ПЕРСПЕКТИВНЫЕ КРИПТОГРАФИЧЕСКИЕ ПРЕОБРАЗОВАНИЯ И ИХ ПРИМЕНЕНИЕ

ПЕРСПЕКТИВНІ КРИПТОГРАФІЧНІ ПЕРЕТВОРЕННЯ ТА ЇХ ЗАСТОСУВАННЯ

UDC 004.056.55 **DOI:10.30837/rt.2019.3.198.01**

I.D. GORBENKO, Dr. Sc. (Technology), O.G.KACHKO, Cand. Sc. (Technology), A.N. ALEKSIYCHUK, Dr. Sc. (Technology), O.O. KUZNETSOV, Dr. Sc. (Technology), YU.I. GORBENKO, Cand. Sc. (Technology), V.V. ONOPRIENKO, Cand. Sc. (Technology), M.V. YESINA, Cand. Sc. (Technology), S.O. Candiy

ALGORITHMS OF ASYMMETRIC ENCRYPTION AND ENCAPSULATION OF KEYS OF POST-QUANTUM PERIOD OF 5 -7 LEVELS OF STABILITY AND THEIR APPLICATIONS

Introduction

At present, significant efforts are being made by the cryptographic community to create practical quantum-stable mechanisms of asymmetric encryption (ASE), key encapsulation protocols (KEP) and electronic signature (ES) [1 – 7]. The results of the implementation of the 1st stage of the international competition for creation of post-quantum ASE, KEP and ES [1], performed by NIST USA, as well is performed comparisons of alternatives [2, 3], make it possible to conclude about the prospects of application of cryptographic transformations in rings of polynomials (algebraic lattices) for their creation. Such transformations have stood the test of cryptographic stability in the form of the NTRU cryptosystem [1]. In general, NTRUEncrypt ANSI X9.98 [4], NTRU Prime [5], and NTRU Prime Ukraine [5] are examples of implementations for the ASE and KEPs of cryptographic transformations on algebraic lattices. Mechanisms for constructing the ASE and KEPs are proposed in [2], their application makes it possible to provide the 5^{th} level of cryptographic stability inclusive, i.e. 128 bits of quantum and 256 bits of classical crypto-stability. But in our view, the problem of providing encryption and encapsulation is important both from theoretical and practical point of view, including up to the 7^{th} level of stability, since the 5^{th} level of cryptographic stability is not enough for the quantum period [7 – 9]. That is, the current problem is the problem of creation and standardization of the ASE and KEP algorithms of 256 bit quantum and 512 bit classical crypto-stability [7, 5]. Moreover, in accordance with the requirements, the draft standard of the ASE and KEP, which is considered as the draft national standard of Ukraine officially [8], should provide different modes of operation: *asymmetric encryption; asymmetric encapsulation of keys; asymmetric encryption and encapsulation of keys; asymmetric encryption, encapsulation of keys and key generation for symmetric encryption, ensuring their crypto survivability in the form of "direct secrecy"* [11 – 13]. That is, nowadays, the problem of creating and standardizing post-quantum ASE and KEP algorithms of 128, 192 and 256 bits of quantum, as well as 256, 384 and 512 bits of classical crypto-stability [9 – 11] for valid and selected security models is important.

The purpose of this article is to present and review the constructed algorithms of asymmetric encryption and encapsulation of keys in polynomial rings (algebraic lattices), analysis of the essence of the cryptographic transformations of the used ASE and KEPs. The said purpose, in our view, is achieved by outlining the following problems:

- calculation of general and additional parameters for crypto transformations of the 5-7 levels of cryptographic stability;
- generation of asymmetric key pairs of encryption keys and encapsulation of keys for crypto transformations of the 5-7 levels of cryptographic stability;

- development of algorithms of asymmetric encryption (encryption and decryption) for crypto transformations of the 5-7 levels of cryptographic stability;

- development of algorithms of asymmetric encapsulation and decapsulation of keys for crypto-transformation of the 5-7 levels of cryptographic stability;

- development of proposals (algorithms) for calculating secure session keys for their use in symmetric encryption of data in communication channels;

- estimation of complexity of forward and reverse transformations at asymmetric transformations (encryption and encapsulation).

Questions of estimation of cryptographic stability and justification of the ASE and KEP mechanism parameters are given in [6 – 9].

In the following, we will consider the mechanisms of encryption and encapsulation with three sets of parameters that determine stability, marking them as follows: SKELYA-KEM 256/128; SKELYA-KEM 384/192 and SKELYA-KEM 512/256 [9].

1. Parameters of cryptographic transformations of keys encryption and encapsulation

According to [7, 8, 10], the parameters of cryptographic transformations are divided into basic (general), additional and mechanism parameters.

First and foremost, general parameters are defined to ensure the specified crypto stability as well as to ensure the success of operations (such as the absence of decryption errors), and to reduce computational complexity.

Only general parameters are used to calculate additional parameters. Re-calculating of additional parameters makes it possible to reduce the computational complexity of basic cryptographic transformation operations.

