makes TMVP-based algorithms more advantageous in some cases considering the high degree modulo polynomials and the number of multiplications performed in schemes.

In this paper, we work on improving multiplication in $\mathcal{R}_q$ with $q = 2^k$ by developing TMVP-based algorithms. The main objective of this work is to speed up the cryptographic schemes that do not support NTT such as Saber and NTRU which are the lattice-based finalists of the NIST post-quantum standardization competition . Following this purpose, we present three- and four-way Toeplitz matrix-vector product formulas TMVP-3 and TMVP-4 that we derive from Toom-3 and Toom-4 methods, respectively, using the technique explained in [33]. Moreover, we propose an algorithm for TMVP-based multiplication in the ring $\mathbb{Z}_{2^m}[x]/\langle x^{256}+1\rangle$ exploiting TMVP-4 formula. We implement the proposed algorithm on ARM Cortex-M4. The proposed polynomial multiplication algorithm surpasses the one given in [19] that uses a combination of Toom-4, Karatsuba, and schoolbook polynomial multiplication by 20.16%. We integrate this algorithm to Saber by replacing the assembly code for polynomial multiplication in the software package accompanying the paper [19] with ours. We speed up the key generation by 7.93%, encapsulation by 8.67% and decapsulation by 10.67%.

The application of the proposed algorithm in this paper is not explicitly optimized for Saber. In [27], such optimization is done by using the *lazy interpolation* and the *precomputation* methods. These methods reduce the number of evaluation and recombination steps, hence increases efficiency. The lazy interpolation method for polynomials can be thought of as the counterpart of the *block recombination* method [15] for matrices. Thereby, the proposed algorithm in this paper can be optimized for Saber using the block recombination method, and similar improvements in [27] can be achieved.

Based on the results we receive from our application to Saber, TMVP-based residue polynomial multiplication algorithms seem to work well for the rings that are not NTT friendly, and they might be good alternatives to Karatsuba and Toom-Cook methods. Apparently, lattice-based cryptosystems are very popular for post-quantum cryptography, and research on the efficiency of lattice-based schemes will continue for a long time. To the best of our knowledge, TMVP-based algorithms have not been used in post-quantum cryptography applications so far. Having options other than Karatsuba and Toom-Cook methods for multiplication in $\mathcal{R}_q$ when NTT is out of the question would provide diversity in research.

Availability of the software: All source codes are available at `https://github.com/iremkp/Saber_tmvp4_m4`

Organization of this paper: In Section 2, we provide preliminary information and describe the notations we use throughout this paper. We explain the derivation of the new TMVP-3 and TMVP-4 formulas and introduce the algorithm we propose for multiplication in $\mathbb{Z}_{2^m}[x]/\langle x^{256}+1\rangle$ in Section 3. Finally, in Section 4, we summarize our work and conclude the paper with some ideas for future study.

2 Preliminaries

In this section, we introduce some definitions and properties to build a background. Throughout the paper, we use the notation $M_{ALG}(n)$ to state the arithmetic complexity of the algorithm ALG for dimension n, and $M(n)$ to state the arithmetic complexity of the most efficient algorithm for the computation in question for dimension n. $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n+1\rangle$ denotes the finite polynomial ring modulo x^n+1 where the coefficients of the polynomials are integers in $\mathbb{Z}_q$.

2.1 Toeplitz Matrix Vector Product

There are many cryptographic applications utilize theToeplitz matrix-vector product (TMVP) in the literature. The use of TMVP in cryptographic computations first appeared in [14] for multiply elements of binary extension fields. Then, many proposals were suggested [1, 14, 15, 16, 30, 31]. Recently, in [1] and [31] the use of TMVP for integer modular multiplication is proposed to speed up the residue multiplication modulo the Mersenne prime $2^{521}-1$ and the prime $2^{255}-19$ respectively. TMVP can be used to calculate the product of two polynomials modulo a polynomial as explained in [33].

Definition 2.1. *Let m and n be two positive integers. A Toeplitz matrix T is an $m \times n$ matrix whose entry in i-th row and j-th column is defined as $T_{i,j} = T_{i-1,j-1}$ for $i = 2, \dots, m$ and $j = 2, \dots n$.*

Throughout this paper, we focus only on square Toeplitz matrices:

$$T = \begin{pmatrix} a_0 & a'_1 & a'_2 & \dots & \dots & \dots & a'_{n-1} \\ a_1 & a_0 & a'_1 & a'_2 & \dots & \dots & \vdots \\ a_2 & a_1 & a_0 & a'_1 & \ddots & & \vdots \\ \vdots & a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & a'_1 & a'_2 \\ \vdots & & \ddots & \ddots & a_1 & a_0 & a'_1 \\ a_{n-1} & \dots & \dots & \dots & a_2 & a_1 & a_0 \end{pmatrix}. \tag{1}$$

The matrix T in (1) shows the special form of an $n \times n$ Toeplitz matrix. Clearly, specifying only $2n-1$ of its elements would suffice to identify T. Therefore, addition of two Toeplitz matrices requires only $2n-1$ additions while addition of regular matrices requires n^2. Moreover, every submatrix of a Toeplitz matrix is also a Toeplitz matrix. These properties become very handy when it comes to calculating a TMVP efficiently. Instead of using the naive matrix vector multiplication, the divide and conquer method works very well for TMVP for large n. Suppose we want to compute the product of the Toeplitz matrix T in (2.1) by a vector B where the transpose of B is $B^T = (b_0, b_1, \dots, b_n)$. We may apply different splitting methods [16] to compute the following TMVP:

$$T.B = \begin{pmatrix} a_0 & a'_1 & \dots & a'_{n-2} & a'_{n-1} \\ a_1 & a_0 & \dots & a'_{n-3} & a'_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-2} & a_{n-3} & \dots & a_0 & a'_1 \\ a_{n-1} & a_{n-2} & \dots & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix}. \tag{2}$$

