Journal of Theoretical and Applied Information Technology
15th April 2021. Vol.99. No 7

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

NEW FAST MODULAR MULTIPLICATION ALGORITHM APPLIED TO RING-LWE SCHEME

[1]FIRST AUTHOR El HASSANE LAAJI, ,[1]ABDELMALEK AZIZI, [2]TAOUFIK SERRAJ

[1]LACSA Laboratory Mohammed First University, Oujda, Morocco

[2]MASI Laboratory, Multidisciplinary Faculty, Nador, Morocco

E-mail: [1]e.laaji@ump.ac.ma, [1]abdelmalekazizi@yahoo.fr, [2]taoufik.serraj@gmail.com

ABSTRACT

The post-quantum cryptosystem schemes based on Ring-LWE (Ring Learning With Error) problem are among important candidates submitted to the NIST post-quantum cryptosystem standardization project. The majority of them use one or more algorithms for speeding-up the polynomials multiplication in the cryptographic process, like the Number Theoretic Transform algorithm (NTT), the Montgomery algorithm, the Karasuba algorithm, etc. We contribute herein, by creating a new Fast Modular Multiplication algorithm **(FMMA)**, which is a variant of the Montgomery algorithm, and we created an improved post-quantum key exchange release based on Ring-LWE problem called "**RingLWE_FMMA**", by using our FMMA algorithm combined with the NTT algorithm. Our algorithm is tailored specially for the Fermat prime numbers and all the numbers of the form $q = 2^k + 1$. In this work, we considered the post-quantum key exchange "NewHope" based on Ring-LWE problem, as a study case, which uses the NTT algorithm combined with the Montgomery algorithm. . After a benchmarking, we obtained a good result; our RingLWE_FMMA release is faster than NewHope by a rate of up to **35%**, and our FMMA is faster than the Montgomery algorithm by a factor of up to (X2) two times.

Keywords: *Quantum cryptography, Modular Multiplication, Montgomery, Ring-LWE, NTT, KEM.*

1. INTRODUCTION

The recent advances development of quantum computers presents a potential threat to our public-key cryptosystems, which are currently widely deployed; the reader can see the NIST report [1].

Therefore, the innovation in public-key cryptography focuses on the standardization of the post-quantum cryptography primitives and their protocols. The challenge of the cryptographic community right now is to build a Post-Quantum cryptosystem able to resist quantum computer attacks. The classical cryptosystems based on the number-theoretic difficulty such as RSA, ECDH, etc [2], will be easily broken by a quantum computer using quantum algorithms. Now, NIST's post-quantum cryptosystem standardization project reached its third round with different family schemes, the public-key encryption (PKE), the key exchange mechanism schemes(KEM), and the digital signatures schemes(DS). For more details, the reader can see the report [3].

The NIST competition is still in process, and the NIST organization invites the researchers to contribute to this project by improving the submitted post-quantum cryptosystems. On our side, we contribute by improving the post-quantum key exchange "NewHope" proposed by Alkim et.al [4], which is based on Ring-LWE (Ring-Learning With Error) scheme. *"NewHope"* is considered as among important lattice-based cryptosystem and Google had already implemented it in its new browser namely *"Chrome Canary"*.

In 2005, Regev et.al [5] are first introduced the Learning With Errors (LWE) problem, and since its creation, it is used as the foundation for many new lattice-based constructions.

In 2010, Lyubashevsky et.al. [6] introduced Ring-LWE. It is a special instance of the LWE problem that is essentially obtained by adding a ring structure inspired by the NTRU scheme [7]. They showed that an algorithm that solved Ring-LWE in the average case implicated a quantum algorithm that solves the worst cases of SVP. Since its creation, several Ring-LWE-based schemes have been proposed for : Public-Key Encryption, digital signatures, and Key Exchange Mechanism

(KEM) inspired by Diffie&Hellman public key exchange which has been widely used [8][12].

In this work, we contribute by creating an improved post-quantum key exchange release based on the Ring-LWE problem by using our new fast modular multiplication algorithm, which is a variant of the Montgomery algorithm [Montgomery1985]. It is inspired by the recent Ring-LWE key exchange instantiation *"NewHope"*, due to Alkim et.al [4].

The main bottleneck of the Ring-LWE based scheme is the cost of multiplication operations performed over a polynomials ring of the form $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ (with N power of two and q prime number). Therefore, we will use NTT (Number Theorem Transform) [10, 11] combined with our algorithm for speeding up the performance of the polynomials multiplication in the cryptographic process (KeysGeneration, Encapsulation, and Decapsulation).

Our algorithm especially applied to the Fermat prime numbers and all the numbers of the form $q = 2^k + 1$. Its principle is to transform the modulation of a prime number to the number of power of two $\emptyset = q - 1$, and it leads the reduction and the modular multiplication to be fast and more efficient on general-purpose computers, signal processors, and microprocessors. The reduction (%) and multiplication (*) will be respectively replaced by logic operators (&) and (<<) and (>>), as we are going to show in the subsection of the algorithm description.