Algorithm parameters are parameters that need to be agreed to share encryption and encapsulation algorithms. Now it concerns identifiers of algorithms, constant messages and the like.

Detailed data on generating general, additional and mechanism parameters are given in [7].

Table 1 lists general parameters of SKELYA algorithms of keys encryption and encapsulation. The justification and calculation of the general parameters are given in [7, 9].

Table 1

General parameters of encryption and encapsulation of SKELYA algorithms keys

SKELYA-KEM 256/128			SKELYA-KEM 384/192			SKELYA-KEM 512/256		
n	*t*	*Q*	*n*	*t*	*q*	*n*	*t*	*q*
881	159	7673	1201	192	9221	1471	255	12251

The following notations are used in Table 1:

n – the degree of the polynomial;

t – the number of nonzero elements in *t* – small polynomial;

q – a large module, a simple number that is relatively simple with a polynomial

$x^n - x - 1$, and the value of *q* is determined by the condition of guaranteeing no decryption error;

p – a small module, $p = 3$.

Table 2 lists the additional parameters of SKELYA algorithms of keys encryption and encapsulation. The justification and calculation of the general parameters are given in [7, 10].

Table 2

Additional parameters of encryption and encapsulation algorithms of SKELYA algorithms keys

Sign	Purpose	Formula or value
qBits	The number of bits in *q* given as a binary string	$qBits = \lceil \log_2 q \rceil$
db	Length *b* (bit)	db= λ
bufferLenBytes	The length of the octet string for the conversion functions between the small polynomial and the octet string (code2to3, code3to2 functions)	*Ceil* (((n-1)/2)*3/8)
maxMsgLenBytes	Maximum message length for encryption (octets)	*(bufferLenBytes–λ/8)–1*
EncMsgLenBytes	The length of the encrypted message (octets)	*Ceil(qBits * n/8)*
cBits	The number of bits for specifying the degree of a polynomial as a binary string	cBits = Ceil($\log_2$n) + 1
Llen	The number of octets to specify the length of the encrypted message	1
pkLen	The number of bits *h* that are used during encryption	pkLen=db

Ceil function in Table 2 defines the smallest integer that is not less than the input argument.

The parameters of the encryption and encapsulation algorithms of SKELA algorithm algorithms are shown in Table 3. The justification and calculation of the general parameters are given in [7, 10].

Table 3

The parameters of the encryption and encapsulation algorithms of the keys and their values for the test version

Sign	Purpose	Formula or value
OID	Method identifier (3 octets)	For the test version OID[0]=0, OID[1]=0, OID[2]=1
m_kem	Permanent message used for encryption in the key encapsulation and decapsulation protocol	A row of octets is used for the test version { 'I' ⊕ 0xFF, 'I' ⊕ 0xFF, 'T' ⊕ 0xFF, ' ' ⊕ 0xFF, 'E' ⊕ 0xFF, 'n' ⊕ 0xFF, 'c' ⊕ 0xFF, 'a' ⊕ 0xFF, 'p' ⊕ 0xFF, 's' ⊕ 0xFF, 'u' ⊕ 0xFF, 'l' ⊕ 0xFF, 'a' ⊕ 0xFF, 't' ⊕ 0xFF, 'e' ⊕ 0xFF, 0 ⊕ 0xFF}
m_kemBytes	Length m_kem (octets)	*m_kemBytes* ≤ *maxMsgLenBytes* *m_kemBytes = 16 for m_kem*

2. Generation of key pairs of asymmetric cryptographic transformations

When generating a specific key pair – private key f and public key h the polynomials G, F are used, whose degree is *n,* and coefficients modulo *p (p = 3)*, that is, they take values 0, -1,1. The result is simultaneously calculated private key f and public key h (polynomials) and h (byte task h). Generation is carried out in such sequence.

Generation of Polynomial G. Polynomial G is a polynomial having (2n + 1) / 3 of nonzero coefficients.

Generation of polynomial F. Polynomial F is a polynomial having 2t nonzero coefficients.

Calculation of the private key f is carried out according to the formula [9]

$$f = pF + 1 \tag{1}$$

Calculation of the public key h is carried out to the formula [9]

$$h = p * G * f^{-1} \text{ в полі } (Z / q[x] / (x^n - x - 1)) \tag{2}$$

To calculate the public key, it is necessary to calculate f^{-1} in the field $(Z / q[x] / (x^n - x - 1))$ and multiply the polynomials in the same field.

Calculation of inverse element

To calculate the inverse element, an advanced Euclidean algorithm for polynomials is used, i.e., the equation:

$$ax + by = d \tag{3}$$

Moreover, in equation (3) it is necessary to set *a* as polynomial f, *b* – as polynomial $x^n - x - 1$. The value of the right-hand side of equation (3), i.e. (d), determines the greatest common divisor for a and b (GCD). All the calculations must be performed modulo *q*. Computed values *d, x, y* can be the result of using an advanced algorithm. If the degree of the polynomial *d* is 0, it is a guarantee for the presence of inverse element since *d* is an integer. In our case $||f||_1 \neq 0$, so the inverse element f^{-1} *mod* $(x^n - x - 1)$ exists, and the value of the variable *x* is the inverse element.

