Development of modification of the post-quantum public-key cryptosystem NTRUEncrypt

Nikita Lyashenko¹, Larisa Cherchesova¹, Elena Revyakina¹*, Dmitry Medvedev², and Alexander Gavlitsky²

¹Don State Technical University, 1, Gagarin Sq., 344002 Rostov-on-Don, Russia
²Institute of Service and Entrepreneurship (branch) DSTU, Shevchenko St., 147, 346500 Shakhty, Rostov region

Abstract. This work is devoted to the development of a modification of the post-quantum public-key cryptosystem NTRUEncrypt. Given that two main requirements for modern cryptographic algorithms are resistance to attacks (including quantum attacks) and performance, the developed modification offers an improvement in both aspects. Karatsuba algorithm for fast polynomial multiplication is employed to achieve better performance. The modification also includes additional protection against a chosen ciphertext attack that can be effectively against standard NTRUEncrypt. Performance test of the developed modification confirmed that less time is required for key generation, encryption and decryption in comparison with the classical algorithm. The modified algorithm is then applied to implement an asymmetric encryption system with a graphical user interface that allows establishing communication between two users with resistance to both classical and quantum attacks. Keywords: post-quantum cryptography, quantum attack, public-key cryptography, NTRUEncrypt.

1 Introduction

In the modern world, cryptography is becoming more and more important due to the rapid development of information technology and the increasing threat of cyber attacks. Recent advances in quantum computing make it possible to build powerful quantum computers that can be used by malicious users to break classical cryptographic algorithms. Classic asymmetric encryption algorithms such as RSA and the ElGamal algorithm can be easily broken by the Shor’s algorithm, which can be implemented on a quantum computer to solve both factorization and discrete logarithm problems. For this reason, it is necessary to develop new post-quantum algorithms that will be crypto-resistant even against the attacker who has an access to quantum computer. In practical applications, performance of the encryption algorithm is also significant and should be taken into consideration.

* Corresponding author: revyelena@yandex.ru

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).
The object of research is the post-quantum cryptosystem of asymmetric encryption NTRUEncrypt. The subject of research is the computational complexity of the NTRUEncrypt algorithm and its modification.

The aim of the study is to develop a modified NTRUEncrypt algorithm that will be resistant to modern quantum attacks and will be able to achieve better performance for key generation and encryption in comparison to the classical NTRUEncrypt algorithm.

In accordance with the purpose of the study, the following objectives were determined:
- Study modern attacks on the NTRUEncrypt algorithm
- Develop a modification of the NTRUEncrypt algorithm
- Implement the classical algorithm and its modification
- Implement a software tool for data encryption using a modified post-quantum algorithm NTRUEncrypt.

The main research methods are analysis, comparison and experiment.

2 Materials and methods

2.1 Description of the NTRUEncrypt algorithm

The first version of the NTRUEncrypt algorithm was presented by scientists Jeffrey Hoffstein, Jill Pipher and Joseph H. Silverman in 1996. The original version of the cryptosystem was called NTRU. After analyzing this system, changes were made to the recommended parameter sets because new parameters provided better resistance to quantum attacks. In 2011, NTRUEncrypt was included in the IEEE P1361.1 standard [1].

In 2016, the system was announced for participation in the NIST competition. The purpose of this competition is selecting post-quantum cryptography algorithms for standardization [2]. There are two main criteria for choosing the best algorithms: performance and resistance to modern attacks [3]. In the second round of the competition, NTRUEncrypt was merged with the NTRU-HRSS-KEM system under the common name NTRU. This merged algorithm was subsequently selected for participation in the third round of the competition [4].

NTRUEncrypt is an asymmetric algorithm. This means that each of the two participants in the exchange of messages uses two keys: a public key and a private key. The public key is used for encryption. The NTRUEncrypt cryptosystem includes algorithms for generating a public key, encrypting a message with a public key, and decrypting a message using a private key [5].

An important advantage of the NTRUEncrypt algorithm is its high performance compared to other asymmetric algorithms [6]. The speed of the algorithm is achieved due to the fact that the main operation in encryption is the multiplication of polynomials with a relatively low degree. For a message of length n, the computational complexity of encryption is \(O(n^2)\), while for RSA, the complexity of encryption and decryption is \(O(n^3)\) [7]. Also, the NTRUEncrypt cryptosystem does not require large amount of memory and can be implemented on mobile devices [8].