The remainder of our work is organized as follows: the section.1 contains this introduction; in Section.2, we recall the mathematics background of the Lattices-Based-Cryptography and a brief description of the NTT and Montgomery algorithms; the Section.3 concerns the related works based on the Ring-LWE problem and brief recall of the "NewHope" post-quantum key exchange; in Section.4, we present our contribution by describing our modular multiplication algorithm called ***"FMMA"***, and we describe our Ring-LWE post-quantum key exchange release called ***"RingLWE_FMMA"***; in Section.5, we present the obtained results of the benchmarking between *"FMMA"* and Montgomery algorithms, and the benchmarking between our *"RingLWE_FMMA"* and "NewHope"versions; finally, in Section.6, we give a conclusion on our work and our future research orientation.

2. MATHÉMATIC BACKGOUND

In this section, we will focus to provide only descriptions of the principal subjects that are evoked in this work. So, we give a brief definition of the lattice-based cryptography, a brief description of the NTT algorithm, and the Montgomery algorithm which will be compared to our modular multiplication algorithm in Section.4.

Notation: In this document, we refer to $\boldsymbol{R_q = \mathbb{Z}_q[X]/(X^N + 1)}$ the polynomial ring with the modulus q is prime number and N an integer power of two; we let (a, b, …, z) denote the integers and coefficients of the polynomials; we let (***a, b, …, z***) denote the elements of **Rq** in normal form; and we refer the $(\hat{a}, \hat{b}, \ldots, \hat{z})$ the elements of **Rq** in NTT form.

2.1 Lattice-Based cryptography

The Lattice-Based-Cryptography is defined by Hoffstein et.al in [10] as follow:

The Lattice $L_{(B)}$ *of* $\mathbb{R}^n$ is the set of vectors $\vec{v}$ generated by the basis $B = (\vec{e_1}, \ldots, \vec{e_n})$ with all vector coefficients are integer numbers in $\mathbb{Z}$, formally:

$$L_{(B)} = \{\vec{v} \in \mathbb{R}^n, (a_1, \ldots, a_n) \in \mathbb{Z}^n \text{ and } \vec{v} = a_1\vec{e_1} + \ldots + a_n\vec{e_n}\}.$$

The Lattice Cryptography is based on the complexity to break Lattices Cryptosystems by posing problems that are hard to solve. The principal Lattice problems are SVP and CVP and their definitions are as follow:

2.1.1 The Shortest Vector Problem (SVP)

Finding (SVP) in Lattice $L_{(B)}$ is finding a non-zero vector that minimizes the Euclidean norm. Formally the problem SVP is to find a non-zero vector:

$$\vec{v} \in L_{(B)} \quad \forall \vec{x} \in L_{(B)} \text{ we have } \|\vec{v}\| \leq \|\vec{x}\| \qquad (1)$$

2.1.2 The Closest Vector Problem (CVP)

Given the Lattice $L_{(B)}$, and a vector $\vec{w} \in \mathbb{R}^m$ to find a vector $\vec{v} \in L_{(B)}$ "Closest" to $\vec{w}$, is to find a vector $\vec{v} \in L_{(B)}$ that minimizes the Euclidean norm $\|\vec{w} - \vec{v}\|$ where:

$$\min\{\|\vec{w} - \vec{v}\|\} \rightarrow \text{CVP} = \vec{v}. \qquad (2)$$

Many others approximate problems of SVP and CVP exist, like uSVP, CoreSVP[5], etc. The CoreSVP is the cost of lattice attacks of one call to solve a SVP with block size ***b,*** by using BKZ Lattice reduction algorithm, as we will describe in section 6 [11].

2.2 Number Theoretic Transform (NTT)

The number-theoretic transform (NTT) is a generalization of the Discrete Fourier Transform (DFT), see [6,7], which is carried out in positive Integer group and finite fields whereas the DFT is defined in complex numbers group.

The Number Theoretic Transform (NTT) provides efficient polynomials multiplication in the ring of the form $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ (with N power of two and q prime number). NTT has many applications in computer arithmetic and cryptographic domain, because it reduces the time complexity from $O(n^2)$ *to* $O(n*log(n))$.

To use NTT algorithm we must choosing the modulus that satisfying, $q = kN + 1$, then the multiplicative group $\mathbb{Z}_q^n$ has size $\varphi(q) = q - 1 = k.n$ and a generator g, and computing the primitive nth root of unity Omega ω:

$$\omega = g^k \pmod{q} \text{ and } \omega^n = g^{kn} = g^{\varphi(q)} = 1 \pmod{q}. \quad (3)$$

2.3.1 Transforming a polynomial from Normal form to NTT form

For a polynomial $\boldsymbol{f} = \sum_{i=0}^{n-1} f_i X^i \in R_q$, the NTT function is defined by:

$$NTT(\boldsymbol{f}) = \hat{\boldsymbol{f}};$$

$$\hat{f}_i = \sum_{j=0}^{n-1} \gamma^j f_j \omega^{ij} \pmod{q}. \quad (4)$$

Where Gamma $\gamma = \sqrt{\omega}$ is the 2nd root of unity.