For example, a two-way TMVP formula allows us to compute an n dimensional TMVP via three $n/2$ dimensional TMVPs. For this, we denote the TMVP in (2) by

$$T.B = \begin{pmatrix} T_1 & T_0 \\ T_2 & T_1 \end{pmatrix} \begin{pmatrix} B_0 \\ B_1 \end{pmatrix}$$

where T_0, T_1, T_2 are $\frac{n}{2} \times \frac{n}{2}$ Toeplitz matrices and B_0, B_1 are $\frac{n}{2} \times 1$ matrices. The n dimensional Toeplitz matrix vector product $T.B$ can be calculated as follows:

$$\begin{pmatrix} T_1 & T_0 \\ T_2 & T_1 \end{pmatrix} \begin{pmatrix} B_0 \\ B_1 \end{pmatrix} = \begin{pmatrix} P_1 + P_2 \\ P_1 - P_3 \end{pmatrix},$$

where

$$\begin{aligned} P_1 &= T_0(B_0 + B_1), \\ P_2 &= (T_0 - T_1)B_1, \\ P_3 &= (T_1 - T_2)B_0. \end{aligned}$$

The arithmetic complexity of the given two-way TMVP formula (TMVP-2) above is $M_{TMVP-2}(n) = 3M(n/2) + 3n - 1$. Similarly, a three-way TMVP formula allows us to compute an n dimensional TMVP via six $n/3$ dimensional TMVPs. For this, we denote (2) by

$$T.B = \begin{pmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \\ T_4 & T_3 & T_2 \end{pmatrix} \begin{pmatrix} B_0 \\ B_1 \\ B_2 \end{pmatrix}$$

where T_0, T_1, T_2, T_3, T_4 are $\frac{n}{3} \times \frac{n}{3}$ Toeplitz matrices and B_0, B_1, B_2 are $\frac{n}{3} \times 1$ matrices. The n dimensional Toeplitz matrix vector product $T.B$ can be calculated as follows:

$$\begin{pmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \\ T_4 & T_3 & T_2 \end{pmatrix} \begin{pmatrix} B_0 \\ B_1 \\ B_2 \end{pmatrix} = \begin{pmatrix} P_1 + P_4 + P_5 \\ P_2 + P_4 + P_6 \\ P_3 + P_5 + P_6 \end{pmatrix},$$

where

$$\begin{aligned} P_1 &= (T_0 + T_1 + T_2)B_2, \\ P_2 &= (T_1 + T_2 + T_3)B_1, \\ P_3 &= (T_2 + T_3 + T_4)B_0, \\ P_4 &= T_1(B_1 + B_2), \\ P_5 &= T_2(B_0 + B_2), \\ P_6 &= T_3(B_0 + B_1). \end{aligned}$$

The arithmetic complexity of three-way TMVP algorithm is $M_{TMVP3}(n) = 6M(n/3) + 5n - 1$. These split formulas for TMVP can be derived from any polynomial multiplication formula. The technique of derivation of a TMVP formula from a polynomial multiplication algorithm is explained elaborately in [33]. We use the same technique given in [33] to derive three- and four-way TMVP formulas from Toom-3 and Toom-4 algorithms using five and seven multiplications, respectively. These formulas will be denoted by TMVP-3 and TMVP-4. We present the formulas and explain the derivation technique in detail in Section 3.

2.2 Polynomial Multiplication Modulo $x^n \pm 1$ via TMVP

Let us denote the product of the polynomials $a(x) = \sum_{i=0}^{n-1} a_i x^i$ and $b(x) = \sum_{i=0}^{n-1} b_i x^i$ in $\mathbb{Z}[x]$ by $c'(x) = \sum_{i=0}^{2n-2} c'_i x^i \in \mathbb{Z}[x]$ where $c'_i = \sum_{j+k=i} a_j b_k$. Let $\mathcal{R} = \mathbb{Z}[x]/\langle x^n \pm 1\rangle$. The product $c(x) = \sum_{i=0}^{n-1} c_i x^i$ of the polynomials $a(x)$ and $b(x)$ in $\mathcal{R}$ can be calculated by reducing $c'(x)$ modulo $x^n \pm 1$. Clearly, $c_i = c'_i \mp c'_{i+n}$ for $i = 0, \dots, n-2$ and $c_{n-1} = c'_{n-1}$. The coefficients of the polynomial $c(x)$ can be expressed as follows:

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} a_0 & \mp a_{n-1} & \mp a_{n-2} & \dots & \mp a_3 & \mp a_2 & \mp a_1 \\ a_1 & a_0 & \mp a_{n-1} & \dots & \mp a_4 & \mp a_3 & \mp a_2 \\ a_2 & a_1 & a_0 & \dots & \mp a_5 & \mp a_4 & \mp a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{n-2} & a_{n-3} & a_{n-4} & \dots & a_1 & a_0 & \mp a_{n-1} \\ a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix}. \qquad (3)$$

The matrix in (3) is a Toeplitz matrix which has a more special form. It contains only n different entries which are the coefficients of one of the multiplicand polynomials. Having n components instead of $2n-1$ means we can simplify the formulas we use. Therefore, we can calculate this type of TMVPs even more efficiently. To be more specific if we use a k-split method (TMVP-k formula) we would have k different components instead of $2k-1$. Thus, it allows us to reduce the number of additions and the memory usage.

2.3 Saber

Saber [12, 13] is a lattice-based key encapsulation mechanism (KEM) and one of the finalists of the NIST PQC standardization competition. Its security relies on the Module Learning with Rounding (MLWR) problem [4, 26]. Saber defines an IND-CPA secure public key encryption scheme (Saber.PKE) consisting of key generation (Saber.PKE.Keygen), encryption (Saber.PKE.Enc), decryption (Saber.PKE.Dec) algorithms as described in Algorithms 1, 2, 3. It uses a version of the Fujisaki-Okamoto transformation to have an IND-CCA secure key encapsulation mechanism (Saber.KEM), which also consists of three algorithms key generation (Saber.KEM.KeyGen), encapsulation (Saber.KEM.Encaps), decapsulation (Saber.KEM.Decaps).