The number of steps that need to be done depends only on the degree of the polynomials, so the completion time of the inversion calculation operation is independent of the specific key

Multiplication of polynomials

The polynomial multiplication operation is defined mathematically as:

$$c(X) = a(X) * b(X), \tag{4}$$

where

$$c'_k = \sum_{\substack{i+j=k \\ i \in 0...n-1 \\ j \in 0...n-1}} (a_i * b_j) \bmod q; \quad c(X) = c'(X) \bmod (x^n - x - 1).$$

The specificity of polynomials used for multiplication is the presence of a large number of null elements in one of the polynomials (n/3 or even more if you use this operation when encrypting). In the implementation of this operation it is necessary to solve 2 problematic tasks:

- providing the least computational complexity;
- ensure the independence of the execution time of a particular key.

Many works are devoted to solving these problems [5 – 11].

3. Algorithms of asymmetric encryption and decryption

This section is devoted to discussion of the asymmetric encryption (encryption and decryption) Skelya algorithms.

When developing encryption algorithms, we have taken into account the NTRU encryption algorithm [4], which has been successfully used for almost 10 years. That is why most of the designations coincide with the designations adopted in [5]. But the algorithm [5] has a significant drawback associated with the possibility of decryption error, which was taken into account when developing a new Skelya algorithm [10].

In addition, the modern practice of using as a module the prime number q instead of the number 2^k and the polynomial $x^n - x - 1$ instead of the polynomial $x^n - 1$ is taken into account, which provides protection against known attacks [5, 10]. That is why the field $(Z / q[x] / (x^n - x - 1))$ is used as the field, as for NTRU Prime [5], but the parameters are

calculated taking into account the required levels of crypto-stability λ = {256, 384, 512} and ensuring no decryption errors [5, 10].

3.1. Encryption algorithm

Input:

- a string for encryption (m);
- the length of the string m (mLen);
- the recipient's public key h (polynomial R/q) and the corresponding octet string (h)
- Output:
- a sign of success *Success* (OK, ERROR);
- Encrypted string E in case of successful operation.

The length of the encryption line is limited by the fact that this line is built with additions that will provide semantic security, it can be specified by a polynomial of degree *n*. If the length of the string exceeds this value, the encryption operation returns an error. The maximum length of the valid message (octets), depending on the level of crypto-stability (cryptographic strength) λ (degree of polynomial n), is given in Table.4.

Table 4

The maximum length of the message, depending on crypto-stability (cryptographic strength) λ

λ	256 (n= 881)	384 (n=1201)	512 (n = 1471)
EncMsgLenBytes (octets)	132	176	210

Algorithm for encryption is an iterative algorithm, that continues until a small polynomial is formed, which is used to mask the encrypted message, until it satisfies the conditions:

- number (units) + number (minus) units not less than *2t*
- number (zeros) is not less than *t*.

The masking polynomial is formed on the basis of random components, so it is highly probable that the polynomial coefficients are equally probable. If *t* is significantly less than n/3 the probability of fulfilling this condition is high and the algorithm usually does not require a return for recalculations.

Iteration algorithm

1 An octet line of bufferLenBytes length is formed, where they write:

- random string of λ bits, denoted by *b* (provides semantic security);
- encryption string to which its length is transmitted;;
- zero octets (to complement the required length)

Denote this string *M*. This line depends on the random sreing and the incoming message and has a constant length that does not depend on the length of the message.

2 The octet string *M* is converted to R/3 by a polynomial, for which every 3 bits of the string are converted to two polynomial coefficients according to Table 5.

Table 5

Convertion of bit line into polynomial R/3 and vice versa

Bit string	Polynomial coefficients	Bit string	Polynomial coefficients
000	0, 0	100	1, 1
001	0, 1	101	1, -1
010	0, -1	110	-1, 0
011	1, 0	111	-1, 1

As a result of the transformation the corresponding polynomial *MTrin* is obtained. This polynomial, like string *M*, depends on random data and the encrypted string. It should be noted that *MTrin* is uniquely determined by string *M*, and conversely, string *M* can be restored by *MTrin*

3 An octet string is formed, where they write:

- method identifier (3 octets) that can be selected by agreement of the parties;
- message for encryption;
- random string b;
- a part of the public key h of the *pkLen* bits length

Let's mark this string as S. Let's note that string S has a variable length that depends on the length of the encryption string. String S depends not only on the string for encryption and the random string, but also on the identifier's algorithm and the recipient's public key.

4 Let us transform string S into a dazzling polynomial *r*. The dazzling polynomial is also a polynomial of degree *n* having *2t* nonzero coefficients, the other coefficients being 0. Let us determine the length of the bit string required to define the dazzling polynomial. We will define separately the signs of nonzero elements and their indices. All that should be defined is *2t* characters and *2t* indices. Let us apply *2t* bits to specify non-zero element characters and *2t* indices of these elements. To set each index it is enough to have *cBit* bits. Given the possibility of obtaining an index that has already been used, to set the indices, we form a string twice as long as necessary. Thus, the total length of the string, which should be formed is *2t + 4t * cBit*. In general, the transformation is carried out as follows:

- the pseudorandom data generator is initialized with string S;
- pseudo-random bytes of desired length are generated ();
- signs of non-zero elements are defined;
- indices of non-zero elements are defined;
- if not all indexes are formed, then pseudo-random bytes are generated for the remainder of the indices and we go to the previous step.