2.3.2 Transforming a polynomial from NTT form to Normal form

The inverse of NTT function to return to normal form is computed by the below formula:

$invNTT\ (\hat{\boldsymbol{f}}) = \boldsymbol{f}$, with :

$$f_i = n^{-1}\gamma^{-i} \sum_{j=0}^{n-1} \hat{f}_j\ \omega^{-ij} \pmod{q}. \quad (5)$$

So the NTT algorithm can perform the multiplication of two polynomials $h = f * g \in R_q$; by transforming them to NTT form ($\hat{\boldsymbol{f}}$ and $\hat{\boldsymbol{g}}$); computing the product in NTT form by the point-wise multiplication noted by ○ (point-wise multiplication) $\hat{\boldsymbol{h}} = \hat{\boldsymbol{f}} \circ \hat{\boldsymbol{g}}$ (that means we obtain $\hat{h}_i = \hat{f}_i * \hat{g}_i \pmod{q}$; and finally transforming the $\hat{\boldsymbol{h}}$ polynomial from NTT form to normal form by the inverse of NTT function: $\boldsymbol{h} = invNTT(\hat{\boldsymbol{h}})$.

Consequently, an important reduction cost of multiplication can be achieved by pre-computing and storing the powers values related to the parameters: ω *and* $\gamma = \sqrt{\omega}$ [6].

2.4 Montgomery Algorithm

In 1985, In His paper titled "Modular Multiplication Without Trial Division"[11] Peter Montgomery introduced an efficient algorithm for reduction and modular multiplication.

The Montgomery algorithm performs divisions by a number of the form 2^k, which is an intrinsically fast operation on general-purpose computers.

The Montgomery algorithm is implemented by "NewHope". It is performed for each multiplication of two coefficients of the polynomials used by the NTT functions in the cryptographic process.

For using the Montgomery algorithm, we choose two co-prime integers S and q, with $S = 2^k > q$ et $\text{pgdc}(S, q) = 1$. The Montgomery reduction of an integer $T \pmod q$ with $0 < T < q < S$, and return $TS^{-1} \pmod q$. So the description of the algorithm is as follow:

Algorithme.1 :

Input: The integers T,S, and q.

1. $m = T(-q^{-1}) \ (mod\ S)$;
2. $t = (T + mq)/S$;
3. $If(q < t)\ return\ t = t - q$
   $else\ return\ t.$

Output: The reduced value of $T : t = TS^{-1} \ (mod\ q)$.

The claim is: $TS^{-1} = (T + T(-q^{-1})(mod\ S)q)/S \ (mod\ q)$.

3. RELATED WORKS

There are many protocols based on Ring-LWE assumption [13], that use NTT algorithm and Montgomery algorithm or others algorithm commonly to speeding-up the polynomials multiplication; in signature scheme functions like [14] and BLISS [15]; in "Public Key Encryption" functions[6]; and in "Key Exchange Mechanism" functions like [4].

In this section, we present an overview of Ring-LWE a brief description of "NewHope", which is our case study, and on which our improvement is based on.

3.1 Ring Learning With Error (Ring-LWE)

In 2005, Oded Regev introduced the Learning With Errors (LWE) problem. He proposed a public-key cryptosystem based on the hardness of the LWE problem, and he showed that to break this system is equivalent to solve the SVP in polynomial

time by using a quantum algorithm [15]. And in 2008, Peikert et.al[19] defined new cryptosystem release based on LWE which considered as dual of Regev's version .

In 2010, Lyubashevsky et.al [6] introduced Ring-LW and inspired from NTRU scheme [7]. They showed that an algorithm that solved Ring-LWE in the average case implicated a quantum algorithm that solves the worst cases of SVP.

The worst-case hardness of Ring-LWE is defined by Peikert's theorem [19] as follow:

Theorem.1: For any $m = poly(n)$, cyclomotic ring **R** of degree n over $\mathbb{Z}$, and appropriate choices of modulus q and an error distribution X of error rate $\alpha < 1$, solving the $RingLWE_{q,x,m}$ problem is at least as hard as quantum solving the SVP_{γ} problem on arbitrary ideal lattices in **R**, for some $\gamma = poly(n)/\alpha$.

3.1.1 Ring-LWE cryptosystem description

The domain of Ring-LWE is the ring of cyclomotic integers $\boldsymbol{R_q = \mathbb{Z}_q[X]/(X^N + 1)}$, with *N* power of two, and q is prime number satisfying the condition $q = 1\ mod\ (2N)$.

Thus, any element $\$\boldsymbol{a} \in \boldsymbol{R_q}$ is a polynomial of degree at most *(N-1)* with integer coefficients, written as $\boldsymbol{a} = \sum_{i=0}^{N-1} a_i X^i$; with $a_i \in \mathbb{Z}\ mod\ q$. The cryptographic processes are :

- ✓ *KeysGeneration*: Generating randomly an element $\boldsymbol{a}$ and private key $\boldsymbol{s}$, and an error $\boldsymbol{e}$ in **R**, the public key is the peer $(\boldsymbol{a}, \boldsymbol{b} = \boldsymbol{a} * \boldsymbol{s} + \boldsymbol{e}) \in \boldsymbol{R_q^2}$;
- ✓ *Encryption*: Encrypting a message $\boldsymbol{m}$ by choosing randomly $\boldsymbol{s_1}, \boldsymbol{e_1}, \boldsymbol{e_2} \in \boldsymbol{R}$ and compute the peer $(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{R_q^2}$ where

$$\boldsymbol{u} = \boldsymbol{a} * \boldsymbol{s_1} + \boldsymbol{e_1}\ (mod\ q)\ and$$
$$\boldsymbol{v} = \boldsymbol{b} * \boldsymbol{s_1} + \boldsymbol{e_2} + (\frac{q}{2})\,\boldsymbol{m}\ (mod\ q).$$