The cryptographic system NTRUEncrypt uses operations on the ring \(Z[X] / (X^N - 1)\) of polynomials, the degree of which does not exceed \(N - 1\) [9]. Any polynomial \(A\) used in the cryptosystem can be written as:

\[
A = a_0 + a_1 \cdot X^1 + a_0 \cdot X^2 + \cdots + a_{N-1} \cdot X^{N-1}
\]  

(1)

where \(a_0, a_1, a_2 \ldots a_{N-1}\) are integers.
The cryptosystem is determined by a number of parameters, the main of which are: N, p and q. The parameters must meet the following requirements:

1) N is a prime number,
2) q \(> p\),
3) q and p are coprime.

Three additional parameters (df, dg and dr) should also be defined. They affect the properties of the polynomials that are used in public key generation and encryption.

Consider an example in which Bob sends a message to Alice. To send a message, Bob needs to choose a private key and use the private key to generate a public key. Alice can then encrypt the message for Bob using the public key. Next, Bob chooses two polynomials f and g. They must meet the following conditions:

The polynomial f must have df coefficients equal to 1, df-1 coefficients equal to -1, and all other coefficients of f must be equal to 0. Also, the polynomial f must have inverse polynomials in modulus f and q.

The polynomial g must have dg coefficients equal to 1 and dg coefficients equal to -1. The rest of its coefficients should be equal to 0 [10]. After choosing the polynomials f and g, the inverse polynomials \( f_p \) and \( f_q \) are calculated for them using the following formulas:

\[
f \ast f_p = 1 \pmod{p} \tag{2}
\]

\[
f \ast f_q = 1 \pmod{q} \tag{3}
\]

The polynomials (f, \( f_p \)) form the secret key. The public key is calculated by the formula [11]:

\[
h = p \ast f_q \ast g \pmod{q} \tag{4}
\]

Bob sends the public key to Alice. After that, Alice can encrypt the message M according to the following algorithm:

1) Represent the message m as a polynomial with coefficients modulo p from the range \(-p/2, p/2\).
2) Select the polynomial r (the blinding value). The polynomial r must have dr coefficients equal to 1 and dr coefficients equal to -1. The rest of the coefficients must be 0. The encrypted message is calculated by the formula:

\[
e = (r \ast h + m) \pmod{q} \tag{5}
\]

After sending the encrypted message, Bob can decrypt the message using his private key. To do this, Bob calculates the polynomial a using the following formula:

\[
a = f \ast e \pmod{q} \tag{6}
\]

Next, Bob must choose his coefficients from the range \(-q/2, q/2\). After that, polynomial b is calculated as:

\[
b = a \pmod{p} \tag{7}
\]

The original message m is calculated by the formula:

\[
m = f_p \ast b \pmod{p}. \tag{8}
\]
2.2 Related works

In the paper [12] a modification that uses the Toeplitz matrix-vector product (TMVP) was proposed. The resulting modification was tested on the ARM Cortex-M4 processor and made it possible to perform encryption 13% faster and decryption 17% faster than the classical algorithm. In [13], the NTRUEncrypt algorithm was implemented for 8-bit AVR microcontrollers.

In [5], the NTRUEncrypt algorithm is used to create a key encapsulation mechanism (KEM). KEM is an algorithm that allows a symmetric cipher key to be transferred using an asymmetric algorithm.

Modifications based on multidimensional algebras are developed in order to increase cryptographic strength. The article [14] is devoted to the development of the QuiTRU modification of the NTRU algorithm, which is based on transformations in five-dimensional algebra. The algorithm developed by the authors provides a larger key space, which increases the cryptographic strength of the algorithm. In the article [15] the authors developed a modification based on three-dimensional algebra (“tripternion algebra”) with the basis \{1, x, x^2\}. The article [16] proposes a modification that is called NTRTE that is based on quaternions. The authors has proved that if for some parameters NTRU has a cryptographic strength of 2^x, then the cryptographic strength of the developed modification using the same parameters will be 2^{2x}.

