ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.11, No.22, 1–6

Efficient interleaved Montgomery modular multiplication for lattice-based cryptography

Sedat Akleylek[1,2a)] **and Zaliha Yüce Tok**[2]

[1] *Department of Computer Engineering, Ondokuz Mayıs University, 55139, Samsun, Turkey*

[2] *Institute of Applied Mathematics, Middle East Technical University, 06531, Ankara, Turkey*

a) *sedat.akleylek@bil.omu.edu.tr*

Abstract: In this paper, we give modified version of interleaved Montgomery modular multiplication method for lattice-based cryptography. With the proposed algorithms, we improve the multiplication complexity and embed the conversion operation into the algorithm with almost free cost. We implement the proposed methods for the quotient ring $(\mathbb{Z}/q\mathbb{Z})[x]/(x^n - 1)$ and $(\mathbb{Z}/p\mathbb{Z})[x]/(x^n + 1)$ on the GPU (NVIDIA Quadro 600) using the CUDA platform. NTRUEncrypt is accelerated approximately 35% on the GPU by using the proposed method. We receive at least 19% improvement with the proposed method for the polynomial multiplication in $(\mathbb{Z}/p\mathbb{Z})[x]/(x^n + 1)$, where $n \in \{1024, 2048, 4096\}$.

Keywords: interleaved Montgomery modular multiplication, lattice-based cryptography, NTRUEncrypt, GPU implementation

Classification: Electron devices, circuits, and systems

References

[1] D. Bernstein: Multidigit multiplication for mathematicians (2001) http://cr.yp.to/papers/m3.pdf.

[2] S. Akleylek, M. Cenk and F. Özbudak: IET Information Security **7** (2013) 165. DOI:10.1049/iet-ifs.2010.0271

[3] J. Hoffstein and J. Silverman: Algorithmic Number Theory Symposium-III (1998) LNCS 1423 267.

[4] C. O'Rourke and B. Sunar: IEEE Trans. Comput. **52** (2003) 440. DOI:10.1109/TC.2003.1190585

[5] V. Lyubashevsky, C. Peikert and O. Regev: Advances in Cryptology EUROCRYPT (2010) LNCS 6110 1.

[6] V. Lyubashevsky, D. Micciancio, C. Peikert and A. Rosen: Fast Software Encryption (FSE) (2008) LNCS 5086 54.

[7] IEEEp1363.1: Draft Standard (2008).

[8] M.-K. Lee, J. W. Kim, J. E. Song and K. Park: IEICE Trans. Fundamentals **E96-A** (2013) 206. DOI:10.1587/transfun.E96.A.206

[9] S. Akleylek and Z. Yüce Tok: IEEE 22nd Signal Processing and Communications Applications Conference (2014) 854.

DOI: 10.1587/elex.11.20140960
Received October 7, 2014
Accepted October 15, 2014
Publicized October 30, 2014
Copyedited November 25, 2014

1 Introduction

In modern public key cryptographic schemes such as ECDSA modular multiplication operation is the most time consuming part. There are several algorithms to obtain efficient results for multiplication operation: Karatsuba-Ofman method, Toom-Cook method, FFT-based techniques, Montgomery method [1]. In this paper we focus on Montgomery modular multiplication method and its efficient adaptation to lattice-based cryptographic schemes. To do this, we eliminate one multiplication by using the quotient ring properties. We extend the idea given in [2]. We also improve the complexity results in [2].

Post-quantum cryptographic schemes have received much more attention after introducing polynomial time quantum algorithms to solve the hard problems for some parameters (e.g. integer factorization problem, discrete logarithm problem) which most of the public key algorithms depends on. Lattice-based cryptographic schemes are the most studied ones since the operations can be considered over the quotient ring enabling very efficient modular reduction. NTRU cryptosystem is the first proposed scheme as an alternative to RSA and elliptic curve based systems in the lattice-based cryptography [3]. In NTRU the main operation is polynomial multiplication in $(\mathbb{Z}/q\mathbb{Z})[x]/(x^n - 1)$. This multiplication can also be considered as cyclic convolution of two polynomials.

1.1 Our contribution

We give modified versions of interleaved Montgomery modular multiplication algorithms for NTRUEncrypt (the quotient ring $(\mathbb{Z}/q\mathbb{Z})[x]/(x^n - 1)$) and the quotient ring $(\mathbb{Z}/p\mathbb{Z})[x]/(x^n + 1)$. With the proposed methods we improve the multiplication complexity. We implement the modified algorithms on the GPU by using CUDA platform. We also compare the proposed algorithms with the previous ones. In original Montgomery modular multiplication algorithm computes the product in Montgomery form i.e. one needs one more multiplication to obtain the real result. In the proposed method we make this operation in a clear way and we eliminate the final subtraction as in [2]. The proposed algorithms can also be considered as a generalized version of [4]. By using the proposed methods, polynomial multiplication over the quotient ring is accelerated at least 19% on the GPU.

