UDC 004.056.55

IMPLEMENTING NTRU-similar ALGORITHM ON THE BASIS OF NTRUPrime

Ivan Gorbenko[1], Olena Kachko[2], Gleb Naumenko[3]

[1] V.N. Karazin Kharkiv National University, Svobody sq., 4, Kharkov, 61022, Ukraine
gorbenkoi@iit.kharkov.ua

[2,3] Kharkiv National University of Radioelectronic, Nauka Ave, 14 Kharkov, 61166 Ukraine
kachko@iit.com.ua, naumenko.gs@gmail.com

Reviewer: Roman Oliynikov, Dr., Full Professor, V.N. Karazin Kharkiv National University, Svobody sq., 4, Kharkov, 61022, Ukraine
roliynykov@gmail.com

Received on December 2016

Abstract. *The modern attacks uses special structures of the rings in the NTRU similar algorithms. The article was proposed post-quantum parameters NTRUPrime without these structures. Have investigated the possibility of using these parameters for encryption on the part of the most important characteristic which distinguish NTRU methods from the rest of algorithms, namely, speed regulation characteristics. In fact, standard ANSI X9.98 – 2010 is used for this, but with NTRUPrime mathematics. Use AVX2 commands for multiplication of polynomials and effective implementation of necessary operations yielded minimal reduction in performance*

Keywords: *ANSI X9.98 – 2010, NTRUPrime, Encription Speed, Decryption Speed.*

1 Introduction

In connection with problems concerned with scope for using modern asymmetric algorithms under the conditions of the availability of quantum computers, cryptographic algorithms, the reliability of which is based on lattices [1], are of the greatest interest. It is this class that NTRU similar algorithms belong to. The standard of asymmetric encryption ANSI X9.98 – 2010 [2] has a number of drawbacks [3]. The work [3,4] offers new parameters that is, a different polynomial and different modulus for use (NTRU Prime, hereinafter referred to as NTRUPrime). We have investigated the possibility of using these parameters for encryption on the part of the most important characteristic which distinguish NTRU methods from the rest of algorithms, namely, speed regulation characteristics. In fact, standard ANSI X9.98 – 2010 is used for this, but with NTRUPrime mathematics. This is the third paper out of a series of works devoted to this class of algorithms [5,6]. Further analysis on the part of cryptographic security (*further - cryptosecurity*), including that under the conditions of using quantum computers will be carried out in the next works of the series.

2 Distinctions of classic NTRU-method from NTRUPrime and their analyses in view of computing complexity [1]

1. The field $(Z/qZ)[X]/(X^N - X - 1)$ is used in place of the ring $(Z/qZ)[X]/(X^N - 1)$. The value of 739 is suggested as N.

2. The value of 9829 is used instead of the value of the parameter q = 2048, which is employed to calculate the modulus of the polynomial coefficients. The values of parameters N and q define the name of the method, which was given by the authors [3], namely, Streamlined NTRU Prime 9829^{739}.

3. The number of 1 and -1 in a private key (the value d_f), and in a blinding polynomial (the value d_r) depend upon N and the required cryptosecurity, but for all the parameters $d_f = d_r$. For NTRUPrime $d_f = d_r = 202$.

[1] - We use the notation parameters adopted in the NTRU standard [2].

4. The full set of parameters for the classic NTRU ensure security from 112 to 256. The chosen set of parameters (N=739, q = 9829), as the authors [3] state, ensures the value of security no less than 128.

Using the ring in the classic method in carrying out multiplication (the most complex operation in encryption - decryption) was just realized with the help of calculating the coefficient index according to the modulus N, the use of the field requires multiplication according to the modulus $X^N - X - 1$.

The value q defines the modulus for calculating all the polynomial coefficients. Calculating by the modulus q = 2048 does not require using the operation of division, q = 9829 requires the said operation. Besides, the value q specifies the length of a public key. The public key is a polynomial of power N, whose boundary coefficients values are defined by the value q. The public key length is $11N$ bits for q = 2048, it is $14N$ for q = 9829. We obtain 10346 bits or 1294 bytes for N = 739. The authors [3] suggest using a combined radix (2-10) for assuming a public key in a packaged format, than the public key requires 1232 bytes. The saving of less than 5% of address space for assuming the public key will lead to time consumption for setting this key, the more so, as it is an outside user public key, and such a transformation must be performed for each new user.