In [17], the authors proposed to employ NTRUEncrypt for the purpose of increasing the cryptographic strength of the BB84 quantum key distribution protocol.

The NTRU algorithm can also be used for homomorphic encryption. Papers [18] and [19] implemented a Fully Homomorphic Encryption (FHE) scheme based on NTRU. It bears mentioning that this scheme requires the correct choice of parameters in order to avoid the attack proposed in [20]. In [21], the authors employed the NTRU algorithm and developed a post-quantum blockchain architecture for the Internet of Things.

Based on the analysis of current publications, it can be concluded that scientists continue to explore the possibilities of modifying the NTRUEncrypt cryptosystem to improve its performance and resistance to modern attacks.

3 Results and discussion

3.1 Developing the modified NTRUEncrypt algorithm

When executing the NTRUEncrypt algorithm, the most expensive calculations are operations on polynomials. This means that there are two approach to acceleration of the NTRUEncrypt algorithm:

1) Select polynomials for which the algorithm will run faster
2) Use an optimized polynomial multiplication algorithm

The developed modification combines both approaches for the highest performance of the algorithm. The polynomial f is replaced by the polynomial \( f = 1 + pF \). In this case, f always has an inverse polynomial modulo p:

\[ f^{-1} = f_p = 1 \mod p \]  

Since \( f_p = 1 \), the secret key simplifies to f (instead of a pair of polynomials (f, fp)). Thus, acceleration is achieved for the following reasons:

1) It is no longer necessary to choose the polynomial f in such a way that it has an inverse polynomial.
2) No additional multiplication by polynomial p is required when decrypting.
In order to further speed up the execution of the algorithm for multiplying the coefficients of polynomials, the Karatsuba fast multiplication algorithm is used. The standard multiplication algorithm has computational complexity $O(n^2)$, while Karatsuba's algorithm is $O(n^{\log_2 3}) \approx O(n^{1.58})$ [22].

Consider Karatsuba's algorithm for two numbers $x$ and $y$. The numbers $x$ and $y$ can be represented as:

$$x = x_1 \cdot c + x_0$$

(10)

$$y = y_1 \cdot c + y_0$$

(11)

$$(x_1 \cdot c + x_0) \cdot (y_1 \cdot c + y_0) = x_1 y_1 c^2 + (x_1 y_0 + x_0 y_1)x + x_0 y_0 = x_1 y_1 c^2 + ((x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0)x + x_0 y_0$$

(12)

Thus, multiplying two numbers can be reduced to doing three multiplications on smaller numbers. These multiplications are also performed using the Karatsuba algorithm (in software implementations, recursion is used to implement this algorithm).

This algorithm works faster than classical multiplication for polynomials with degree $n > 32$ [23]. Since polynomials with a high degree ($N > 1000$) are used to ensure cryptographic strength of 256 bits, Karatsuba's algorithm will achieve better performance. The disadvantage of Karatsuba's algorithm is that for polynomials with degree $n < 32$, this algorithm is slower than unoptimized polynomial multiplication. To overcome this shortcoming, Karatsuba's algorithm is used only for those polynomials whose degree is greater than 32.