Algorithm 1 Saber.PKE.KeyGen

```
1: $seed_{\boldsymbol{A}} \leftarrow \mathcal{U}(\{0,1\}^{256})$
2: $\boldsymbol{A} = \mathsf{gen}(seed_{\boldsymbol{A}}) \in \mathcal{R}_q^{l\times l}$
3: $r \leftarrow \mathcal{U}(\{0,1\}^{256})$
4: $\boldsymbol{s} = \beta_\mu(\mathcal{R}_q^{l\times 1}; r)$
5: $\boldsymbol{b} = ((\boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{h}) \mod q) \gg (\epsilon_q - \epsilon_p) \in \mathcal{R}_p^{l\times 1}$
6: **return** $(pk = (seed_{\boldsymbol{A}}, \boldsymbol{b}), sk = (\boldsymbol{s}))$
```

Algorithm 2 Saber.PKE.Enc($pk = (seed_{\boldsymbol{A}}, \boldsymbol{b}), m \in \mathcal{R}_2; r$)

```
1: $seed_{\boldsymbol{A}} \leftarrow \mathcal{U}(\{0,1\}^{256})$
2: **if** $r$ *is not specified* **then**
3:     $r \leftarrow \mathcal{U}(\{0,1\}^{256})$
4: $\boldsymbol{s'} = \beta_\mu(\mathcal{R}_q^{l\times 1}; r)$
5: $\boldsymbol{b'} = ((\boldsymbol{A}\boldsymbol{s'} + \boldsymbol{h}) \mod q) \gg (\epsilon_q - \epsilon_p) \in \mathcal{R}_p^{l\times 1}$
6: $v' = \boldsymbol{b}^T(\boldsymbol{s'} \mod p) \in \mathcal{R}_p$
7: $c_m = (v' + h_1 - 2^{\epsilon_p - 1}m \mod p) \gg (\epsilon_p - \epsilon_T) \in \mathcal{R}_T$
8: **return** $c = (c_m, \boldsymbol{b'})$
```

Algorithm 3 Saber.PKE.Dec($sk = \boldsymbol{s}, c = (c_m, \boldsymbol{b'})$)

```
1: $Qv = \boldsymbol{b'}^T(\boldsymbol{s} \mod p) \in \mathcal{R}_p$
2: $m' = ((v - 2^{\epsilon_p - \epsilon_T}c_m + h_2) \mod p) \gg (\epsilon_p - 1) \in \mathcal{R}_2$
3: **return** $m'$
```

The scheme specifies three values for the parameter l that determines the security level. The values $l = 2$ (LightSaber), $l = 3$ (Saber), $l = 4$ (FireSaber) provide level 1, level 3, level 5 security, respectively. Saber operates on the finite polynomial rings $\mathcal{R}_q = \mathcal{R}_{2^{13}} = \mathbb{Z}_{2^{13}}[x]/\langle x^{256}+1\rangle$ and $\mathcal{R}_p = \mathcal{R}_{2^{10}} = \mathbb{Z}_{2^{10}}[x]/\langle x^{256}+1\rangle$. Like most of the lattice-based cryptosystems defined on polynomial rings, multiplication directly affects the efficiency of the scheme. The rings $\mathcal{R}_q$ and $\mathcal{R}_p$ that Saber is defined on are not suitable for using the Number Theoretic Transform (NTT), which is the most efficient polynomial multiplication algorithm. Regardless of the platform, all implementations of Saber use a combination of Toom-Cook, Karatsuba, and schoolbook methods for efficient polynomial multiplication. Details of existing implementations on different platforms can be found in [19, 23, 27].

2.4 Implementation Platform: ARM Cortex-M4

We choose the ARM Cortex-M4 microcontroller as the implementation platform. The Cortex-M4 implements the ARMv7E-M instruction set and it is recommended by NIST as a reference implementation platform for evaluation of PQC candidates on microcontrollers. It has sixteen 32-bit registers and aside from Program Counter (PC) and Stack Pointer (SP) registers, they are all available for development. We use the STM32F4DISCOVERY development board which is used in many implementations of PQC candidates [3, 6, 19, 20, 23]. The ARM Cortex-M4 is designed especially for digital signal processing (DSP) and it supports many useful single instruction multiple data (SIMD) instructions that can perform parallel arithmetic operations on 16-bit halfwords of multiple registers in one cycle. In Table 1, descriptions of some instructions we use in our implementation are given.

Table 1: Example instructions

General data processing		
ADD Rd, Rn, Rm	$\mathsf{Rd} = \mathsf{Rn} + \mathsf{Rm}$	
USUB16 Rd, Rn, Rm	$\mathsf{Rd}_b = (\mathsf{Rn}_b - \mathsf{Rm}_b) \bmod 2^{16}$	$\mathsf{Rd}_t = (\mathsf{Rn}_t - \mathsf{Rm}_t) \bmod 2^{16}$
Multiply-Accumulate		
SMUADX Rd, Rn, Rm	$\mathsf{Rd} = \mathsf{Rn}_b\mathsf{Rm}_t + \mathsf{Rn}_t\mathsf{Rm}_b$	
SMLADX Rd, Rn, Rm, Rt	$\mathsf{Rd} = \mathsf{Rn}_b\mathsf{Rm}_t + \mathsf{Rn}_t\mathsf{Rm}_b + \mathsf{Rt}$	
Packing-Unpacking		
PKHBT Rd, Rn, Rm LSL # k	$\mathsf{Rd}_b = \mathsf{Rn}_b$	$\mathsf{Rd}_t = (\mathsf{Rm} \ll k)_t$
PKHTB Rd, Rn, Rm ASR # k	$\mathsf{Rd}_b = (\mathsf{Rm} \gg k)_b$	$\mathsf{Rd}_t = \mathsf{Rn}_t$

The indices b and t denote the bottom (bits $0-15$) and top (bits $16-31$) halfwords of the relevant register. The symbols $\ll$ and $\gg$ denote the left and right shifts respectively.

