Application of MATLAB in Practical Teaching of Post-Quantum Cryptography

Aleksei Vambol

***Abstract*—**The paper is aimed at proposing the approach to practical teaching of the basics of the code-based and multivariate classes of post-quantum cryptosystems, which consists in using MATLAB by a learner to perform the main steps of algorithms that make up the considered cryptographic schemes. The practical tasks proposed in the paper are dedicated to the McEliece and Niederreiter ciphers, as well as the Matsumoto-Imai digital signature scheme, since these cryptosystems are classical representatives of their classes.

***Keywords*—**Niederreiter cryptosystem, MATLAB, Matsumoto-Imai cryptosystem, McEliece cryptosystem, post-quantum cryptography, practical teaching.

I. INTRODUCTION

Asymmetric cryptosystems are indispensable tools in the field of modern network security. They serve the following cryptographic purposes [1]:

1. Providing confidentiality of data transmission. Asymmetric ciphers are used in hybrid cryptosystems to transmit symmetric encryption keys over insecure channels.

2. Ensuring authenticity, integrity and non-repudiation of data. Digital signature schemes make possible assuring all of these properties for the signed message.

A list of network security protocols that employ asymmetric cryptography includes, but is not limited to, OpenPGP [2], Tor [3] and TLS [4].

Recent advances in quantum computing pose a potential threat to the most widespread asymmetric cryptosystems, among which RSA, DSA, ECDSA and ElGamal schemes should be mentioned. Each cryptosystem based on the integer factorization problem or the difficulty of a discrete logarithm computation is nonresistant to quantum cryptanalysis [5].

Post-quantum cryptography is a set of cryptographic schemes considered to be invulnerable to cryptanalytic attacks performed using quantum computers [5]. Post-quantum asymmetric cryptosystems, which do not require any hardware that uses quantum properties, are divided into the five classes described in Table I, which was compiled using information from [6].

TABLE I
CLASSES OF POST-QUANTUM CRYPTOSYSTEMS AND THE PROBLEMS ON WHICH THEY ARE BASED

Class	Underlying problem
Code-based cryptosystems	Decoding random linear codes
Multivariate cryptosystems	Solving a system of multivariate polynomial equations over finite fields
Lattice-based cryptosystems	Finding the closest vector in a lattice
Hash-based digital signatures	Finding collisions or preimages in cryptographic hash functions
Isogeny-based cryptosystems	Finding an unknown isogeny between a pair of supersingular elliptic curves

Further development of post-quantum cryptography depends on approaches to education in this field. Practical teaching is of great importance for consolidating a basic understanding of post-quantum cryptosystems.

The current paper is aimed at proposing the approach to practical teaching of the basics of code-based and multivariate cryptosystems, which consists in using MATLAB by a learner to

A. Vambol, National Aerospace University «KhAI», Kharkiv, Ukraine (e-mail: o.vambol@csn.khai.edu)

perform the main steps of algorithms that make up the considered cryptographic schemes. Sections II-IV describe the proposed practical educational tasks on the McEliece, Niederreiter and Matsumoto-Imai cryptosystems, respectively. The first two encryption schemes are the classical representatives of code-based cryptography [6]. The McEliece cipher based on the random binary Goppa codes is a candidate in the second round of the NIST Post-Quantum Cryptography Standardization Process [7]. The Matsumoto-Imai digital signature scheme [8] has been the first multivariate cryptosystem. Although the given cryptographic scheme has been broken, its consideration can be useful for understanding modern multivariate digital signature systems.

II. The McEliece Cipher

A. Task description

Construct a key pair for the McEliece cryptosystem, use it to encrypt and decrypt a randomly chosen message. To build a pair of keys, the linear binary code, which is given in the task, can be used. The scrambling and permutation matrices, as well as the error vector, should be chosen arbitrarily. The report should contain the values of the key pair, initial and encrypted messages, as well as a stepwise description of the performed computations. The computing environment recommended for performing this task is MATLAB.