2 Proposed methods

In this section we explain the proposed methods for $(\mathbb{Z}/q\mathbb{Z})[x]/(x^n - 1)$ and $(\mathbb{Z}/p\mathbb{Z})[x]/(x^n + 1)$, where q is a power of 2 and p is an odd prime. Let $R_q = (\mathbb{Z}/q\mathbb{Z})[x]/(x^n - 1)$, $R_p = (\mathbb{Z}/p\mathbb{Z})[x]/(x^n + 1)$, $\mathbb{Z}_q = (\mathbb{Z}/q\mathbb{Z})$ and $\mathbb{Z}_p = (\mathbb{Z}/p\mathbb{Z})$.

In Montgomery modular multiplication method one needs to transform the elements to the required form. For studied quotient ring case that is for given $a(x)$, $b(x) \in R_q$, first compute $a(x) \cdot b(x)$ in the form enabling very efficient computation and then transform the result to final computation [2].

In NTRU, polynomial arithmetic is performed in R_q i.e., $x^n = 1$. In Montgomery modular multiplication algorithm one needs $M'(x) \equiv -M(x)^{-1} \pmod{x^w}$, where w is the word length of the target platform. In Lemma 1 we give the computation of

DOI: 10.1587/elex.11.20140960
Received October 7, 2014
Accepted October 15, 2014
Publicized October 30, 2014
Copyedited November 25, 2014

$M'(x)$ for the NTRU case. Note that this also helps us to eliminate the precomputation phase.

Lemma 1. *Let* $M(x) = x^n - 1$ *and* $M'(x) \equiv -M(x)^{-1} \pmod{x^w}$, *where* $w \leq n$. *Then,* $M'(x) = 1$.

In Algorithm 1 we give modified interleaved Montgomery modular multiplication algorithm for NTRU. After using the observation in Lemma 1, we decrease the required number of multiplications by one with omitting the multiplication $M'(x)$ (see Step 4). Then, we replace the multiplication with $M(x) = x^n - 1$ by shifting n times and one subtraction (see Step 5). Recall that shifting operation is almost free. We convert the multiplication with $M(x)$ to shifting and subtraction operations which improves the complexity of the algorithm. Since we are working on the Montgomery form, we need to convert the elements to the desired form. Conversion of the result is done by shifting operation (see Step 8 and 9). In Algorithm 1, the required number of multiplications is reduced to 1 (see Step 3) and the required number of additions is 3 (see Step 3 and 5).

Algorithm 1 Interleaved Montgomery Modular Multiplication Algorithm for NTRU

Input: $A(x) = \sum_{i=0}^{n-1} a_i x^{iw}$, $B(x) = \sum_{i=0}^{n-1} b_i x^{iw}$, $M(x) = x^n - 1$, with $a_i, b_i \in \mathbb{Z}_q$, where q is a prime power, $deg(A(x)) < deg(M(x))$, $deg(B(x)) < deg(M(x))$, $gcd(r(x), M(x)) = 1$, $r(x) = x^w$ and $n_w = \lceil \frac{n}{w} \rceil$.

Output: $C(x) = A(x) \cdot B(x) \pmod{M(x)}$

1: $C(x) \leftarrow 0$
2: **for** $i = 0$ to $n_w - 1$ **do**
3: $\quad C(x) \leftarrow C(x) + A(x) \cdot b_i(x)$
4: $\quad q(x) \leftarrow C(x) \pmod{r(x)}$
5: $\quad C(x) \leftarrow (C(x) + q(x) \cdot x^n - q(x))/r(x)$
6: **end for**
7: $T(x) \leftarrow (r(x))^{n_w}$
8: $C(x) \leftarrow (C(X) \cdot T(X)) \pmod{M(x)}$
9: Return $C(X)$

The ring variant of learning with errors problem (R-LWE) have been mostly used in new generation public key cryptosystems [5] and hash functions [6]. Ideal lattices with special properties are needed to construct R-LWE based schemes. These lattices can be considered as the ideals in $R_p = (\mathbb{Z}/p\mathbb{Z})[x]/(x^n + 1)$. The parameters are $n = 2^k$ and $p \equiv 1 \pmod{2n}$, where k is positive integer and p is prime. In Lemma 2 we give the computation of $M'(x)$ for R_p case. By using this, one multiplication in interleaved Montgomery method turns out to the subtraction operation.