3 Our Work

We derive TMVP-3 and TMVP-4 formulas from Toom-3 and Toom-4 algorithms using the same technique given in [33], which require five and seven smaller TMVPs, respectively. We also propose a TMVP-based algorithm for multiplication in the ring $\mathcal{R}_{2^m} = \mathbb{Z}_{2^m}[x]/\langle x^{256}+1\rangle$ which utilizes our TMVP-4 formula, and implement it on the ARM Cortex-M4 microcontroller. An important note here is the proposed algorithm includes the polynomial reduction step and does not require additional polynomial reduction outside of polynomial multiplication, unlike Karatsuba and Toom-Cook. We integrate the assembly code of the proposed multiplication algorithm to an existing implementation of Saber that accompanying the paper [19], and we improve the efficiency of multiplication, key generation, encryption, and decryption algorithms comparing to the results given in [19]. Moreover, we reduce stack usage. This section explains the derivation of our new three- and four-way TMVP formulas and describes the proposed multiplication algorithm.

3.1 TMVP-3 formula from Toom-3 algorithm

Let $a(x) = a_0 + a_1x + a_2x^2$ and $b(x) = b_0 + b_1x + b_2x^2$ be two polynomials in $\mathbb{Z}[x]$. The product of these polynomials $a(x)b(x) = c(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4x^4$ can be calculated using different methods.

The coefficients of the product polynomial $c(x)$ are computed using the schoolbook method as follows:

$$\begin{aligned}
c_0 &= a_0b_0,\\
c_1 &= a_0b_1 + a_1b_0,\\
c_2 &= a_0b_2 + a_1b_1 + a_2b_0,\\
c_3 &= a_1b_2 + a_2b_1,\\
c_4 &= a_2b_2.
\end{aligned} \tag{4}$$

This computation requires 9 multiplications and 4 additions. On the other hand, using evaluation points $\{0, 1, -1, -2, \infty\}$ leads to

$$\begin{aligned}
c(0) &= a(0)b(0) = a_0b_0,\\
c(1) &= a(1)b(1) = (a_0 + a_1 + a_2)(b_0 + b_1 + b_2),\\
c(-1) &= a(-1)b(-1) = (a_0 - a_1 + a_2)(b_0 - b_1 + b_2),\\
c(-2) &= a(-2)b(-2) = (a_0 - 2a_1 + 4a_2)(b_0 - 2b_1 + 4b_2),\\
c(\infty) &= a(\infty)b(\infty) = a_2b_2.
\end{aligned}$$

The coefficients c_i of the product polynomial $c(x)$ are interpolated and the following equalities are obtained:

$$\begin{aligned}
c_0 &= c(0),\\
c_1 &= c(0)/2 + c(1)/3 - c(-1) + c(-2)/6 - 2c(\infty),\\
c_2 &= -c(0) + c(1)/2 + c(-1)/2 - c(\infty),\\
c_3 &= -c(0)/2 + c(1)/6 + c(-1)/2 - c(-2)/6 + 2c(\infty),\\
c_4 &= c(\infty).
\end{aligned} \tag{5}$$

This method is known as Toom-3.

Now, we derive a three way TMVP (TMVP-3) formula from Toom-3. To this end, we multiply each equation of both (4) and (5) that corresponds to c_i by a symbolic variable z_{4-i} for $i = 0, \dots, 4$ and we take the sum of all equations. Then, we rearrange the terms to obtain two equations of the form $z_4c_0 + z_3c_1 + z_2c_2 + z_1c_3 + z_0c_4 = k_2b_0 + k_1b_1 + k_0b_2$. From (4) we get

$$\begin{aligned} k_2 &= z_4a_0 + z_3a_1 + z_2a_2, \\ k_1 &= z_3a_0 + z_2a_1 + z_1a_2, \\ k_0 &= z_2a_0 + z_1a_1 + z_0a_2 \end{aligned} \tag{6}$$

and from (5) we get

$$\begin{aligned} k_2 =& \frac{1}{2}a_0\left(2z_4 + z_3 - 2z_2 - z_1\right) + \frac{1}{6}(a_0 + a_1 + a_2)\left(2z_3 + 3z_2 + z_1\right) \\ &+ \frac{1}{2}(a_0 - a_1 + a_2)\left(-2z_3 + z_2 + z_1\right) + \frac{1}{6}(a_0 - 2a_1 + 4a_2)\left(z_3 - z_1\right), \\ k_1 =& \frac{1}{6}(a_0 + a_1 + a_2)\left(2z_3 + 3z_2 + z_1\right) - \frac{1}{2}(a_0 - a_1 + a_2)\left(-2z_3 + z_2 + z_1\right) \\ &- \frac{1}{3}(a_0 - 2a_1 + 4a_2)\left(z_3 - z_1\right), \\ k_0 =& \frac{1}{6}(a_0 + a_1 + a_2)\left(2z_3 + 3z_2 + z_1\right) + \frac{1}{2}(a_0 - a_1 + a_2)\left(-2z_3 + z_2 + z_1\right) \\ &+ \frac{2}{3}(a_0 - 2a_1 + 4a_2)\left(z_3 - z_1\right) + a_2\left(-2z_3 - z_2 + 2z_1 + z_0\right). \end{aligned} \tag{7}$$

Clearly, (6) can be expressed as a TMVP which gives the left hand side of the equation (8) while the right hand side comes from (7). Finally, we have the following TMVP-3 formula:

$$\begin{pmatrix} z_2 & z_1 & z_0 \\ z_3 & z_2 & z_1 \\ z_4 & z_3 & z_2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} P_1 + P_2 + 4P_3 + P_4 \\ P_1 - P_2 - 2P_3 \\ P_0 + P_1 + P_2 + P_3 \end{pmatrix} \tag{8}$$

where

$$\begin{aligned} P_0 &= \frac{1}{2}a_0\left(2z_4 + z_3 - 2z_2 - z_1\right), \\ P_1 &= \frac{1}{6}(a_0 + a_1 + a_2)\left(2z_3 + 3z_2 + z_1\right), \\ P_2 &= \frac{1}{2}(a_0 - a_1 + a_2)\left(-2z_3 + z_2 + z_1\right), \\ P_3 &= \frac{1}{6}(a_0 - 2a_1 + 4a_2)\left(z_3 - z_1\right), \\ P_4 &= a_2\left(-2z_3 - z_2 + 2z_1 + z_0\right). \end{aligned}$$

Therefore, with this formulation, an n dimensional TMVP can be calculated via five smaller TMVPs whose dimensions are 1/3-rd of the original one. In [33, 14], three-way formulas for TMVP which requires six TMVPs of dimension $n/3$ can be seen. The arithmetic complexity of the new TMVP-3 formula is roughly $M_{TMVP-3}(n) = 5M(n/3) + 10n$. We ignore the scalar multiplication and shifting operations while calculating the computational complexity of the algortihms.