That is, the dazzling polynomial depends on the random data, the data being encrypted, and the portion of the public key of who the data are encrypted for.

5 The polynomial R = r * h is calculated in the field R/q

6 The polynomial R4 = R mod 4 is calculated

7 The polynomial R4 is converted to the string of oR4 octets, recording 4 coefficients into one octet. As a result, we get a 2n-bit octet string that depends on the dazzling ng polynomial and the recipient's public key. This line is then used to obtain the polynomial R/3.

8 We define the formation method, the required length of the string that must be used to form the R/3 polynomial with this method. We will use a byte to form five polynomial coefficients. To do this, we present a number in the ternary number system. The numbers 0, 1 ... 241, 242 can be set using 5 digits, each digit corresponds to the numbers 0, 1, 2, or 0, 1, -1. These values will be used to set the coefficients. The number of octets required in this case is Ceil (n/5). But octets with value 243 – 255 can not be used to set coefficients; the input line should be increased taking into account the probability of such octets. A string of a double length guarantees that the required number of coefficients is obtained.

The formation of the masking polynomial is as follows:

- perform initialization of the pseudo-random data generator with oR4 string;
- pseudo-random octets of 2 Ceil (n / 5) length are generated;
- the following 5 polynomial coefficients are calculated for each octet whose value is less than 243;
- the resulting polynomial is denoted as a mask;

9 The polynomial m'= (*MTrin* + maska) mod p is calculated

10 The success of the iteration is checked:

– The number of units (c1), minus units (c2) and zeros (c3) in the polynomial m ' is determined;
– if c1 + c2> 2t and c3> t the iteration is successful, otherwise go to step 1

The following steps are performed, if the iteration is successful in such a sequence:

11 $e = R + m' \pmod q$ polynomial is calculated

12 The polynomial e is converted to the octet string E

The result of the encryption algorithm is a string of octets *E*

The following mathematical transformations are performed in encryption:

1 M = b||mLen||m||0..0 (a string of octets)

2 mTrin = f1 (M) (mTrin – R/3 polynomial, f1 one-to-one function, that is, M can be restored by the value of mTrin)

3 S = oid||m||b|| part of h

4 r = f2 (S) (r – t – small polynomial, f2 – not one-to-one function)

5 R = r * h (in the field R/q)

6 oR4 = f3 (R mod 4) (oR4 – a string of octets, R cannot be restored)

7 maska = f4 (oR4) (maska – R/3 polynomial for which the conditions on the number of 1, -1, 0 are satisfied)

8 m' = mTrin + maska (mod 3)

9 e = R + m' (mod q) (e – polynomial in the field R/q)

10 E = f5 (e) (f5 is the inverse function, so the octet string e can be restored).

3.2 Decryption algorithm

When decrypting, the octet string E is fed to the input of the algorithm, and as a result, the output will receive an open message along with its length (in case of successful completion)

Entry:

E – string of bytes with an encrypted message;

f – recipient's private key;

recipient public key h (polynomial R / q) and corresponding octet string (h)

Output:

1 Sign of success *Success* (OK, ERROR).

2 Open message (in case of successful operation).

3 The length of the open message (in case of successful operation).

Algorithm

1 e polynomial is restored.

2 a = e*f in the field R/q polynomial is calculated

3 m' = a mod 3 is restored

4 m'requirements are checked

– the number of units (c1), minus units (c2) and zeros (c3) is determined in the polynomial m';
– if c1 + c2 > 2t and c3 > t then Success = OK, otherwise Success = ERROR and go (transition) to step 14.

Further steps are performed only in cases when Success = OK

5 R' = e – m' (mod q) is restoring

6 Polynomial R4 = R' mod 4 is calculated

7 R4 polynomial transformation into a oR4 octet string is carried out, recording 4 coefficients in one octet.

8 Calculation of the *maska* polynomial (see item 9 of the encryption algorithm)

9 mTrin = m' – maska (mod p) polynomial is calculated and M string is restored.

10 Determining the individual fields of M row:

– first λ bits – restored random sequence (*b*);
– next octet – length of encrypted data (restored mLen value);
– next mLen of octets – recovered message that was encrypted (m);

– next octets – is a restored string of addition.

11 Checking the success of the operation. The operation is considered successful if all conditions given below are met simultaneously:

– mLen restored value does not exceed the maximum admissible *EncMsgLenBytes* value;
– supplement (addition) string contains zeroes, the number of which is equal bufferLenBytes – db/8 – 1 – mLen

If at least one condition is not met, then Success = ERROR and go to step 14. Further steps are performed only in cases of successful operation.

12 The recovered values *b*, mLen, m are used to create a string S, calculate r, and calculate R = r * h in R/q field (steps 3-5 of the encryption algorithm).