- ✓ *Decryption*: Computing : $\boldsymbol{v - u * s} = (\boldsymbol{r} * \boldsymbol{e} - \boldsymbol{s} * \boldsymbol{e_1} + \boldsymbol{e_2}) + (\frac{q}{2})\,\boldsymbol{m}\ (mod\ q)$, and rounding their coefficients to *0* or *q/2*, finally we decrypt to $m_i = 0$ if the coefficient is 0 else $m_i = 1$.

3.2 Post-quantum Key Exchange "NewHope"

Alkim et.al [4], proposed a lattice-based cryptosystem based on KEM scheme to define the reconciliation mechanism, inspired from Diffie\&Hellman scheme[8]. It is based on Ring-LWE problem, and its domain is the ring of the form $\boldsymbol{R_q = \mathbb{Z}_q[X]/(X^N + 1)}$.

The most critical issue is to choose a as the fixed public parameter because this allows attacks [19]. Therefore in *NewHope*, the polynomial $\boldsymbol{a}$ is not a public parameter, the cryptosystem uses the same ***seed*** to generate uniformly the polynomial $\boldsymbol{a}$ in both keys generation and encapsulation functions. *NewHope* applies the Montgomery algorithm and the improved NTT algorithm, together to increase the speed performance of the cryptographic process.

The authors created two releases with two sequences parameters respectively { n=512,q=12289} and { n=1024,q=12289}. The Ring-LWE secret and errors are sampled according to the centered binomial distribution, with a standard deviation σ=√8.

On term of security NewHope warrants 2^{260} for classical security level and 2^{230} for quantum security level. These values have been obtained by using Albrecht et.al Estimator [17]. For more details, the reader can see the Albrecht et.al [18].

The authors implemented two software versions, the first assumes the security of indistinguishability under the chosen-plaintext attack (IND-CPA), and the second assumes the security of indistinguishability under adaptive chosen ciphertext attack (IND-CCA2).

In this work, we focused only (IND-CPA) secure implementation [4].

4. OUR CONTRIBUTION

In this section, we present and describe our fast modular multiplication algorithm FMMA, and we describe our post-quantum key exchange version based on Ring-LWE problem, called RWE FMMA, defined in the polynomials ring of the form $\boldsymbol{R_q = \mathbb{Z}_q[X]/(X^N + 1)}$with n is power of two, and the modulus q is the fourth Fermat prime number.

The Ring-LWE_FMMA implementation uses the FMMA rather than using the Montgomery algorithm [11], and integrates the new standard SHA-3 (Keccak hash functions) [20].

4.1 Fast Modular Multiplication Algorithm (FMMA)

The motivation for studying high-speed algorithms for modular multiplication comes from their applications in some public-key cryptographies. The FMMA algorithm is efficient when several modular multiplications regarding the

ame modulus q, because the computation requires a lot of normalization stages, and also, it can speed up the modular multiplications and square required during the exponentiation process.

FMMA is constructed specially for Fermat prime numbers and all numbers of the form $q = 2^k + 1$. In our case study, we use the fourth Fermat prime number, see [21], as the modulus for our implementation, as we will describe below.

4.1.1 Algorithm 2: FMMA

FMMA is more suitable for computing modular multiplication by transforming the modulus into a power of two integer in the form $\emptyset = (q-1) = 2^k$. This form allows the computers, signal processors, and microprocessors to replace the reduction, the multiplication, and the division by simple logical operations (&), (<<), and (>>) respectively.

Algorithm1: FMMA

Input: $\boldsymbol{integers\ x,\ y}$; $\boldsymbol{modulus}$ $q = 2^k + 1; and\ \emptyset = (q-1)$.

1. $p \leftarrow x * y;$
2. $z \leftarrow p\ .\&\ (\phi - 1);$
3. $d \leftarrow (p - z) >> log_2(\phi);$
4. $result \leftarrow (z - d);$
5. $if(result < 0)\ then\ return\ result + q.$
6. $else\ return\ result.$

Output: *Reduced number:* $result = x * y \pmod q$.

Comment: The algorithm performs the multiplication of two integers x; y modulus a prime number of the form $q = 2^k + 1$, as follow: (1) computing the product p of x and y; (2) computing an integer z by using the logic operator "&" of p and $(\phi - 1)$; (3) computing an integer d by lifting the difference (p-z) with $k = log_2(\phi)$ bits by the operator (>>); (4) computing the $result = (z - d)$; finally in line.5 the algorithm tests if result < 0 then it return the integer $result + q$; else it returns the reduced value result as in line.6.

4.2 Parameters definition

To use the NTT algorithm combined with FMMA algorithm, we define herein the necessary parameters as follow:

The multiplicative group $\mathbb{Z}^n$ has a generator g and a $\varphi(q) = q - 1 = k.n$, a generator g, and computing the primitive nth root of unity Omega $\omega = g^k \pmod q$ and $\omega^n = g^{kn} = 1 \pmod q$. And defining the nth primitive root of unity ω and its primitive 2n-th root of unity such that $\boldsymbol{\gamma} = \sqrt{\omega}$.