3.2 TMVP-4 formula from Toom-4 algorithm

Let $a(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ and $b(x) = b_0 + b_1x + b_2x^2 + b_3x^3$ be two polynomials in $\mathbb{Z}[x]$. The product of these polynomials $a(x)b(x) = c(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4x^4 + c_5x^5 + c_6x^6$ can be calculated using different methods. The coefficients c_i of the product polynomial $c(x)$ are computed using the schoolbook method as follows:

$$\begin{aligned}
c_0 &= a_0b_0,\\
c_1 &= a_0b_1 + a_1b_0,\\
c_2 &= a_0b_2 + a_1b_1 + a_2b_0,\\
c_3 &= a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0,\\
c_4 &= a_1b_3 + a_2b_2 + a_3b_1,\\
c_5 &= a_2b_3 + a_3b_2,\\
c_6 &= a_3b_3.
\end{aligned} \tag{9}$$

This computation requires 16 multiplication and 9 additions.

On the other hand, the set of evaluation points $\{0, 1, -1, 2, -2, 3, \infty\}$ leads to

$$\begin{aligned}
c(0) &= a_0b_0,\\
c(1) &= (a_0 + a_1 + a_2 + a_3)(b_0 + b_1 + b_2 + b_3),\\
c(-1) &= (a_0 - a_1 + a_2 - a_3)(b_0 - b_1 + b_2 - b_3),\\
c(2) &= (a_0 + 2a_1 + 4a_2 + 8a_3)(b_0 + 2b_1 + 4b_2 + 8b_3),\\
c(-2) &= (a_0 - 2a_1 + 4a_2 - 8a_3)(b_0 - 2b_1 + 4b_2 - 8b_3),\\
c(3) &= (a_0 + 3a_1 + 9a_2 + 27a_3)(b_0 + 3b_1 + 9b_2 + 27b_3),\\
c(\infty) &= a_3b_3.
\end{aligned}$$

The coefficients c_i of the product polynomial $c(x)$ are interpolated and the following equalities are obtained:

$$\begin{aligned}
c_0 &= c(0),\\
c_1 &= -c(0)/3 + c(1) - c(-1)/2 - c(2)/4 + c(-2)/20 + c(3)/30 - 12c(\infty),\\
c_2 &= -5c(0)/4 + 2c(1)/3 + 2c(-1)/3 - c(2)/24 - c(-2)/24 + 4c(\infty),\\
c_3 &= 5c(0)/12 - 7c(1)/12 - c(-1)/24 + 7c(2)/24 - c(-2)/24 - c(3)/24 + 15c(\infty),\\
c_4 &= c(0)/4 - c(1)/6 - c(-1)/6 + c(2)/24 + c(-2)/24 - 5c(\infty),\\
c_5 &= -c(0)/12 + c(1)/12 + c(-1)/24 - c(2)/24 - c(-2)/120 + c(3)/120 - 3c(\infty),\\
c_6 &= c(\infty).
\end{aligned} \tag{10}$$

This method is known as Toom-4.

To derive four way TMVP (TMVP-4) formula, first we multiply each equation of both (9) and (10) that corresponds to c_i by a symbolic variable z_{6-i} for $i = 0, \ldots, 6$. Then, we take the sum of all equations to obtain two equations of the form $z_6c_0 + z_5c_1 + z_4c_2 + z_3c_3 + z_2c_4 + z_1c_5 + z_0c_6 = k_3b_0 + k_2b_1 + k_1b_2 + k_0b_3$. From (9) and (10), we get the following TMVP-3, we get the TMVP-4 formula using the similar rearrangements as in TMVP-3:

$$\begin{pmatrix} z_3 & z_2 & z_1 & z_0 \\ z_4 & z_3 & z_2 & z_1 \\ z_5 & z_4 & z_3 & z_2 \\ z_6 & z_5 & z_4 & z_3 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} P_1 - P_2 + 8P_3 - 8P_4 + 27P_5 + P_6 \\ P_1 + P_2 + 4P_3 + 4P_4 + 9P_5 \\ P_1 - P_2 + 2P_3 - 2P_4 + 3P_5 \\ P_0 + P_1 + P_2 + P_3 + P_4 + P_5 \end{pmatrix} \tag{11}$$

where

$$\begin{aligned}
P_0 &= \frac{1}{12}a_0\left(12z_6 - 4z_5 - 15z_4 + 5z_3 + 3z_2 - z_1\right), \\
P_1 &= \frac{1}{12}(a_0 + a_1 + a_2 + a_3)\left(12z_5 + 8z_4 - 7z_3 - 2z_2 + z_1\right), \\
P_2 &= \frac{1}{24}(a_0 - a_1 + a_2 - a_3)\left(-12z_5 + 16z_4 - z_3 - 4z_2 + z_1\right), \\
P_3 &= \frac{1}{24}(a_0 + 2a_1 + 4a_2 + 8a_3)\left(-6z_5 - z_4 + 7z_3 + z_2 - z_1\right), \\
P_4 &= \frac{1}{120}(a_0 - 2a_1 + 4a_2 - 8a_3)\left(6z_5 - 5z_4 - 5z_3 + 5z_2 - z_1\right), \\
P_5 &= \frac{1}{120}(a_0 + 3a_1 + 9a_2 + 27a_3)\left(4z_5 - 5z_3 + z_1\right), \\
P_6 &= a_3\left(-12z_5 + 4z_4 + 15z_3 - 5z_2 - 3z_1 + z_0\right).
\end{aligned} \tag{12}$$

Therefore with this formulation, an n dimensional TMVP can be calculated via seven smaller TMVPs whose sizes are 1/4-th of the original one. The arithmetic complexity of TMVP-4 formula is roughly $M_{TMVP-4}(n) = 7M(n/4) + 79n/4 - 27$. We ignore scalar multiplication and shifting operations in aritmetic complexity calculations.