The preliminary analysis of the main parameters show that the computing complexity for NTRUPrime is anticipated to be more, than that for the classic NTRU. It is this analysis that is realized in the paper.

3 Calculating additional parameters for NTRUPrime

In connection with the need for implementing the classic NTRU for new parameters, it is necessary to calculate additional parameters which are used to implement the classic NTRU.

This parameter and formulae for their calculation are given in Table 1.

Basic operations

Operations of converting between byte strings and polynomials, operations of polynomial multiplication and inversion calculation are used as basic operations. The conversion algorithms are given in [2].

3.1 Operations of polynomial multiplication

Their own multiplication algorithms are used for encryption and decryption:

for encryption: c = a01_1 * b;

for decryption and generation of a public key: c = 3(a01_1 * b) + b;

where:

a01_1 is a polynomial in field $\frac{\left(\frac{Z}{3Z}\right)[X]}{(X^N - X - 1)}$;

b, c is a polynomial in field $\frac{\left(\frac{Z}{qZ}\right)[X]}{(X^N - X - 1)}$.

The work [3] proposes using Karatsuba, multiplication methods and their combinations. We have verified the efficiency of these methods for the polynomials with N = 739 and have not achieved the operation speed increase. This is due to a great number of zero elements in the polynomial coefficients (not less than $N/3$) as well as the actual lack of multiplication operations, an addition operation is employed in place of them.

We used the algorithm of multiplication of polynomials [4]. Operations AVX2 are used instead of SSE operations.

This algorithm has been redesigned with the new modules:

We used the algorithm of multiplication of polynomials [4]. Operations AVX2 are used instead of SSE operations.

Table 1 - Additional parameters for NTRUPrime [2]

Designation	Purpose	Calculation formula	Value for NTRUPrime
d_G	Amount 1(-1) in polynomial G	$d_G = N/3$	246
bLen	Length of a random sequence. It is defined by cryptosecurity	$bLen = S$	192
maxMsgLenBytes	Maximum length of a message for encryption. It is determined taking into account *bLen* and the manner of encoding bytes of a message.	$\frac{\frac{3(N-1)}{2} - bLen}{8} - 1$	113
Llen	Number of bytes to define the length of a message to be encrypted	$\lceil Log_{256} \max MsgLenBytes \rceil$	1
Buffer LenBits	The length of the encoded data buffer	*bLen + (Llen + maxMsgLenBytes)*8*	1104
dm0	Number of 1(-1) in a polynomial, which is created after transforming the message being encrypted	$dm0 = d_f$	202
Hash HashLen	Hash being used [3]	*Depends upon crypto security*	SHA256, 256
C	Number of bits for defining the number of a polynomial coefficient	More $\lceil Log_2 N \rceil$	11
minCallsR	Minimum number of calling up a hash function to create a blinding polynomial	More $\left\lceil \frac{4 d_r c}{HashLen} \right\rceil$	35
minCallsMask	Minimum number of calling up a hash function to mask a message	More $\left\lceil \frac{16N}{5 * HashLen} \right\rceil + 1$	11

Remark: [2] - *Here we use the notation parameters adopted in the NTRU standard [2];*

[3] - *The authors [3] declare cryptosecurity of over 128 for the parameters chosen, therefore the cryptosecurity value is chosen to be 192, and the appropriate hash function is SHA256.*

This algorithm has been redesigned with the new modules:
$x^N - x - 1$ modulo reduction is performed using code:

```
    int j, k = N;
    c[0] += c[N];
    for (j = 1; j < N-1; j++, k++);
          c [j] += c [k] + c[k + 1];
c [N-1] += c[k];
```

This operation requires only N additions. Loop may be performed in parallel.
Q modulo reduction is performed with AVX2 operations.

3.2 Inversion of the polynomials

The inversion operation is applied to calculate the public key that is defined by the formula

$$h = f^{-1}gp,$$

where:

$f = 3F+1$, F is a polynomial with coefficients -1, 0, 1. The number 1, -1 is the same and is equal to d_f;

g - is a polynomial with coefficients -1, 0, 1. The number -1 is equal to $d_g = N/3$, the number 1 equals $d_g + 1$;

$p = 3$.