Lemma 2. *Let* $M(x) = x^n + 1$ *and* $M'(x) \equiv -M(x)^{-1} \pmod{x^w}$, *where* $w \leq n$. *Then,* $M'(x) = -1$.

Now we have $M'(x) = -1$. In Step 4 we replace the multiplication by $M'(x)$ with multiplication by (-1). Note that this can be also considered as an addition

DOI: 10.1587/elex.11.20140960
Received October 7, 2014
Accepted October 15, 2014
Publicized October 30, 2014
Copyedited November 25, 2014

modulo p. Since in R-LWE based schemes coefficients of the elements are chosen from the set $\{-1, 0, 1\}$, this multiplication by (-1) does not effect the efficiency of the algorithm. In Step 5 instead of multiplication with $M(x) = x^n + 1$ we use shifting the corresponding polynomial n times and then add it. With this observation we decrease the number of multiplication in the algorithm. In Step 8 and 9 we convert the elements to the desired form. Note that these are not the real multiplications, they are just shifting operations. The multiplication and addition complexity of Algorithm 2 is only 1 (see Step 3) and 3 (see Step 3 and 5), respectively.

Algorithm 2 Interleaved Montgomery Modular Multiplication Algorithm in R_p

Input: $A(x) = \sum_{i=0}^{n-1} a_i x^{iw}$, $B(x) = \sum_{i=0}^{n-1} b_i x^{iw}$, $M(x) = x^n + 1$, with $a_i, b_i \in \mathbb{Z}_q$, where q is a prime power, $deg(A(x)) < deg(M(x))$, $deg(B(x)) < deg(M(x))$, $gcd(r(x), M(x)) = 1$, $r(x) = x^w$ and $n_w = \lceil \frac{n}{w} \rceil$.

Output: $C(x) = A(x) \cdot B(x) \pmod{M(x)}$

1: $C(x) \leftarrow 0$
2: **for** $i = 0$ to $n_w - 1$ **do**
3: $\quad C(x) \leftarrow C(x) + A(x) \cdot b_i(x)$
4: $\quad q(x) \leftarrow -C(x) \pmod{r(x)}$
5: $\quad C(x) \leftarrow (C(x) + q(x) \cdot x^n + q(x))/r(x)$
6: **end for**
7: $T(x) \leftarrow (r(x))^{n_w}$
8: $C(x) \leftarrow (C(X) \cdot T(X)) \pmod{M(x)}$
9: Return $C(X)$

3 Experimental results

In this section we give the implementation details both for Algorithm 1 and Algorithm 2 on the GPU using CUDA platform. We use NVIDIA Quadro 600 GPU having 96 CUDA cores. To show the effectiveness of the proposed methods, we compare them with the interleaved Montgomery modular multiplication method.

In Fig. 1 polynomial multiplication algorithms are compared in view of the number of parallel multiplications per second on $R_q = (\mathbb{Z}/q\mathbb{Z})[x]/(x^n - 1)$ by using the parameter sets given in [7]. To perform the polynomial multiplication the required random data is generated on the GPU with CUDA platform. Since transferring data between CPU and GPU needs more time, we prefer this choice. According to the implementation results, polynomial multiplication in R_q is accelerated almost 29% by using Algorithm 1. Our design for ees401ep1 parameter set achieves the throughput of 12156 polynomial multiplications per second while it's 9391 in the original one. Note that degree of the polynomial has an important affect on the performance of parallel multiplication.

Table I summarizes our experimental findings for NTRUEncrypt from $n = 401$ up to $n = 853$. In Table I ees*n*ep1 means that $n \in \{401, 449, 653, 853\}$ and $q = 2048$. The timings for encryption operation are given for 1000 trials and the input of NTRUEncrypt is randomly generated. The data is generated on the CPU and then it's transferred to the GPU. While implementing NTRUEncrypt, we use a

DOI: 10.1587/elex.11.20140960
Received October 7, 2014
Accepted October 15, 2014
Publicized October 30, 2014
Copyedited November 25, 2014