4.2.1 The RingLWE _FMMA parameters values.

Our Ring-LWE implementation herein, defines a sequence parameters $\{N = 1024;\ q = 65537;\ \sigma = \sqrt{8}\}$ with σ is the standard deviation. The modulus used is the fourth Fermat prime number $q = 2^6 * 1024 + 1 = 65537$. Fermat prime numbers were first studied by Pierre Fermat, for more details the reader can see [21].

For computing the needed parameters values of NTT, we should follow the steps below:

- ✓ The parameters $N = 1024;\ k = 64;\ q = 64 * 1024 + 1 = 65537$, which satisfy the condition: $q = kN + 1$;
- ✓ Finding a generator g = 4441 that satisfying the conditions: $g^k \neq 1$ and $g^{kN} = 1$;
- ✓ Computing n-th primitive root of unity: $\omega = g^k\ (mod\ q) = 1089$, and $\gamma = \sqrt{\omega} = 33$; and the inverse of gamma: $\gamma^{-1}(mod\ q) = 1986$, and the inverse of omega: $\omega^{-1}(mod\ q) = 11976$; The particularity of the fourth Fermat prime number is that it has only one root of unit $\omega = 1089$.
- ✓ The inverse of N modulo q: $N^{-1}\ (mod\ q) = 65473$; which will be used by the inverse of NTT function to transform the polynomials from NTT form to normal form.

4.3 RingLWE_FMMA Protocol Description

In this part, we describe the proposed RingLWE_FMMA post-quantum key exchange inspired by the NewHope protocol. And for illustrating our contribution and our improvement, we modified the original algorithms of keys generation, encapsulation, and decapsulation by replacing the Montgomery function by our FMMA function.

All the polynomials are sampled according to Centered Binomial Distribution by the function sampCBD(seed) function except the public parameter a , it is generated according to Uniform distribution by the function sampUF(**seed**). Also, we use SHA3-256 hash function rather than using the SHAKE-256 hash function. For more detail, the reader can see [4], and [20].

4.3.1 Algorithm 3: Keys Generation.

Input : The parameters N, q, and ***seed***.

1. $\boldsymbol{a} \leftarrow sampUF(\boldsymbol{seed})$;
2. $\boldsymbol{s}, \boldsymbol{e} \leftarrow sampCBD(\boldsymbol{seed})$;
3. $\hat{\boldsymbol{s}}, \hat{\boldsymbol{e}} \leftarrow NTT(\boldsymbol{s}, \boldsymbol{e})$;
4. $\hat{\boldsymbol{b}} \leftarrow \hat{\boldsymbol{a}} \circ \hat{\boldsymbol{s}} + \hat{\boldsymbol{e}}$;
5. $\boldsymbol{pk} \leftarrow encode(\hat{\boldsymbol{b}} + \boldsymbol{seed})$;
6. $\boldsymbol{sk} \leftarrow encode(\hat{\boldsymbol{s}})$

Output: the public key ***pk*** in NTT form and the private key ***sk*** in NTT form.

4.3.2 Algorithm 4: Encapsulation.

Input: The public key ***pk, seed***.

1. Choosing randomly a message (shared key): ***msg*** and transforming it to polynomial ***m*** ;
2. $\left(\hat{\boldsymbol{b}}, \boldsymbol{seed}\right) \leftarrow decode(\boldsymbol{pk})$;
3. $\boldsymbol{m} \leftarrow codify(\boldsymbol{m})$; with $m_i \in \left\{0, \frac{q-1}{2}\right\}$;
4. $\boldsymbol{a} \leftarrow sampUF(\boldsymbol{seed})$;
5. $\boldsymbol{s_1}, \boldsymbol{e_1}, \boldsymbol{e_2} \leftarrow sampleCBD(seed)$;
6. $\hat{\boldsymbol{s}}_1, \hat{\boldsymbol{e}}_1 \leftarrow NTT(\boldsymbol{s_1}, \boldsymbol{e_1})$;
7. $\hat{\boldsymbol{u}} \leftarrow \hat{\boldsymbol{a}} \circ \hat{\boldsymbol{s}}_1 + \hat{\boldsymbol{e}}_1$;
8. $\boldsymbol{v} \leftarrow invNTT(\hat{\boldsymbol{b}} \circ \hat{\boldsymbol{s}}) + \boldsymbol{e} + \boldsymbol{m}$
9. $\boldsymbol{SSc} \leftarrow SHA3-256(\boldsymbol{msg})$;(***SSc*** is the shared key).
10. $\boldsymbol{C} \leftarrow encode(\hat{\boldsymbol{u}} + v)$

Output: The ciphertext **C**.

4.3.3 Algorithm 5: Decapsulation.

Input: The private key sk and the ciphertext C.