As it is stated in the work [7] the algorithm "almost inversion" is the most efficient one among the considered algorithms of calculating inversion for the classic NTRU, but, unfortunately, this algorithm cannot be used for calculating inversion with reference to the polynomial $x^p - x - 1$ so as with reference to q that is not the power 2. In this case it is necessary to apply Extended Euclidean Algorithm that is optimized at the expense of replacing data swapping by address swapping.

4 Algorithms of encryption and decryption

4.1 Encryption algorithm

Algorithm 23 and Algorithm 24, respectively, are defined in [2].

The main stages of these algorithms and their optimization are considered next.

Components:

Parameters *N*, *q*.

Additional parameters (Table 1).

Input:

The message *m*, which is a byte string of l bytes length,

The public key *h*.

Output:

Chiphertext e, which is a byte string or Error, if length l exceeds the parameter maxMsgLenBytes.

The algorithm of encryption is composed of the following steps.

Step 1. Forming byte string *M* of bufferLenBits/8 bytes length as:

$$b \,||\, octL \,||\, m \,||\, p0,$$

where:

b is a random bit string of length bLen;

octL, *m* are the length of a message (octL) and the message itself (*m*). To specify message length the parameter Llen is used;

p0 is the number of zero bytes, which add a message m to the maximum length. It is calculated by the formula *maxMsgLenBytes + 1 – l.*

To optimize this step padding byte string *p0* with zeroes is not obligatory it is sufficient to add one 0.

Step 2. Transformation of the obtained byte string M into the polynomial MTrin with coefficients {-1, 0, 1}. To optimize this step we transform 6-bit sequences in place of converting 3-bit ones. As a result, we obtain 4 bytes that comply with 4 MTrin coefficients. To replace the 6-bit data an array of constant, with a size of 64 elements is used that has the form:

```
0x00000000,     //{0, 0, 0, 0},
0x01000000,     //{0, 0, 0, 1},
0xFF000000,     //{0, 0, 0, -1},
…
```

If we do not obtain all the coefficients as a result of processing the byte string M, the remaining coefficients are filled with zeroes.

Step 3. Forming the byte string *sData* with the format:

OID || *m* || *b* || *hTrunc*,

where:

OID - is an identificator that is taken from parameters;

m - is a message for encryption (byte string);
b - is a random bit string of length bLen;
hTrunc - is a part of a packaged public key of length bLen bits.

To optimize this step, a common memory is used for the byte strings *sData* and M, which enables not to copy messages for forming the byte string *sData*. The public key is written in a container at its setting, which enables not to realize its transformation in a byte string at forming *sData*.

Step 4. Forming a *blinding polynomial r*. The formation algorithm (Algorithm 18 [2]) uses IGF algorithm to form a bit string, which is further used to define the numbers (indices) of the polynomial coefficients that possess the value 1, and then -1. Defines two algorithms of forming IGF [2]: IGF-2 (Algorithm 20) and IGF-RBG (Algorithm 21). We use Algorithm 20. According to algorithm 20, the hash function is called minCallsR times. For each call the byte string *sData* is added by the call number. To optimize this algorithm the hash calculation step for constant part is performed one time.

Step 5. The multiplication operation $R = r*h$ is the most time-consuming one.

Step 6. Calculating the bit string oR4. We calculate Ri according to the modulus 4 for each coefficient *R*. Then we perform packing of the obtained coefficients: 4 coefficients per byte. Finally, we obtain the bit string oR4.

Step 7. MGF transforming for the byte string oR4 with allowance for the parameter minCallsMask and obtaining the polynomial *mask*. The byte string is formed similar to Step 4. But oR4 is used as an input bit string and minCallsMask is applied as a call up number. The obtained byte string is used to form the polynomial. 5 polynomial coefficients are formed out of each byte whose value does not exceed 3^5. To optimize the coefficients calculation an array of constants of 273 sequence long is used.

The beginning of the sequence is:

```
{ 0, 0, 0, 0, 0 };
{ 1, 0, 0, 0, 0 };
{ -1, 0, 0, 0, 0 };
…
```

Step 8. Calculating the ciphertext

```
r1 = (r + Mtrin) %3
if (r1.count (1) < df or r1.count (-1) < df or r1.count (0) < df ) goto Step 1
rh := rh + r1
```

To optimize step all the necessary operations are carried out for each coefficient.