B. Source data

In this task, the McEliece cryptosystem can be built on top of (n, k) linear binary code with the error-correction capability t and the generator matrix G, where n = 15, k = 5, t = 3 and

$$G = \begin{vmatrix} 1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0 \\ 1\,1\,1\,1\,0\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,0\,0\,1\,0\,0 \\ 1\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,1\,0 \\ 0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,1 \end{vmatrix}.$$

C. Instruction with an example

Key pair generation can be described as follows:

1. Random generation of n × n permutation matrix P [8]. Construction of P is performed by applying a random permutation R to the columns of the identity matrix. The vector R_i is computed as the representation of the permutation inverse to R. The components of R_i are calculated in accordance with the formula $R_i(R(j)) = j$ for all j in [1 .. n].

```
>> R = randperm(15)
>> I = eye(15)
>> P = gf(I(:, R))
>> Ri(R(1:15)) = [1:15]
```

2. Random generation of a nonsingular k × k matrix S [8]. Computing S_i as S^{-1}.

```
>> tmp = [0 1 0 0 1,
          1 1 1 0 1,
          0 0 1 1 0,
          0 0 1 0 0,
          0 1 1 1 0]
>> S = gf(tmp)
>> Si = inv(S)
```

3. Computation of k × n matrix E by the formula $E = S \cdot G \cdot P$ [8].

```
>> G = [1 0 1 0 0 1 1 0 1 1 1 0 0 0 0,
        1 1 1 1 0 1 0 1 1 0 0 1 0 0 0,
        0 1 1 1 1 0 1 0 1 1 0 0 1 0 0,
        1 0 0 1 1 0 1 1 1 0 0 0 0 1 0,
        0 1 0 0 1 1 0 1 1 1 0 0 0 0 1]
>> E = S * gf(G) * P
```

4. The public and private keys are the tuples (E, t) and (S_i, G, R_i), respectively [8].

Encryption consists of the following steps:

1. Representation of the message, which in this task is generated randomly, in the form of k-dimensional vector m [8].

```
>> m = gf([1 0 0 1 1])
```

2. Selection of a random n-dimensional vector z of weight t [8].

```
>> z = gf([0 0 0 1 0 0 0 0 0 1 0 0 0 1 0])
```

3. Computation of the ciphertext c by the formula $c = m \cdot E + z$.

```
>> c = m * E + z
```

Decryption requires the following operations:

1. Calculation of the n-dimensional vector u by means of applying the permutation R_i to the ciphertext vector c [8].

```
>> u = c(Ri)
```

2. Obtainment of the vector v as the result of decoding of u by (n, k) linear binary code with the generator matrix G [8].

```
>> v = decode(double(u.x), 15, 5, 'linear/binary', G)
```

3. Computation of the decrypted message vector d by the formula $d = v \cdot S_i$ [8].

```
>> d = gf(v) * Si
```

The result of decryption should be tested for equality to the initial message.

```
>> isequal(d, m)
```

D. Assessment questions

1. What parameters represent the keys of the McEliece cryptosystem?
2. How does the McEliece cryptosystem provide probabilistic encryption?
3. How do the parameters of the underlying linear binary code determine the sizes of the initial message and ciphertext in the McEliece cryptosystem?
4. What is the generator matrix of a linear code?
5. How are addition and multiplication performed in GF(2)?

III. The Niederreiter Cipher

A. Task description

Construct a key pair for the Niederreiter cryptosystem, use it to encrypt and decrypt a randomly chosen message. To build a pair of keys, the linear binary code, which is given in

the task, can be used. The scrambling and permutation matrices, as well as the error vector, should be chosen arbitrarily. The report should contain the values of the key pair, initial and encrypted messages, as well as a stepwise description of the performed computations. The computing environment recommended for performing this task is MATLAB.