1. $(\hat{s}) \leftarrow decode(sk)$;
2. $\hat{\boldsymbol{u}} + v \leftarrow decode(\boldsymbol{C})$;
3. $\boldsymbol{U} \leftarrow invNTT(\hat{\boldsymbol{u}} \circ \hat{\boldsymbol{s}})\ (\boldsymbol{mod}\ \boldsymbol{q})$;
4. $\boldsymbol{m} = \boldsymbol{U} - \boldsymbol{V}\ (mod\ q)$; with $m_i \in \{0, 1\}$;
5. $\boldsymbol{msg} \leftarrow encode(\boldsymbol{m})$;
6. $\boldsymbol{SSd} \leftarrow SHA3-256(\boldsymbol{msg})$;

Output: The shared key ***SSd.***

Comment: The KeysGeneration, Encapsulation, and Decapsulation algorithms describe the protocol operations and illustrate the use of the NTT algorithm and the SHA3-256 hash function. The FMMA modular multiplication function is used in NTT functions and for each coefficients multiplication in all the cryptographic process.

The reconciliation is good, if only if the shared key ***SSc*** (hashed string) in the encapsulation algorithm is equal to shared key ***SSd*** (hashed string) in the decapsulation algorithm, else the complete process must be repeated until the reconciliation is exact. For more details, the reader can see the [4] [19] works. The decryption failure rate of our implementation is about 2^{-216} obtained by using the python script developed by Alkim et.al [22], and executing it in Sage software.

On term of security our release achieves 2^{260} for classical security level and 2^{230} for quantum security level. These values have been obtained by using the Albrecht et.al Estimator [17].

5. RESULT ANALYSIS

We note that all implementations are performed in the platform PCTOSHIBA, Satellite, Processor Intel , Core i7-2630QM CPU, 2GHz, RAM 8GO, under environment Windows 7-32 bits and Dev-C++ 4.9.9.2.

The implementation of our RingLWE_FMMA release described in this paper is available on the Google drive website at [25] and the NewHope implementation is available in the NIST website at [26].

5.1 Result Analysis of FMMA and Montgomery

In this subsection, we present the implementation of FMMA and a performance benchmarking of our algorithm compared to the Montgomery algorithm, by computing the exponentiation of some numbers by a series of square and multiplication operations with modulus $q = 2^{16} + 1 = 65537$.

5.1.1 FMMA algorithm Implementation:

With modulus $q = 2^{16} + 1 = 65537$, $\emptyset = (q-1) = 2^{16}$, $(\emptyset - 1) = 0xFFFF$, and $log2(\emptyset) = 16$:

Listing.1: FMMA function

```
long FMMA(long x, long y, long q) {
1. long p = x * y;
2. long z = p & (∅ − 1);
3. long d = (p − z) >> log2(∅);
4. long result = (z − d);
5. if (result < 0)return result = result + q;
6. else return result; }
```

5.1.2 Montgomery algorithm implementation

The Montgomery algorithm (Algorithm.1) goal is to reduce fastly an integer $T\,(mod\,q)$, $with\,q = 65537$. For implementing it, we choose $S = 2^{18} = 262144$, and we compute $S\,(mod\,q) = 65533$, $S^2\,(mod\,q) = 16$ and $qinv = \frac{1}{q}\,(mod\,S) = 65535$:

Listing.2 : Montgomery Reduction function:

```
long Montgomery(long T, long q) {
1. long m = 0;
2. long t = 0;
3. m = (T * qinv) & (S − 1);
4. t = m * q + T;
5. t = t = S;
6. if (t < q) return t;
7. else return t − q; }
```

As claims by Montgomery, this function returns $t = T * S^{-1}\,(mod\,q)$.

For the Montgomery modular multiplication, the NewHope authors implemented it as follow:

Listing.3 : Montgomery Modular Multiplication: $Z = X * Y\,(mod\,q)$.

```
long ModularMult(long X, long Y, long q) {
1. long t = 0;
2. long Z = 0;
3. t = Montgomery(16 * Y; q);
4. Z = Montgomery(X * t; q);
5. return Z; }
```

The modular multiplication function of the Montgomery calls the Montgomery reduction function two times. First, in line.3, the Montgomery reduction function reduces $S^2 * Y$ to $t = S * Y\,(mod\,q)$; and in line.4, the Montgomery function reduces $X * t\ to\ Z = X * Y\,(mod\,q)$. For more details of the Montgomery implementation in NewHope software see [4][26].

5.2 The result performance of FMMA Compared to Montgomery

The best tool for testing the speed performance of both modular multiplication algorithms is the computation of the integers exponentiation. We give an example by using the value parameters cited above. We compare the performance of both algorithms by executing some exponentiation of a number $\boldsymbol{a^b}\ (\boldsymbol{mod\ q})$ with $\boldsymbol{a} = 63215 < q$, and b takes different sizes (6 digits, 7 digit, 8 digits) of respective values (**b1**=958767,**b2**=9587679,**b3**=95876795). We note that each computation of the exponentiation is replicates 100 times.

In the table.1 below, the average result showed that our algorithm is faster than the Montgomery algorithm by a factor up to 2 times.