### Table 1. NTRUEncrypt parameter sets.

<table>
<thead>
<tr>
<th>№</th>
<th>Name</th>
<th>k</th>
<th>N</th>
<th>p</th>
<th>q</th>
<th>dq</th>
<th>df</th>
<th>dr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ees401ep1</td>
<td>112</td>
<td>401</td>
<td>3</td>
<td>2048</td>
<td>133</td>
<td>113</td>
<td>113</td>
</tr>
<tr>
<td>2</td>
<td>ees449ep1</td>
<td>128</td>
<td>449</td>
<td>3</td>
<td>2048</td>
<td>149</td>
<td>134</td>
<td>134</td>
</tr>
<tr>
<td>3</td>
<td>ees677ep1</td>
<td>192</td>
<td>677</td>
<td>3</td>
<td>2048</td>
<td>225</td>
<td>157</td>
<td>157</td>
</tr>
<tr>
<td>4</td>
<td>ees1087ep2</td>
<td>256</td>
<td>1087</td>
<td>3</td>
<td>2048</td>
<td>362</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>ees659ep1</td>
<td>112</td>
<td>659</td>
<td>3</td>
<td>2048</td>
<td>219</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>6</td>
<td>ees761ep1</td>
<td>128</td>
<td>761</td>
<td>3</td>
<td>2048</td>
<td>253</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>ees1087ep1</td>
<td>192</td>
<td>1087</td>
<td>3</td>
<td>2048</td>
<td>362</td>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td>8</td>
<td>ees1499ep1</td>
<td>256</td>
<td>1499</td>
<td>3</td>
<td>2048</td>
<td>499</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>9</td>
<td>ees541ep1</td>
<td>112</td>
<td>541</td>
<td>3</td>
<td>2048</td>
<td>180</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>10</td>
<td>ees613ep1</td>
<td>128</td>
<td>613</td>
<td>3</td>
<td>2048</td>
<td>204</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>11</td>
<td>ees887ep1</td>
<td>192</td>
<td>887</td>
<td>3</td>
<td>2048</td>
<td>295</td>
<td>81</td>
<td>81</td>
</tr>
<tr>
<td>12</td>
<td>ees1171ep1</td>
<td>256</td>
<td>1171</td>
<td>3</td>
<td>2048</td>
<td>390</td>
<td>106</td>
<td>106</td>
</tr>
</tbody>
</table>

To ensure sufficient cryptographic strength of the algorithm, it is necessary to correctly choose the parameters $N$, $q$, and $p$. Table 1 presents parameters that are safe according to [24]. At the same time, parameter sets 1-4 provide the smallest key size for the required cryptographic strength, parameters 5-8 provide the best performance with a larger key size, and parameter sets 9-12 are selected so as to minimize the value of $C$ for the required cryptographic strength level, which is calculated by the formula:

$$C = S \cdot t^2$$

(13)

where $t$ is the execution time of the algorithm and $S$ is the total length of the public and private keys [25].

When implementing the NTRUEncrypt cryptosystem, it is important to take into account the possibility of an attack based on a chosen ciphertext [26]. Consider this attack:

The attacker creates a ciphertext according to the formula:
\( C(x) = y \cdot h + y \) \hspace{1cm} (14)

where \( y \) is an integer such that \( y < q/2 \) and \( 2y > q/2 \) and \( y \mod p = 0 \) and \( h \) is the public key.

In this case, the message is decrypted with the private key using the formula:

\[
a = f^*C \mod q = (y^*f^*h + y^*f) \mod q = (y^*g + y^*f) \mod q
\] \hspace{1cm} (15)

Since all the coefficients of the polynomials \( g \) and \( f \) belong to the set \{ -1, 0, 1 \}, then the coefficients of the polynomial \( a \), belong to the set \{0, -y, 2y, -2y\}. Since \( y < q/2 \) and \( 2y > q/2 \) are satisfied, when calculating the coefficients of a mod \( q \), only elements of the polynomial with coefficients that are equal to \( 2y \) or \(-2y\) are changed.

If the \( i \)-th coefficient of the polynomial \( a=2y \), then the following equality is true:

\[
a(x) \mod q = y^*g + y^*f - q^*x^i
\] \hspace{1cm} (16)

Further, the decrypted message is calculated by the formula:

\[
m = f_p^*a = (y^*f_p^*g + y^*f^*f_p^*q^*x^i \cdot f_p^*) \mod p
\] \hspace{1cm} (17)

From the fact that \( y \mod p = 0 \) it follows:

\[
y^*f_p^*g \mod p = 0
\] \hspace{1cm} (18)

And the expression (18) can be simplified to:

\[
m = q^*x^i \cdot f_p^* \mod p
\] \hspace{1cm} (19)

In this case, the private key \( f \) can be calculated by an attacker using the formula:

\[
f = -q^*m^{-1} \cdot x^i \mod p
\] \hspace{1cm} (20)

This attack allows generating a relatively small number of ciphertexts to calculate the secret key. As shown in the article [26], 2100 ciphertexts are enough to break the algorithm with parameters \( N=503 \), \( p=3 \), \( q=256 \).

To protect against an attack based on a chosen ciphertext, it is necessary to modify the algorithm. The blinding value must be calculated using the formula:

\[
r(x)=H(m||R)
\] \hspace{1cm} (21)

where \( H \) is a cryptographic hash function, \( m \) is a message, and \( R \) is a random bit sequence.