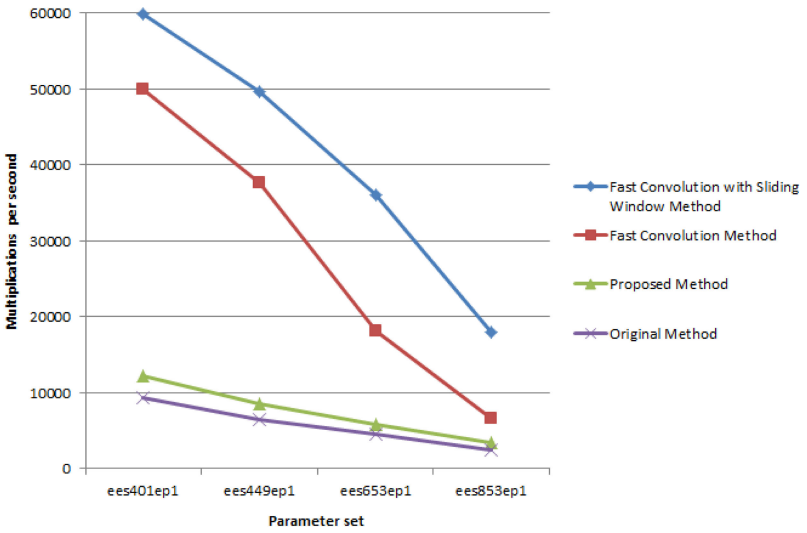

Fig. 1. The number of parallel multiplications per second

set of parameters for different security level recommended in [7]. We implement fast convolution and its sliding window version as described in [8]. We also compare the proposed method with the original Montgomery modular multiplication method. According to the experimental results by using modified interleaved Montgomery multiplication method NTRUEncrypt is accelerated almost 35%. However, the proposed method is not the best choice for NTRUEncrypt. Fast convolution with sliding window method gives better performance since multiplication is performed by only additions and the required number of additions is drastically reduced when we compare this with the fast convolution method. Moreover, fast convolution method and its sliding window version have a nice structure for parallelization.

Table I. Experimental results for NTRUEncrypt on the GPU using CUDA platform (second/parameter set)

Parameter Set	ees401ep1	ees449ep1	ees653ep1	ees853ep1
Fast Convolution with Sliding Window Method [8]	0.179	0.233	0.350	0.749
Fast Convolution Method [8]	0.265	0.384	0.621	1.678
Proposed Method (Algorithm 1)	0.829	1.401	1.986	3.037
Original Method	1.142	1.768	2.502	3.755

In Table II the number of multiplications over the polynomial ring $R_p = (\mathbb{Z}/p\mathbb{Z})[x]/(x^n + 1)$ for selected methods is given. We choose $p = 49201153$ satisfying $p \equiv 1 \pmod{2n}$. We implement parallelized schoolbook method, CUDA Fast Fourier Transform (cuFFT) based multiplication (one can also call this Number Theoretic Transform) [9] and interleaved Montgomery modular multiplication method. We generate the random data on the GPU. Since cuFFT is optimized version of FFT on the GPU for the parallel processing, cuFFT-based multiplication gives the best throughput. According to the experimental results, Algorithm 2 gives better performance than the original one. We also note that the

DOI: 10.1587/elex.11.20140960
Received October 7, 2014
Accepted October 15, 2014
Publicized October 30, 2014
Copyedited November 25, 2014

number of parallel multiplication decreases when one compares with Table I. The reason is that taking modulo with $p = 49201153$ results a delay. The comparison results show that by using Algorithm 2 multiplying two elements in R_p is accelerated at least 19% compared to interleaved Montgomery modular multiplication.

Table II. Experimental results for selected polynomial multiplication methods over the polynomial ring R_p on the GPU using CUDA platform (second/n)

	$n = 1024$	$n = 2048$	$n = 4096$
Parallelized Schoolbook Method [9]	9478	5897	2515
cuFFT-based Multiplication [9]	27650	15308	7691
Proposed Method (Algorithm 2)	9659	6034	2671
Original Method	8074	4965	2153

4 Conclusion

In this paper, we give the required updates for interleaved Montgomery modular multiplication method to be used in lattice-based cryptographic schemes. The major improvement is to reduce the required number of multiplications. Algorithm 1 and Algorithm 2 give better performance than the original one. We give an acceleration of interleaved Montgomery modular multiplication at least 19% on the GPU for lattice-based cryptography.

Acknowledgments

Sedat Akleylek is partially supported by TÜBITAK under 2219-Postdoctoral Research Program Grant. He perfomed this study while he was a postdoctoral researcher at Cryptography and Computer Algebra Group, TU Darmstadt.

DOI: 10.1587/elex.11.20140960
Received October 7, 2014
Accepted October 15, 2014
Publicized October 30, 2014
Copyedited November 25, 2014