3.3 TMVP-based Multiplication in $\mathcal{R}_{2^m} = \mathbb{Z}_{2^m}[x]/\langle x^{256} + 1\rangle$

Let $a(x) = \sum_{i=0}^{255} a_ix^i$ and $b(x) = \sum_{i=0}^{255} b_ix^i$ be two polynomials in the finite polynomial ring $\mathcal{R}_{2^m} = \mathbb{Z}_{2^m}[x]/\langle x^{256}+1\rangle$. The coefficients of the product polynomial $c(x) = \sum_{i=0}^{255} c_ix^i \in \mathcal{R}_{2^m}$ can be calculated via the following TMVP:

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_{254} \\ c_{255} \end{pmatrix} = \begin{pmatrix} a_0 & -a_{255} & -a_{254} & \ldots & -a_3 & -a_2 & -a_1 \\ a_1 & a_0 & -a_{255} & \ldots & -a_4 & -a_3 & -a_2 \\ a_2 & a_1 & a_0 & \ldots & -a_5 & -a_4 & -a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{254} & a_{253} & a_{252} & \ldots & a_1 & a_0 & -a_{255} \\ a_{255} & a_{254} & a_{253} & \ldots & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix}. \tag{13}$$

Applying the new TMVP-4 formula to the TMVP in (13) gives the following:

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} A_0 & -A_3 & -A_2 & -A_1 \\ A_1 & A_0 & -A_3 & -A_2 \\ A_2 & A_1 & A_0 & -A_3 \\ A_3 & A_2 & A_1 & A_0 \end{pmatrix} \begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix} = \begin{pmatrix} P_1 - P_2 + 8P_3 - 8P_4 + 27P_5 + P_6 \\ P_1 + P_2 + 4P_3 + 4P_4 + 9P_5 \\ P_1 - P_2 + 2P_3 - 2P_4 + 3P_5 \\ P_0 + P_1 + P_2 + P_3 + P_4 + P_5 \end{pmatrix} \tag{14}$$

where

$$\begin{aligned}
P_0 &= \frac{1}{12}(5A_0 - 15A_1 - 3A_2 + 9A_3)\,B_0, \\
P_1 &= \frac{1}{12}(-7A_0 + 8A_1 + 11A_2 + 2A_3)\,(B_0 + B_1 + B_2 + B_3), \\
P_2 &= \frac{1}{24}(-A_0 + 16A_1 - 13A_2 + 4A_3)\,(B_0 - B_1 + B_2 - B_3), \\
P_3 &= \frac{1}{24}(7A_0 - A_1 - 5A_2 - A_3)\,(B_0 + 2B_1 + 4B_2 + 8B_3), \\
P_4 &= \frac{1}{120}(-5A_0 - 5A_1 + 7A_2 - 5A_3)\,(B_0 - 2B_1 + 4B_2 - 8B_3), \\
P_5 &= \frac{1}{120}(-5A_0 + 3A_2)\,(B_0 + 3B_1 + 9B_2 + 27B_3), \\
P_6 &= (15A_0 + 3A_1 - 9A_2 + 5A_3)\,B_3,
\end{aligned}$$

and the partitions A_i are regular Toeplitz matrices of dimension 64, and B_i, C_i are vectors of length 64 for $i = 0, \ldots, 3$. Clearly, when the Toeplitz matrix has a more special form as in (14) the computation of P_i become simpler than the ones given in (12). The number of additions required to compute P_i decreases by $4n = 1024$.

Utilizing TMVP formulas allows us to split our computation into many similar computations of smaller size. We can use these splitting methods consecutively to reduce the dimension to a level that the schoolbook matrix-vector multiplication is more efficient than using the formulas. TMVP formulas are more efficient than schoolbook matrix-vector multiplication for large n values, but for small dimensions like $n = 2$, the schoolbook method is more efficient than TMVP formulas. The level of switching the multiplication method to the schoolbook, i.e., the threshold, might differ depending on the value of dimension n, the modulus q, the formula being used, and the implementation platform. The threshold must be chosen carefully depending on those factors to develop efficient algorithms. In our case, we want to establish a TMVP-based multiplication algorithm for Saber and implement it on the ARM Cortex-M4. To make use of the benefits of SIMD instructions, we use the same strategy in [19] and place the components of the matrices (or equivalently, the coefficients of the polynomials) into registers pairwise. It means that we operate on modulo 2^{16}. In other words, we develop an algorithm for multiplication in $\mathcal{R}_{2^{16}}$, and then we apply a modular reduction to obtain a result in $\mathcal{R}_{2^m}$. Fortunately, this reduction is made easily by shifting right by $16 - m$ bits. Since some of TMVP formulas require integer divisions by powers of two, modular reduction may cause a loss in the most significant bits. A formula that requires a division by 2^r can work correctly if $m + r \leq 16$ for the modulus 2^m. To be more precise, for the modulus $q = 2^{13} = 2^m$ of Saber, we have $r \leq 3$; that is, any method that requires a division by 2^r with $r \leq 3$ works correctly. That is, we can afford to lose at most three bits. We already apply a layer of TMVP-4 formula and lose the maximum number of bits we can. For seven TMVPs of dimension 64, we can not use a formula that contains a division by a power of two. It leaves us only two options: the TMVP-2 formula or the schoolbook matrix-vector multiplication since none of them require division by a power of two. At this point, we must determine the threshold, i.e., the dimension at which the schoolbook matrix-vector multiplication is better than the TMVP-2 formula. For this, we implement the schoolbook matrix-vector multiplication and the TMVP-2 formula for small dimensions and compare their cycle counts. Since $n = 256$ and we use only four and two-way split methods, we restrict our search to powers of two. In the next section, we give the threshold value and implementation results.