B. Source data

In this task, the Niederreiter cipher can be built on top of (n, k) linear binary code with the error-correction capability t and the parity-check matrix H, where n = 15, k = 5, t = 3 and

$$H = \begin{vmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0 \\ 0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1 \\ 0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,0\,0 \\ 0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\,1 \end{vmatrix}.$$

C. Instruction with an example

Key pair generation can performed in the following way:

1. Random generation of n × n permutation matrix P [8]. Construction of P is performed by applying a random permutation R to the rows of the identity matrix. The vector R_i is computed as a representation of the permutation inverse to R. The components of R_i are found by the formula $R_i(R(j)) = j$ for all j in [1 .. n].

```
>> R = randperm(15)
>> I = eye(15)
>> P = gf(I(R, :))
>> Ri(R(1:15)) = [1:15]
```

2. Random generation of a nonsingular (n - k) × (n - k) matrix S [8]. Computing S_i as S^{-1}.

```
>> tmp = [1 1 0 0 0 0 1 1 0 1
          0 0 0 0 0 1 1 1 1 1
          0 0 0 1 1 1 0 0 0 0
          0 0 1 0 1 1 0 1 0 1
          0 1 1 1 0 1 1 1 1 0
          1 1 1 1 1 0 0 1 0 1
          1 0 0 1 1 0 1 0 1 0
          0 0 1 1 0 0 0 0 0 0
          0 0 1 1 1 1 0 1 0 0
          0 0 1 1 1 1 0 1 1 0]
>> S = gf(tmp)
>> Si = inv(S)
```

3. Computation of (n - k) × n matrix E in accordance with the formula $E = S \cdot H \cdot P$ [8].

```
>> H = [1 0 0 0 0 0 0 0 0 0 1 1 0 1 0
        0 1 0 0 0 0 0 0 0 0 0 1 1 0 1
        0 0 1 0 0 0 0 0 0 0 1 1 1 0 0
```

```
        0 0 0 1 0 0 0 0 0 0 0 1 1 1 0
        0 0 0 0 1 0 0 0 0 0 0 0 1 1 1
        0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1
        0 0 0 0 0 0 1 0 0 0 1 0 1 1 0
        0 0 0 0 0 0 0 1 0 0 0 1 0 1 1
        0 0 0 0 0 0 0 0 1 0 1 1 1 1 1
        0 0 0 0 0 0 0 0 0 1 1 0 1 0 1]
>> E = S * gf(H) * P
```

4. The public and private keys are the tuples (E, t) and (S_i, H, R_i), respectively [8].

Encryption consists of the following steps:

1. Representation of the initial message as n-dimensional vector m of weight t [8]. In this task, the result of this operation is generated randomly.

```
>> m = gf([0 0 1 0 0 0 0 1 0 0 0 1 0 0 0])
```

2. Computation of the ciphertext by the formula $c = E \cdot m^T$ [8].

```
>> c = E * m.'
```

Decryption requires the following operations:

1. Calculation of (n - k)-dimensional vector u in accordance with the formula $u = S_i \cdot c$ [8].

```
>> u = Si * c
```

2. Computation of the variable v as the transposed error vector, which corresponds to the syndrome u in the error correction procedure of the linear binary code with the parity-check matrix H [8]. In the given task, the syndrome decoding table T is used for this purpose. Each row of the this table contains the error vector corresponding to the syndrome, which is the binary representation of i - 1, where i is the index of the given row. This table is determined by the parity-check matrix H.

```
>> T = syndtable(H)
>> i = bi2de(double(u.x.'), 'left-msb')
>> v = T(i + 1, :).'
```

3. Computation of the decrypted message vector d by transposing the result of applying the permutation R_i to the column vector v [8].

```
d = gf(v(Ri).')
```

The result of decryption should be tested for equality to the initial message.

```
>> isequal(d, m)
```

D. Assessment questions

1. What parameters represent the keys of the Niederreiter cryptosystem?
2. How do the parameters of the underlying linear binary code determine the sizes of the initial message and the ciphertext in the Niederreiter cryptosystem?
3. What is the parity-check matrix of a linear code?
4. What is syndrome decoding of a linear code?
5. How are addition and multiplication performed in GF(2)?

IV. The Matsumoto-Imai Digital Signature Scheme

A. Task description

Construct a key pair for the Matsumoto-Imai digital signature scheme, use it to create and verify a digital signature. To build a pair of keys, the central map, which is given in the task, can be used. The initial and final maps, as well as the hash of the signed message, should be selected arbitrarily. The report should contain the values of the key pair, hash and digital signature, as well as a stepwise description of the performed computations. The computing environment recommended for performing this task is MATLAB.