Step 9. Transforming the polynomial rh into the byte string em.

4.2 Decryption algorithm

It is made up of the following Steps.

Parameters. *N*, *q*

Input. Ciphertext (byte string em) and its length (len), private key f.

Output. OpenText (byte string m) and its length (l) or Error.

Step 1. Transforming byte string em into the polynomial e.

Step 2. Set cm'= f * e.

Step 3. Set d := cm' mod 3.

Step 4. if (d.count (-1) < df or d.count (-1) < df of d.count (0) < df)
Return Error;

Step 5. Set coR4 := ConvertToBytes((e - cm') mod 4).[4]

Step 6. Generating the mask polynomial. MGF transformating for the bit string coR4 in view of the parameter minCallsMask and obtaining the polynomial mask (see Step 7 of the encryption algorithm) and generating cMtrin.

Step 7. Transforming cMtrin into the bit string.

[4] - For optimizing steps 3-5 are realized as one step for each polynomial coefficient.

An index table is used for optimizing:

Input		0	0x0001	0x00FF	0x0100	0x0101	0x01FF	0xFF00	0xFF01
Output	0	3	6	1	4	7	2	5	

If the element 0xFFFF is available among the input elements, then return Error.

Step 8. Defining an open text and its length.

In realizing this Step it is necessary to check the length of an open text (*it cannot exceed the maximum possible one*) and the availability of the required number of zero elements at the end.

5 Experimental results

All the experiments have been performed on the computer Intel® Core™ i5-4400 CPU - 3.10 GHz, OS Windows 7 (64 bit).

With the help of profiling the functions have been defined that require the maximum time at encryption and decryption. These are functions of polynomial multiplication.

The time of realizing the functions of multiplication for polynomials with coefficients (-1, 0, 1) for encryption and coefficients (-3, 0, 3) with the exception of the first coefficient for decryption as well as the functions of encryption and decryption has been determined. For comparison similar results for the classic NTRU and NTRUPrime are given. The parameters which satisfy the following criteria were chosen for the classic NTRU out of all the parameters:

- cryptosecurity level S = 192;
- the value N closest to 739;
- ratio of the number of nonzero elements to N closest to $404/739 \approx 0{,}55$.

$N = 677$, $d_f = 157$ are the closest parameters that satisfy all the conditions.

Table 2 shows the result of the calculation experiment. These results are shown for the implementation [4] (first column), and realization of our (second column).

Table 2 – Results of the calculation experiment

	MUL (ms)		Encryption (ms)		Decryption (ms)	
Classic NTRU	0,021	0,0132	0,054	0,0394	0,076	0,0232
NTRUPrime		0,0288		0,055		0,04

6 Conclusions

Since the advent of quantum computers, the NTRU cryptosystem, that is based on cryptosecurity of lattice [1], will become one of the most perspective asymmetric cryptography methods satisfying security and operation speed conditions.

The classic NTRU [2] has been used since 2010, had shortcomings that can be eliminated due to applying the new parameters [3].

A possibility of using the parameters from [3] to the algorithm [2] has been show and implementing the suggested combination scheme has been carried out in the paper.

The results obtained are given in the table 2. Our implementation is superior to the implementation of [4]. The classic method provides better speed characteristics than the new one by a factor of practically 2, which has been anticipated. Indeed, the use of the module, inconvenient in terms of computational complexity, leads to this result. But in view of the fact that the NTRU algorithm is tens and hundreds of times smaller as the existing asymmetric methods of encryption in speed characteristics, such a deceleration not being critical.

The cryptosecurity of the suggested methods and possibility of using other parameters to increase cryptoresistance will be considered in detail in subsequent works.