Table.1: Speed performance FMMA versus Montgomery (Milliseconds)

ALGORITHMS	a^{b1}	a^{b2}	a^{b3}
MONTGOMERY	36 ms	360 ms	3580 ms
FMMA	17, 5 ms	166 ms	1650 ms
SPEED-UP FACTOR	2 times	2,17 times	2,17 times

5.3 Benchmarking between RingLWE_FMMA and Newhope

In this section, we present benchmarking results of our RingLWE_FMMA implementation of parameters $\{N = 1024;\ q = 65537;\ \sigma = \sqrt{8}\}$ compared to the NewHope version with adequate parameters $\{N = 1024;\ q = 12289;\ \sigma = \sqrt{8}\}$.

We used sampCBD() function to generate the polynomials according to Centered Binomial Distribution inspired from NewHope implementation, with some change for generating two polynomials in the same time, and we use SHA3-256 Keccak hash function rather than using SHAKE-256 , that allows us to hash the shared key directly in binary polynomial form without encoding it into string format as you showed in the encapsulation and the decapsulation algorithm. The reader can see these functions in our RLWE FMMA release implementation at link [25].

We built the software of both implementations on the same machine cited above, and we report the median result of 100 runs. In the table.2, we converted the reported results to approximate cycle counts as given below:

Table 2 Speed performance benchmarking between RingLWE FMMA and NewHope (values in Millions of cycles)

Schemes	KeysGen	Encapsu	Decapsu
NewHope	1. 60	2.40	0.60
RLWE_FMMA	1.20	2.00	0.40
Gains	**0.40**	**0.40**	**0.20**

In this result, we remark that we did better by increasing the speed performance by a factor up-

to $X1.34$ for key generation function with a gain of $4 * 10^5 cycles$; by a factor up-to $X1.2$ for encapsulation function with a gain $4 * 10^5 cycles$, and by a factor up-to $X1.5$ for encapsulation function with a gain of $2 * 10^5 cycles$.

That means our version outperforms the NewHope version by a rate value of up to **35%**. This gap performance, between our release and NewHope release, is essentially due to the performance of our modular multiplication algorithm FMMA compared to the Montgomery algorithm.

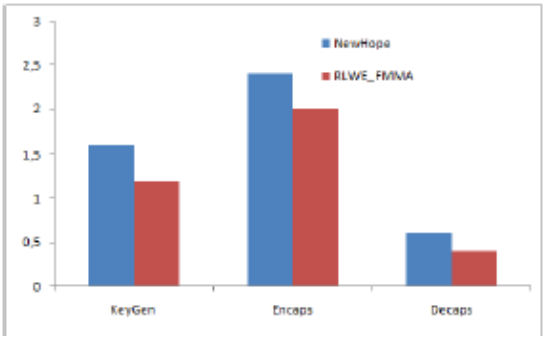

Figure.1: Speed performance benchmarking between RLWE FMMA and NewHope (values in Millions of cycles)

5.4 Security

Many cryptanalysis works are performed; their principal goal was to check the robustness of the Lattices-Based Cryptography, by posing the hardest problems on point lattices in $\mathbb{R}^N$. The best tools used to prove the security is Lattice reduction by the algorithms (Gram-Schmidt, LLL, BKZ algorithms) and Meet-in-The-Middle attack (MIM) [23].

In fact, the mathematicians often estimate the projected security of cryptographic systems by plotting the evolution in "running time" and "space requirements" of the best-known attacks according to level security needed.

For measuring the security level of Lattice-Based Cryptosystems, [17] developed an estimator as described in their paper titled"Estimate all the LWE, NTRU schemes". This tool helps the researchers in this area to check the security level of their cryptosystems.

The result for Ring-LWE assumption provides 2^{260} security level for the primal attack of a unique short vector problem (uSVP) which searches for a short vector in the lattice and achieves 2^{296} security level for the dual attack, which searches for a short vector in the dual lattice, the reader can see the Peikert's paper titled "Decade Lattice-Based Cryptography" [19].

6 CONCLUSION

The cryptography is present in our daily life. Actually it is used in social media software, in cloud computing, and in the Internet of Things (IoT) [27]. It encompasses devices, sensors, people, data, and machines and the interactions between them. But are we secure when the quantum computer will be scalable? Our role and goal are to build a strong and robust cryptosystem able to secure our private life.

So on our side; we contribute by creating a new post-quantum key exchange release based on the Ring-LWE assumption. We obtained significant improvements in performance by using our Fast Modular Multiplication Algorithm (FMMA) with an improved NTT algorithm. Our improvements should be of independent interest and might be applicable to other protocols. We conclude that our methods increase the speed of the cryptographic process without sacrificing the performance in the term of security, and warrant a perfect correctness of the decryption.

The limitation of our method is that it is applied only for the cryptographic schemes defined in the ring of the form $\boldsymbol{R_q = \mathbb{Z}_q[X]/(X^N + 1)}$ with modulus q a prime number of the form $2^k + 1$ and a dimension of lattice N is power of two.

. For future works, we hope to apply this method to other post-quantum cryptosystem based on R-LWE cryptographic schemes and NTRU cryptographic schemes.

REFRENCES:

[1] G.Chen, S.Jordan, D. Moody, L.Yi-Kai, R. Peralta, R. Perlner and D.Smith. D. NISTIR 8105 -Report on Post-Quantum Cryptography. Gaithersburg, Washington, USA ,2016.