3.4 Implementation Results for Saber

As explained in the previous section, we need to determine the threshold for switching the multiplication method to complete our algorithm. We compare the schoolbook and two-way TMVP methods for the power of two dimensions. Table 2 shows the cycle counts of the schoolbook method and the TMVP-2 formula for small values of n. As can be seen in Table 2, the schoolbook method is faster than TMVP-2 for $n = 2$. For $n = 4$, we implement schoolbook matrix-vector multiplication and observe that $SB(4) = 29$. We know that the TMVP-2 formula calls three schoolbook methods of dimension 2, so we have $TMVP2(4) > 3 \times SB(2) = 30$. The schoolbook is the best method for $n = 4$ too. A similar observation shows that also for $n = 8$, the schoolbook is the best algorithm. For $n = 16$, schoolbook method takes 280 cycles which is not less than $3 \times 73 = 219$. So, we implement the TMVP-2 method to see whether it is better than the schoolbook method or not. But it takes 410 cycles, and hence for $n = 16$, the schoolbook method is the best. Finally, we do the same thing for $n = 32$ and conclude that 32 is the smallest dimension at which the TMVP-2 method is faster than the schoolbook method, so 16 is the threshold.

Table 2: Schoolbook vs. TMVP-2

n	$SB(n)$	$TMVP2(n)$
2	10	16
4	29	$> 3 \times 10 = 30$
8	73	$> 3 \times 29 = 87$
16	280	401
32	1313	1082

That is to say, the TMVP-based algorithm we propose for multiplication in $\mathcal{R}_{2^{16}} = \mathbb{Z}_{2^{16}}[x]/\langle x^{256}+1\rangle$ uses the TMVP-4 formula to split the computations into seven 64-dimensional TMVPs. Then, to each of these seven TMVPs, we apply the TMVP-2 formula twice and end up with 16-dimensional TMVPs. We perform sixty-three schoolbook matrix-vector multiplications and recombine their results according to the formulas to obtain the final result. We implement this algorithm on the ARM Cortex-M4 to compare the results with [19]. To make a fair comparison, we evaluate the polynomial multiplication algorithm in [19] with the polynomial reduction step since our algorithm already includes it. As can be seen in Table 3, our algorithm for multiplication in $\mathcal{R}_{2^{16}}$ is 20.83% faster and requires 15.89% less memory than the one in [19], which uses Toom-4 and the Karatsuba algorithms.

Table 3: Multiplication in $\mathcal{R}_{2^{16}}$

Cycles			Stack		
[19]	This work	Imp.	[19]	This work	Imp.
37804	29927	20.83%	3800	3196	15.89%

In this work, we only focus on multiplication, not on a complete optimized implementation of Saber. We use the publicly available codes from [19], and [20] to compose a software package for our application to Saber. We make some adjustments to existing codes to integrate our algorithm into this package. We compare the results with the ones given in [21], and [27]. Table 4 shows the comparison of the cycle counts and the stack usage on the ARM Cortex-M4 microcontroller of key generation, encapsulation, and decapsulation algorithms of all variants of Saber.

Table 4: Results of application to Saber

		[21]	[27] (speed)	[27] (memory)	This work	
LightSaber	KeyGen:	460 k	466 k	612 k	440 k	cycles
		9656	14208	3564	7956	bytes
	Encaps:	651 k	653 k	880 k	615 k	cycles
		11392	15928	3148	9684	bytes
	Decaps:	679 k	678 k	976 k	622 k	cycles
		12136	16672	3164	10428	bytes
Saber	KeyGen:	896 k	853 k	1230 k	825 k	cycles
		13256	19824	4348	12616	bytes
	Encaps:	1161 k	1103 k	1616 k	1060 k	cycles
		15544	22088	3412	14896	bytes
	Decaps:	1204 k	1127 k	1759 k	1073 k	cycles
		16640	23184	3420	15992	bytes
FireSaber	KeyGen:	1449 k	1340 k	2046 k	1319 k	cycles
		20144	26448	5116	20144	bytes
	Encaps:	1787 k	1642 k	2538 k	1621 k	cycles
		23008	29228	3668	22992	bytes
	Decaps:	1853 k	1679 k	2740 k	1649 k	cycles
		24592	30768	3684	24472	bytes

As can be seen in Table 4, our algorithm is the fastest compared to the others. Table 5 shows the percentage of the gain in terms of the execution time that our algorithm achieves. Our algorithm also reduces the stack usage more or less comparing to [21] and the speed-optimized version in [27]. The percentage of improvements in memory utilization can be seen in Table 6. Even though our algorithm requires less execution time, it consumes more memory, comparing the memory-optimized version in [27].

Table 5: Speed ups

		[21]	[27] (speed)	[27] (memory)
LightSaber	KeyGen:	-4.3%	-5.5%	-28.1%
	Encaps:	-5.5%	-5.8%	-30.1%
	Decaps:	-8.3%	-8.2%	-36.2%
Saber	KeyGen:	-7.9%	-3.2%	-32.9%
	Encaps:	-8.6%	-3.8%	-34.4%
	Decaps:	-10.8%	-4.7%	-38.9%
FireSaber	KeyGen:	-8.9%	-1.5%	-35.5%
	Encaps:	-9.2%	-1.2%	-36.1%
	Decaps:	-11.0%	-1.7%	-39.8%

Table 6: Improvements in memory utilization

		[21]	[27](speed)
LightSaber	KeyGen:	-17.6%	-44.0%
	Encaps:	-14.9%	-39.2%
	Decaps:	-14.0%	-37.4%
Saber	KeyGen:	-4.8%	-36.3%
	Encaps:	-4.1%	-32.5%
	Decaps:	-3.8%	-31.0%
FireSaber	KeyGen:	-	-28.8%
	Encaps:	-	-21.3%
	Decaps:	-	-20.4%