13 Final check of the operation success: if R' recovered in step 5 matches the value of R obtained in step 12, the decryption operation ends successfully, the message m and its length mLen bytes is decrypted at the output.

14 If the operation fails (Success = ERROR), an empty message of length 0 is returned.

The following mathematical transformations are performed when decrypting

1 $e = f5^{-1}$ (E)

2 $a = e*f = (r*h + m') * f = (r * 3Gf^{-1} + m') * f = r * 3G + m'* f$ in the field R/q =

3 a mod 3 = m' (r * 3G mod 3 = 0; f mod 3 = 1)

4 R' = e – m' (mod q)

5 R4 = R' mod 4

6 oR4 = f3 (R4)

7 maska = f4 (oR4)

8 mTrin = m' – maska (mod 3)

9 $M = f1^{-1}$ (mTrin)

10 When using M = b || mLen || m || 0..0 (string of octets) separate fields are selected, as a result we get b, mLen, m.

4. Algorithms for encapsulation and decapsulation

For encapsulation algorithms, key data are used that are generated identically as for encryption. In fact, encapsulation algorithms use message encryption and decryption algorithms that are defined in advance and labeled *m_kem*. The length of the message *m_kemBytes*. Full description of the algorithms described above see in section 3.

An arbitrary allowed hash function is used as a hash function, providing a result length of 512 bits. We denote this function by Hash512.

Function input: octet string and its length.

Output: 512-bit octet string

4.1. The encapsulation algorithm

Input:

– length of the key for the symmetric encryption K_bytes (K_bytes $\leq \lambda/8$) ;
– the recipient's public key h (polynomial R/q) and the corresponding octet string (h).

Output:

– sign of success *(Success=OK, ERROR);*
– encapsulated key *Cc*;
– key SKey for symmetric encryption, the length of which is K_bytes (K_bytes).

As with the encryption algorithm, the first step is to check whether the operation can be performed:

– *Success= OK;*
– if *m_kemBytes* > *maxMsgLenBytes* then *Success= ERROR;*
– if K_bytes * 8 > λ then *Success= ERROR*

If *Success= ERROR* then the algorithm is not executed.

Next, the variant is considered in case of successful verification (*Success= OK;*)

1 We first perform steps 1-12 of the encryption algorithm for *m_kem* message of the length of *m_kemBytes*.

2 The dazzling polynomial is converted to the octet string *r* as follows:

- bit string bs1 is formed. The next bit of the string is 0 if the next polynomial coefficient is 0 and 1 if the next coefficient is 1 or -1. The length of the string is bs1 = n bits.
- bit string bs2 is formed. The next bit of the string is 0, if the next non-zero element is equal to1 and vice versa. The length of the string is bs2 = 2t bits.
- form a common bit string *bs = bs1 || bs2*
- denote r string of bytes, which corresponds to the bit string *bs*.

3 The value of the encapsulated key Cc and SKey is calculated.

c = *E*

The rest of the calculation depends on the value of λ.

If λ = 256, then:

H = Hash512 (r_);

C – are younger 32 octets of H;

SKey – are SKeyLen octets of senior 32 octets of H (SKeyLen ≤32);

Cc = C|| *c.*

If λ = 512, then

H1 = Hash512 (r_ || 1);

H2 = Hash512 (r_ || 2);

C – H1;

SKey – SKeyLen octets of H2 (SKeyLen ≤64);

Cc = C|| *c.*

4.2. Decapsulation algorithm

Input:

- length of the key for the symmetric encryption K_bytes (K_bytes ≤λ/8;
- encapsulated key *Cc*;
- recipient's private key *f*;
- recipient's public key h (R/q polynomial) and corresponding octet string (h).

Output:

- sign of success *(Success=OK, ERROR);*
- key SKey for symmetric encryption, the length of which is K_bytes (K_bytes).

As with the encryption algorithm, the first step is to check whether the operation can be performed:

- *Success= OK;*
- if *m_kemBytes* > *maxMsgLenBytes* then *Success= ERROR;*
- if K_bytes * 8 >*λ* then *Success= ERROR*

If *Success=ERROR* then the algorithm is not executed.

Next, the variant is considered in case of successful verification (*Success= OK;*)

1 Decoding *Cc* to *C* and *c.*

2 Decoding *c* is performed (steps 1-13 of the decryption algorithm)

3 If the decryption error *(Success = ERROR),* the algorithm returns an error.

4 If the length of the encrypted message does not match the message itself, or the messageы do not match, then Success = ERROR, the algorithm returns an error.

5 The recovered values *b,* mLen, m are used to create a string *S* and calculate *r.*

7 An octet string is formed for *r (r)* (see step 2 of the encapsulation algorithm)

6 The rest of the calculations depend on the value of λ.

If λ = 256, then:

H = Hash512 (r_);

C’ – are younger 32 octets of H;
SKey’ – are SKeyLen octets of senior 32 octets of H (SKeyLen ≤32).
If λ = 512, then:
H1 = Hash512 (r_ || 1);
H2 = Hash512 (r_ || 2);
C’ – H1;
SKey’ – are SKeyLen octets of H2 (SKeyLen ≤64).
7 If C’coincides with C then SKey = SKey’; sign of success *Success = OK*, otherwise *Success = Error*

5. Data encryption and authentication by a sender

During data encryption, consistent (agreed) functions, available to all subscribers, can be applied. They contain *MAC*(.) messages authentication and a symmetric encryption algorithm (Sym.Encrypt), for example [3, 14].