The \( R \) sequence is sent along with the ciphertext. After the message is received and decrypted, the message must be verified. To do this, \( t = H(m(x)||R) \) is calculated and received message is encrypted again. The message is considered correct if ciphertext obtained after the repeated encryption and received ciphertext are the same.

### 3.2 Implementation

An asymmetric encryption system was implemented in which the developed modification of the NTRUEncrypt algorithm is used as the encryption algorithm. The programming language is Python 3.9. To achieve better performance, the numpy library is used, which
includes implementations of mathematical operations on arrays with optimization in the C
language. Program features include generating a public and private key, encrypting a file
with a public key received from another user, and decrypting a message. The block diagram
for the implemented system is shown in Figure 1.

The user has three sets of parameters to choose from: (N=1087, p=3, q=2048),
(N=1499, p=3, q=2048) and (N=1171, p=3, q=2048). All three parameter sets provide 256-
bit security. The hash function is SHA3-512. The program has a graphical
interface. The user can upload files with public and private keys, files for encryption and files with
ciphertext for decryption. For the convenience of the user, the program displays all the files
uploaded by the user. The File Type column will show one of the following values: public
key, private key, ciphertext (encrypted file), file (plaintext file to be encrypted). In figure
2 an example of successful key generation is shown.

![Block diagram of the implemented program](image)

**Fig. 1.** Block diagram of the implemented program.
Fig. 2. Example of successful key generation.

Figure 3 shows an example of encrypting a text.txt file. The resulting ciphertext was saved to a file.

Fig. 3. Example of successful file encryption.

Figure 4 shows an example of a successful decryption of a file with a successful check against an attack with the chosen ciphertext.
Fig. 4. Example of successful file encryption.

Figure 5 shows an example of an attempt to decrypt the ciphertext chosen in a matched ciphertext attack. In this case, the original ciphertext will differ from the ciphertext received during the reencryption of the message, and the message check will be failed.

Fig. 5. Example of failed message check.

Performance test of the classic NTRUEncrypt algorithm and the modified algorithm was conducted. For each of the two implementations of the algorithm, 200 tests were performed. Key generation time, encryption time, and decryption time were calculated separately. The test results are presented in table 2.

Table 2. Comparison of the standard algorithm with modification.

<table>
<thead>
<tr>
<th></th>
<th>Average execution time of the standard algorithm, seconds</th>
<th>Average execution time of the modified algorithm, seconds</th>
<th>Comparison of the modification with the standard algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key generation</td>
<td>0.09374</td>
<td>0.06728</td>
<td>Modification is 28.23% faster</td>
</tr>
<tr>
<td>Encryption</td>
<td>0.05215</td>
<td>0.03886</td>
<td>Modification is 25.49% faster</td>
</tr>
<tr>
<td>Decryption</td>
<td>0.07482</td>
<td>0.07142</td>
<td>Modification is 4.52% faster</td>
</tr>
</tbody>
</table>
From the test results, it can be concluded that the modification performs key generation, encryption and decryption faster than the standard NTRUEncrypt algorithm.

4 Discussion

The results of the work are a modification of the NTRUEncrypt algorithm and a software implementation of an asymmetric encryption system. The data obtained during performance testing should be discussed in this section. The largest performance gain was obtained for key generation, and the smallest performance improvement was achieved for decryption. These results can be interpreted in accordance with the mathematical description of the modification of the algorithm. Among the three stages of the algorithm, it is the generation of keys that requires the most complex computational operations for multiplying polynomials, and the use of the Karatsuba algorithm has the greatest impact on the acceleration of calculations. The performance improvement in decryption is much less than in encryption, since the modified algorithm uses additional verification of the message after decryption to ensure security against attacks with the chosen ciphertext.

5 Conclusion

The main result obtained as a result of the study is a modification of the NTRUEncrypt algorithm, which, in comparison with the standard algorithm, provides better performance and is resistant to attacks based on the selected ciphertext. Based on the developed modification, an asymmetric encryption system was implemented. Performance of two versions was tested. This test has confirmed the effectiveness of the developed modification.

References