[2] E.Sahinaslana, O.Sahinaslana. Cryptographic methods and development stages used throughout history AIP Conference Proceedings 2086, 030033 (2019); https://doi.org/10.1063/1.5095118 Maltepe University, Istanbul, Turkey, 2019.

[3] G.Alagic, J. Alperin-Sheriff, D.Apon, D.Cooper, Q.Dang, L.Yi-Kai, C. Miller, D. Moody, R.Peralta, R.Perlner, A.Robinson, and D.Smith. Status Report NISTIR 8240 on the First Round of the NIST Post-Quantum Cryptography

Standardization Process. Gaithersburg, Washington USA, 2019.

[4] Alkim, E. Ducas, . Poppelman, T. and Schwabe, P. Post-quantum key exchange- ”New Hope”. Department of Mathematics, Ege University, USA, 2019.

[5] Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pages 84–93, Baltimore, Mryland USA, 2005. Ring-LWE post-quantum cryptoysystem 19

[6] V. Lyubashevsky, C. Peikert, and O.Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, Advances in Cryptology - EUROCRYPT2010, volume 6110 of Lecture Notes in Computer Science, pages 1-23. Springer, 2010.

[7] J.Hoffstein, J.Pipher, and J.H.Silverman. Introduction Mathematics and Cryptography NTRU. Wilmington USA, 1998.

[8] J.Ding, X. Xie, and X.Lin. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688. http://eprint.iacr.org/2012/688, 2012.

[9] Longa, P. and Naehrig, M. Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography. Microsoft Research USA, 2019.

[10] C.Chen, D.Danba, J. Hoffstein, A. Hulsing, J. Rijneveld, J.M. Schanck, P. Schwabe, W.Whyte, and Z.Zhang. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. Wilmington USA, 2019.

[11] P.L. Montgomery. Modular Multiplication Without Trial Division. Math comput. USA, 1985.

[12] M.Hartmann. Ajtai-Dwork Cryptosystem and Other Cryptosystems Based on Lattices. University de Zurich, 2015.

[13] D.Chen, N. Mentens, S.Roy, F.Vercauteren, and I.Verbauwhede. Compact Ring-LWE cryptoprocessor. In Lejla Batina and Matthew Robshaw, editors, ”Cryptographic Hardware and Embedded Systems” - CHES 2014, volume 8731 of Lecture Notes in Computer Science, pages 371-391. Springer, 2014.

[14] V.Lyubashevsky. Lattice signatures without trapdoors. In D. Pointcheval and T. Johansson, editors, Advances in Cryptology - EUROCRYPT 2012, volume 7237 of Lecture Notes in Computer Science, pages 738-55. Springer, 2012.

[15] L.Ducas, A.Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal Gaussians. In R. Canetti and J. A. Garay, editors, Advances in Cryptology – CRYPTO 2013, volume 8042 of Lecture Notes in Computer Science, pages 40-56. Springer, 2013.

[16] H.M.Anwar, A.R. Crystal, and A.T.Chi-En. Energy Consumption of Round 2 Submissions for NIST PQC Standards. University of Waterloo. Canada? 2019.

[17] M.R.Albrecht, B. Benjamin, and B.Curtis. Estimate all the fLWE, NTRU schemes. USA, 2018.

[18] M.RAlbrecht, R. Player , and S.Scott. On the concrete hardness of Learning with Errors. Journal of Mathematical Cryptology. Volume 9, Issue 3, Pages 169–203, ISSN (Online) 1862-2984, ISSN (Print) 1862-2976 DOI: 10.1515/jmc-2015-0016, Germany 2015.

[19] C.Peikert. A Decade of Lattice Cryptography. University of Michigan, USA, 2016.

[20] G.V. Assche, G. Bertoni, J. Daemen, P.Peters, and R.Van. Keccak Hash algorithm. Radboud University,Nederlands, 2016.

[21] P.Ribenboim. The New Book of Prime Number Records. New York, Springer-Verlag, 1996.

[22] E.Alkim, L. Ducas, T.Poppelman, and P.Schwabe. The scripts used for mesuring the the security level and decryption failure rate https://github.com/newhopecrypto/newhope/tree /master/scripts. 2019.

[23] J.Hoffstein, J.Pipher, J.M. Schanck, J.H.Silverman, W. Whyte, and Z. Zhang. Choosing Parameters for NTRUEncrypt. Wilmington USA 2016.

[24] P. Falzon. Les dialogues de diagnostic: L’′evaluation des connaissances de l’interlocuteur. Technical Report 747, INRIA, Rocquencourt, France, 1987.

[25] E.LAAJI, A.AZIZI, Implementation of RLWE FMMA post-quantum key exchange. https : ==drive:google:com=open?id = 1Y Y dPA2syC6fIvCOoiV IZHHRlsEClwT,

[26] E.Alkim, L. Ducas, T.Poppelman, and P.Schwabe. The NewHope post-quantum key exchange implementation. https : ==drive:google:com=open?id = 1Y Y dPA2syC6fIvCOoiV IZHHRlsEClwT.

[27] E.Sahinaslana, O.Sahinaslana. Cross-object information security: A study on new generation encryption AIP Conference Proceedings 2086, 030034 (2019); https://doi.org/10.1063/1.5095119 Maltepe University, Istanbul, Turkey, 2019.