Subsequently, secret keys E_K and M_K, are produced using SKey, where E_K is the key for symmetric encryption and M_K is a suitable authentication key for the message of pre-encrypted data. (Algorithms for generating keys E_K and M_K from SKey can be determined by other standards).

5.1. Algorithm of data encryption and authentication by a sender

The algorithm is performed in the following sequence:

- messages (data, packet) Q are encrypted with the use of symmetric cipher on the key E_K, i.e.

$$C1 = Sym.Encrypt (Q, E_k);$$

- to authenticate C1ciphertext on M_K authentication key, calculate the C2 authentication code, i.e.

$$C2 = MAC (C1, M_k);$$

- finally, the sender sends *(Cc, C1, C2*) to the recipient.
-

5.2. Decryption and authentication of data by the recipient

The algorithms for decrypting and authenticating data by the recipient are executed in the following sequence:

- as a result of decapsulation, the recipient has the value *c',C',*SKey',*r';*
- if *r '*is *t*-small, *c' = c* and *C '= C*, then they output and apply the symmetric authentication and decryption key to *SKey'*. Otherwise, they determine the error and reject the secure message;
- the recipient calculates the message authentication code for C1 on the M_K key

$$C2' = MAC (C1, M_k)$$

- if *C2'* coincides with *C2* received from the sender, then the encrypted message *C1* is considered to be complete and authentic. Otherwise output "integrity violation" and stop executing the decryption algorithm;
- the recipient decrypts the cryptogram *C1* using the symmetric decryption algorithm agreed with the sender on E_K key, i.e.

$$Q = Sym.Decrypt (C1, E_k)$$

- the recipient receives the decrypted and authenticated *Q* data for further processing.

Note. If necessary, the order of encryption and authentication of Q data may be carried out in a different order, that is, first decryption and then authentication.

6. Analysis of cryptographic transformations complexity

The temporal characteristics of key data generation, encryption and decryption algorithms are shown in Table 6. Encapsulation and decapsulation algorithms actually use encryption and decryption primitives. Therefore, their temporal characteristics are not given. Processor (Intel (R) Core (TM) i5-3.1 GHz) clock [10, 11] is used to measure performance. For comparison, the data [5] were used for the ntruees787ep1 encryption algorithm, which is closest to the characteristics with crypto-stability λ=256.

Table 6

Temporal characteristics of key data generation, encryption and decryption algorithms

Stability	Generation	Encryption (59 bytes)	Decryption (59 bytes)
ntruees787ep1	16748110	111142	133926
λ=256 (n = 881)	4789644	87696	96604
λ=384	8065140	130556	143424
λ=512	11670676	160364	182724

The results of optimization of key data generation, encryption and decryption algorithms are presented in [6 – 10].

Key generation. The most laborious part of the algorithm is the inversion calculation and multiplication of polynomials in R/q field.

Inversion calculation. An extended Euclidean algorithm for polynomials is used to calculate the inversion. The computational complexity of Euclidean extended algorithm is determined by the number of division operations and the computational complexity of one operation.

The polynomials of degree *n* and *n-1* are used as data for the inversion calculation. Performing a single division operation reduces the degree of both polynomials by 1, i.e. the number of division operations is n.

The computational complexity of the division operation depends on the degrees of polynomials and the difference between them. It can be considered that the difference between degrees is one, then, the number of coefficients processed by the division operation equals a smaller degree and accordingly changes from n to 1. That is, labor intensity can be estimated as n-1 + n -2 +… + 1 = n (n-1)/2. Thus, the overall complexity of the investment stage is $O(n^3)$.

Methods for optimization of calculations.

1 To apply SIMD operations to work with the coefficients – there are limitations associated with variable block addresses.

2 Recalculation of inverse elements for coefficients.

Operation of polynomial multiplication

Computational complexity of multiplication of polynomials of general form $O(n^2)$.

Optimization through the use of polynomial F/3 features with compensation of dependence on the key format enables you to pass from the quadratic to the linear dependence on the degree of the polynomial.

The computational complexity of encryption and encapsulation operations

In fact, the encapsulation algorithm uses the encryption algorithm, and the decapsulation algorithm uses the decryption algorithm. The rest of the operations performed for encapsulation / decapsulation take relatively little time. That is why it is enough to consider the computational complexity for encryption and decryption algorithms.

Conclusions

1. The generalized results of the implementation of the 1st stage of the international competition for the creation of post-quantum ASEs, KEPs and ESs, proposed by NIST USA, as

well as comparisons of most alternatives, allow us to make a conclusion about the prospects of creating standards for these cryptographic transformations in polynomial rings (algebraic lattices) [1 – 7]. Another interesting area that also deserves attention and exploratory research is the construction of post-quantum cryptographic transformations using methods of the theory of fault-free coding [15 – 17].