References

[1] Report on Post-Quantum Cryptography [Electronic resource]. – Way of access: http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf.
[2] American National Standard X9.98-2010. Lattice-Based Polynomial Public Key Encryption Algorithm Part 1: Key Establishment: Part 2: Data Encryption, 2010.
[3] J. Daniel. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange and Christine van Vredendaal1. NTRUPrime [Electronic resource]. – Access mode: https://ntruprime.cr.yp.to/ntruprime-20160511.pdf.
[4] [Electronic resource]. – Access mode: https://github.com/NTRUOpenSourceProject/ntru-crypto.
[5] Gorbenko I.D. Features of Parameters Calculation for NTRU Algorithm / I.D. Gorbenko, O.G. Kachko, K.A. Pogrebnyak // Applied Radio Electronics: Sci. Journ. – 2015. – Vol. 14. – № 3. – P. 272-277.
[6] Kachko O.G. Analysis, Estimates and Suggestions With Respect To System parameters Generation Method in NTRU – Similar Assymetric Systems / O.G. Kachko, K.A. Pogrebnyak, L.V. Makytonina. // Radio Eengineering: Sci. Journ. – 2016. – Vol. 186. – P. 103 – 110 [in Ukrainian].
[7] Kachko E.G. Research of methods of calculating of inversion in NTRU Algorithm / E.G. Kachko, D.S. Balagura, K.A. Pogrebnyak, Yu.I. Gorbenko // Applied Radio Electronics: Sci. Journ. – 2013. – Vol. 12. – № 2. – P. 254 – 257 [in Russian].

Рецензент: Роман Олійников, д.т.н., проф., Харківський національний університет імені В.Н. Каразіна, м. Харків, Україна.
E-mail: roliynykov@gmail.com

Надійшло: Грудень 2016.

Автори:
Іван Горбенко, д.т.н., проф., лауреат Державної премії України, Харківський національний університет імені В.Н. Каразіна, Харків, Україна. E-mail: gorbenkoi@iit.kharkov.ua

Олена Качко, к.т.н., проф., Харківський національний університет радіоелектроніки (ХНУРЕ), Харків, Україна.
E-mail: kachko@iit.com.ua

Глеб Науменко, студент, факультет комп'ютерних наук**,** Харківський національний університет радіоелектроніки, Харків, Україна. E-mail: naumenko.gs@gmail.com

Реалізація NTRU –подібного алгоритму на базі NTRUPrime.

Анотація. Сучасні атаки використовують спеціальну структуру кільця в NTRU-подібних алгоритмах. Постквантові NTRUPrime не використовують такого кільця. В роботі досліджується можливість використання цих параметрів для шифрування з боку найважливішої характеристики, яка відрізняє NTRU методи від решти методів несиметричного шифрування, а саме швидкодії. Фактично стандарт ANSI X9.98 – 2010 використовується для цього, але з NTRUPrime математикою. Застосування AVX2 команд для множення поліномів та ефективна реалізація необхідних операцій забезпечили мінімальне зменшення продуктивності.

Ключові слова: стандарт ANSI X9.98 – 2010, NTRUPrime, швидкість шифрування, швидкість дешифрування.

Рецензент: Роман Олейников, д.т.н., проф., Харьковский национальный университет имени В.Н. Каразина, Харьков, Украина. E-mail: roliynykov@gmail.com

Поступила: Декабрь 2016.

Авторы:
Иван Горбенко, д.т.н., проф., лауреат Государственной премии Украины, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина. E-mail: gorbenkoi@iit.kharkov.ua

Елена Качко, к.т.н., проф., Харьковский национальный университет радиоэлектроники (ХНУРЭ), Харьков, Украина.
E-mail: kachko@iit.com.ua

Глеб Науменко, студент, факультет компьютерных наук, Харьковский национальный университет радиоэлектроники, Харьков, Украина. E-mail: naumenko.gs@gmail.com

Реализация NTRU-подобного алгоритма на основе NTRUPrime.

Аннотация. Современные атаки используют специальную структуру кольца в NTRU подобных алгоритмах. Пост квантовый NTRUPrime не использует такого кольца. В работе исследуется возможность применения этих параметров для шифрования с точки зрения самой важной характеристики, которая отличает NTRU методы от остальных методов несимметричного шифрования, а именно быстродействия. Фактически стандарт ANSI X9.98 – 2010 используется для этого, но с NTRUPrime математикой. Применение AVX2 команд для умножения полиномов и эффективная реализация необходимых операций обеспечили минимальное уменьшение производительности.

Ключевые слова: стандарт ANSI X9.98 – 2010, NTRUPrime, скорость шифрования, скорость дешифрования.