4 Conclusion/Discussion

In this work, we focus on the non-NTT style of multiplication algorithms for post-quantum candidates utilizing the Toeplitz matrix-vector product. We derive new Toeplitz matrix-vector product (TMVP) formulas TMVP-3 and TMVP-4 from Toom-3 and Toom-4 algorithms using a similar technique explained in [33]. Moreover, we propose an algorithm for multiplication in the ring $\mathcal{R}_{2^m} = \mathbb{Z}_{2^m}[x]/\langle x^{256}+1\rangle$ which is exploiting the new TMVP-4 formula. The proposed multiplication algorithm includes the polynomial reduction step, unlike the Toom-Cook and the Karatsuba algorithms. We implement the proposed algorithm on ARM Cortex-M4 and integrate it into an existing implementation of Saber. Even though our implementation is not optimized specifically for Saber, the results indicate that TMVP-based multiplication algorithms might be good alternatives to Toom-Cook and Karatsuba methods. For an optimized implementation specifically for Saber, the scheme-specific adjustments can be made using the block recombination method for matrices [15], like done in [27] using the lazy interpolation method. Based on the outcomes of this work, we expect that TMVP-based multiplication algorithms exploiting TMVP-3 and TMVP-4 formulas improve the efficiency of lattice-based post-quantum schemes that are not NTT friendly.

References

[1] Shoukat Ali and Murat Cenk. Faster residue multiplication modulo 521-bit mersenne prime and an application to ecc. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(8):2477–2490, 2018.

[2] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—a new hope. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 327–343, 2016.

[3] Erdem Alkim, Philipp Jakubeit, and Peter Schwabe. Newhope on arm cortex-m. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 332–349. Springer, 2016.

[4] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 719–737. Springer, 2012.

[5] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[6] Joppe W Bos, Simon Friedberger, Marco Martinoli, Elisabeth Oswald, and Martijn Stam. Fly, you fool! faster frodo for the arm cortex-m4. *IACR Cryptol. ePrint Arch.*, 2018:1116, 2018.

[7] Murat Cenk and M Anwar Hasan. Some new results on binary polynomial multiplication. *Journal of Cryptographic Engineering*, 5(4):289–303, 2015.

[8] Murat Cenk, Christophe Negre, and M Anwar Hasan. Improved three-way split formulas for binary polynomial and toeplitz matrix vector products. *IEEE Transactions on Computers*, 62(7):1345–1361, 2012.

[9] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hulsing, Joost Rijneveld, John M Schanck, Peter Schwabe, William Whyte, and Zhenfei Zhang. Ntru: algorithm specifications and supporting documentation (2019). *URL: https://csrc. nist. gov/projects/post-quantum-cryptography/round-2-submissions. Citations in this document*, 1.

[10] Stephen A Cook and Stål O Aanderaa. On the minimum computation time of functions. *Transactions of the American Mathematical Society*, 142:291–314, 1969.

[11] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[12] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In *International Conference on Cryptology in Africa*, pages 282–305. Springer, 2018.

[13] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Mod-lwr based kem. *Second PQC Standardization Conference, 2019, University of California, Santa Barbara, USA*, 2019.

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

[15] M Anwar Hasan, Nicolas Meloni, Ashkan H Namin, and Christophe Negre. Block recombination approach for subquadratic space complexity binary field multiplication based on toeplitz matrix-vector product. *IEEE Transactions on Computers*, 61(2):151–163, 2010.

[16] M Anwar Hasan and Christophe Negre. Multiway splitting method for toeplitz matrix vector product. *IEEE Transactions on Computers*, 62(7):1467–1471, 2012.

[17] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pages 267–288. Springer, 1998.

[18] Andreas Hülsing, Joost Rijneveld, John Schanck, and Peter Schwabe. High-speed key encapsulation from ntru. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 232–252. Springer, 2017.

[19] Matthias J Kannwischer, Joost Rijneveld, and Peter Schwabe. Faster multiplication in $\mathbb{Z}_{2^m}[x]$ on cortex-m4 to speed up nist pqc candidates. In *International Conference*

on Applied Cryptography and Network Security, pages 281–301. Springer, 2019. https://github.com/mupq/polymul-z2mx-m4.

[20] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4. https://github.com/mupq/pqm4.

[21] Matthias J Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking nist pqc on arm cortex-m4. 2019.

[22] Anatolii Karatsuba. Multiplication of multidigit numbers on automata. In *Soviet physics doklady*, volume 7, pages 595–596, 1963.

[23] Angshuman Karmakar, Ingrid Verbauwhede, et al. Saber on arm cca-secure module lattice-based key encapsulation on arm. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 243–266, 2018. https://github.com/KULeuven-COSIC/SABER.

[24] Hugo Krawczyk. Lfsr-based hashing and authentication. In *Annual International Cryptology Conference*, pages 129–139. Springer, 1994.

[25] Hugo Krawczyk. New hash functions for message authentication. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 301–310. Springer, 1995.

[26] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[27] Jose Maria Bermudo Mera, Angshuman Karmakar, and Ingrid Verbauwhede. Time-memory trade-off in toom-cook multiplication: an application to module-lattice based cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 222–244, 2020. https://github.com/KULeuven-COSIC/TCHES2020_SABER/tree/master/Cortex-M4.

[28] Dustin Moody, Gorjan Alagic, Daniel C. Apon, David A. Cooper, Quynh H. Dang, John M. Kelsey, Yi-Kai Liu, Carl A. Miller, Rene C. Peralta, Ray A. Perlner, and et al. *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process.* Jul 2020.

[29] National Instıtute of Standarts and Technology (NIST) Post-Quantum Cryptography Standardization. 2017. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization.

[30] Jeng-Shyang Pan, Chiou-Yng Lee, Anissa Sghaier, Medien Zeghid, and Jiafeng Xie. Novel systolization of subquadratic space complexity multipliers based on toeplitz matrix–vector product approach. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(7):1614–1622, 2019.

[31] Halil Kemal Taşkın and Murat Cenk. Speeding up curve25519 using toeplitz matrix-vector multiplication. In *Proceedings of the Fifth Workshop on Cryptography and Security in Computing Systems*, pages 1–6, 2018.

[32] Andrei L Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. In *Soviet Mathematics Doklady*, volume 3, pages 714–716, 1963.

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