2. Mechanisms for constructing ASE and KEP are proposed in [2], the application of which makes it possible to provide 128 bits of quantum and 256 classical crypto-stability, i.e. including the 5^{th} level of cryptographic stability. But in our opinion, the problem of ensuring encryption and encapsulation, including up to 7 levels of stability, is important both from theoretical and practical point of view, since the 5th level of cryptographic stability for the quantum period is insufficient.

3. Therefore, the current problem is the creation and standardization of ASE and KEP algorithms of 256 bit quantum and 512 classical cryptographic stability. Practically the problem of creation and standardization of post-quantum algorithms of ASE and KEP of 5 – 7 levels of cryptographic stability is solved in Ukraine based on cryptographic transformations using algebraic lattices (polynomial rings).

4. The common parameters for ASE and KEPs of cryptographic transformations should be calculated, provided that the required level of crypto-stability is ensured, as well as to ensure the success of operations (such as no decryption errors) and to reduce the computational complexity. A list of general parameters of the encryption algorithms and key encapsulation algorithms of SKELA algorithms is shown in Table 1.

5. Only general parameters are used to calculate additional parameters. Recalculating additional parameters makes it possible to reduce the computational complexity of basic cryptographic transformation operations.

6. The asymmetric key pairs – private key f and public key h – are used for asymmetric encryption and encapsulation of keys. They are calculated on the basis of the use of polynomials G, F, whose degree is *n*, and the coefficients modulo *p (p = 3)*, i.e. take the values 0, -1, 1. The result is as follows: the private key f and the public key h (polynomials) and h (byte task h) calculated simultaneously.

7. In the process of developing encryption algorithms, the NTRU encryption algorithm [4] has been taken into account, which has been successfully tested over time since it has been in use for almost 10 years. But it has a significant drawback that has to do with the possibility of a decryption error. This drawback is absent in the SKELYA algorithm [10].

8. In developing the SKELYA algorithm, the modern practice of using the prime q as a module instead of the number 2k is taken into account, and the polynomial xn – x – 1 instead of the polynomial xn – 1, which provides protection against known attacks. Therefore, field $(Z/q[x]/(x^n - x - 1))$ is used as a field for NTRU Prime [5], but the parameters are calculated taking into account the required levels of crypto-stability $\lambda = \{256, 384, 512\}$ and ensuring that no decryption errors exist.

9. In the encryption algorithm, the following data are used as input: string for encryption (m); line length m (mLen); the recipient's public key h (R / q polynomial) and the corresponding octet string (h). The following data are used is output: sign of Success (OK, ERROR); encrypted string E in case of successful operation.

10. The following data are used as input data in the decryption algorithm: E – a byte string with an encrypted message; f – private key of the recipient; h (R / q – the recipient's public key polynomial) and the corresponding octet string (h). The following data are used as output data: sign of Success (OK, ERROR); open message (in case of successful operation) and length of open message (in case of successful operation).

11. Key data, generated identically as for encryption, are used in the encapsulation algorithm. In fact, encapsulation algorithms use encryption and decryption algorithms of the message defined in advance and indicated by m_kem

12. An arbitrary allowed hash function, that provides a result length of 512 bits, is used as a hash function.

13. The input data in the encapsulation algorithm are the length of the key for symmetric encryption of K_bytes (K_bytes $\leq \lambda / 8$) and the recipient's public key h (polynomial R/q) and the corresponding octet string (h), and the output data are a sign of success (Success = OK, ERROR), Cc encapsulated key and a SKey key for symmetric encryption of K_bytes (K_bytes) length.

14. The input data in the decapsulation algorithm are as follows: symmetric encryption key of K_bytes (K_bytes $\leq \lambda / 8$ length; encapsulated Cc key; recipient private key f and recipient public key h (polynomial R/q) and corresponding octet string (h).

15. The joint use of encryption algorithms and encapsulation of keys makes it possible to produce (calculate) symmetric encryption and authentication keys on communication channels, which in turn makes it possible to exchange the protected information with high speed and ensuring its integrity, confidentiality and cryptographic integrity of such key data.

16. When developing algorithms for encryption and encapsulation of keys, optimization methods of cryptographic transformations based on multiplication of polynomials were applied. To measure the characteristics of time complexity, processor (Intel (R) Core (TM) i5-3.1 GHz) clock cycles were used [9,10]. To compare the obtained complexity data, we used the data [5] (ntruees787ep1 encryption algorithm, which is closer to the characteristics with cryptographic strength $\lambda = 256$).

17. Thus, this article presents the main results and data on encryption algorithms, key encapsulation and their use for encryption and authentication of data on communication channels. They are achieved by means of solving such particular problematic tasks as: calculating general and additional parameters; generating asymmetric pairs of encryption keys and encapsulation; development of asymmetric encryption algorithms (encryption and decryption); development of asymmetric encapsulation algorithms and key decapsulation; developing proposals (algorithms) for calculating the keys of secure communication sessions for their use in symmetric data encryption in communication channels, as well as optimizing and assessing the complexity of direct and inverse transformations during encryption and encapsulation.

18. The main advantages of asymmetric encryption and key encapsulation algorithms are: providing 5 – 7 levels of post-quantum and classical stability; security against special attacks, as well as providing symmetric data encryption on communication channels is the use of block and stream high-speed transformations

References:

1. Lily Chen Report on Post-Quatum Cryptography. NISTIR 8105 (DRAFT) / Lili Chen, Stephen Jordan, Yi-Kai-Liu, Dustin Moody, Rene Peralta, Ray Perlner, Daniel Smith-Tone // Electronic resource. Access mode: http://csrc.nist.gov/publications/drafts/nistir-8105/nistir_8105_draft.pdf.

2. Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process / Gorjan Alagic and others // https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf

3. Gorbenko Yu.I. Methods for construction and analysis of cryptographic systems. Kharkiv : Fort, 2015. 959 p. (In Ukr.).

4. American National Standard X9.98-2010. Lattice-based polynomial public key encryption algorithm, Part 1: key establishment, Part 2: data encryption. 2010.

5. Daniel J. Bernstein NTRU Prime / Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, Christine van Vredendaal // Electronic resource. Access mode: https://ntruprime.cr.yp.to/ntruprime-20160511.pdf. https://bench.cr.yp.to/results-encrypt.html

6. Gorbenko I.D. General statements and analysis of the end-to-end encryption algorithm NTRU Prime IIT Ukraine / I.D. Gorbenko, E.G. Kachko, MV Esina // Radiotekhnika. Kharkov : KNURE, 2018. Is. 193. P. 5-16.

7. Gorbenko I. D. Calculation of general parameters for NTRU Prime Ukraine of 6-7 levels of stability / I. D. Gorbenko, A. N. Alekseychuk, O. G. Kachko, M. V. Yesina, I. V. Stelnik, S. O. Kandy, V. A. Bobukh, V. A. Ponomar // Telecommunications and Radio Engineering, 2019. Vol. 78, Is. 4. P.327-340. DOI: 10.1615/TelecomRadEng.v78.i4.40.

8. Gorbenko I.D. Methods of building general parameters and keys for NTRU Prime Ukraine of 5th–7th levels of stability. Product form / I.D. Gorbenko,O.G. Kachko, Yu.I. Gorbenko, I.V. Stelnik, S.O. Kandyi, M.V. Yesina //

Telecommunications and Radio Engineering, 2019. Vol. 78, Is. 7 P. 579-594. DOI: 10.1615/TelecomRadEng.v78.i7.30.98.

9. CALCULATION OF GENERAL PARAMETERS FOR NTRU PRIME UKRAINE OF 6-7 LEVELS OF STABILITY / I. D. Gorbenko, A. N. Alekseychuk, O. G. Kachko, M. V. Yesina, I. V. Stelnik, S. O. Kandy, V. A. Bobukh,. A. Ponomar . pages 327-340 DOI: 10.1615/TelecomRadEng.v78.i4.40. Vol. 78, 2019 Is. 4.

10. Kachko O., Gorbenko I., Yesina M., Kandiy S. POLYNOMIALS MULTIPLICATION FUNCTIONS FOR ORDINARY AND PRODUCT FORM OF ONE OF THE POLYNOMIALS REPRESENTATION: https://github.com/KandiyIIT/NTRU-POLYNOMIALS-MULTIPLICATION.

11. Ran Canetti, Hugo Krawczyk Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. Electronic resource. Access mode: http://iacr.org/archive/eurocrypt2001/20450451.pdf.

12. Post-Quantum Cryptography. Electronic resource. Access mode: https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions.

13. EUF-CMA and SUF-CMA. Electronic resource. Access mode: https://blog.cryptographyengineering.com/euf-cma-and-suf-cma/.

14 DSTU ISO / IEC 18033-2: 2015 (ISO / IEC 18033-2: 2006, IDT) Information Technology. Methods of Protection. Encryption Algorithms. Part 2. Asymmetric Ciphers. (In Ukr.)

15. Kuznetsov A., Pushkar'ov A., Kiyan N. and Kuznetsova T. Code-based electronic digital signature // 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), Kyiv, Ukraine, 2018, pp. 331-336. DOI: 10.1109/DESSERT.2018.8409154.

16. Kuznetsov A. A., Gorbenko Yu. I., Prokopovych-Tkachenko D. I., Lutsenko M. S., Pastukhov M. V. NIST PQC: Code-Based Cryptosystems // Telecommunications and Radio Engineering. 2019. Vol. 78. Is. 5, pp. 429-441. DOI: 10.1615/TelecomRadEng.v78.i5.50.

17. Gorbenko Y., Svatovskiy I. and Shevtsov O. Post-quantum message authentication cryptography based on error-correcting codes // 2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), Kharkiv, 2016. P. 51-54. DOI: 10.1109/INFOCOMMST.2016.7905333.

Kharkiv National V.N. Karazin University;
JSC "Institute of Information Technologies";

Received 09.08.2019