B. Source data

The central map used in the Matsumoto-Imai cryptosystem is an invertible map between n-dimensional vectors over GF(q), the characteristic of which equals 2 [9]. This bijection is built as follows. The extension field $GF(q^n)$ is constructed using some irreducible n-degree polynomial g(z) over GF(q), so each element of $GF(q^n)$ is a polynomial with coefficients in GF(q) and degree less than n. The map w, which is a bijection between n-dimensional vector space over GF(q) and $GF(q^n)$, is defined in the following way [9]:

$$w: (v_0, v_1, ..., v_{n-1}) \to v_{n-1} \cdot z^{n-1} + ... + v_1 \cdot z + v_0.$$

The definition of its inverse w^{-1} is obvious and does not require consideration. The maps m and m^{-1}, which transform elements of $GF(q^n)$, can be described as follows [9]:

$$m: \varepsilon \to \varepsilon^{1+q^a},$$
$$m^{-1}: \varepsilon \to \varepsilon^{b},$$

where a and b are positive integers such that $(1 + q^a) \cdot b \bmod (q^n - 1) = 1$. The central map F and its inverse are defined in the following way [9]:

$$F = w^{-1} \circ m \circ w,$$
$$F^{-1} = w^{-1} \circ m^{-1} \circ w,$$

The map $F: (v_0, ..., v_{n-1}) \to (u_0, ..., u_{n-1})$ possesses a representation in the form of a tuple $(f_0(c_0, ..., c_{n-1}), ..., f_{n-1}(c_0, ..., c_{n-1}))$, whose elements are multivariate quadratic polynomials over GF(q). The foregoing implies that $u_j = f_j(v_0, ..., v_{n-1})$, where $0 \le j \le n - 1$ [9].

In the given task, the central map can be constructed using n = 6, q = 2, a = 4, b = 26 and $g(z) = z^6 + z + 1$. Therefore, $F: (v_0, ..., v_5) \to (u_0, ..., u_5)$ can be represented by the equations over GF(2) as follows:

$$u_5 = v_5 \cdot v_4 + v_5 \cdot v_2 + v_3^2 + v_5^2 + v_2^2 + v_4^2 + v_1^2 + v_3 \cdot v_2 + v_4 \cdot v_1 + v_5 \cdot v_1,$$
$$u_4 = v_4 \cdot v_2 + v_4 \cdot v_0 + v_0 \cdot v_1 + v_3 \cdot v_2 + v_5^2 + v_5 \cdot v_1 + v_5 \cdot v_2 + v_5 \cdot v_3 + v_2 \cdot v_1 + v_5 \cdot v_4 + v_4 \cdot v_1,$$
$$u_3 = v_3 \cdot v_2 + v_2 \cdot v_0 + v_4 \cdot v_2 + v_3^2 + v_2 \cdot v_1 + v_4 \cdot v_1 + v_5 \cdot v_4,$$
$$u_2 = v_2^2 + v_2 \cdot v_0 + v_5 \cdot v_1 + v_4 \cdot v_2 + v_5^2 + v_3 \cdot v_0 + v_4 \cdot v_3 + v_5 \cdot v_0 + v_3 \cdot v_2 + v_2 \cdot v_1 + v_5 \cdot v_3 + v_1^2 + v_3 \cdot v_1,$$
$$u_1 = v_4 \cdot v_0 + v_5 \cdot v_0 + v_5 \cdot v_3 + v_3^2 + v_4 \cdot v_2 + v_1^2 + v_5 \cdot v_2 + v_3 \cdot v_2,$$
$$u_0 = v_3^2 + v_0 \cdot v_1 + v_4 \cdot v_3 + v_2 \cdot v_0 + v_3 \cdot v_0 + v_2 \cdot v_1 + v_5^2 + v_0^2 + v_3 \cdot v_2.$$

The map F^{-1} is defined in the following way:

$$F^{-1} = w^{-1} \circ (\varepsilon \to \varepsilon^{26}) \circ w,$$

where w maps 6-dimensional vectors over GF(2) to the elements of $GF(2^6)$, which has been constructed using the polynomial $z^6 + z + 1$.

C. Instruction with an example

Construction of a key pair can be described as follows:

1. Creation of the initial map $S(x) = x \cdot S_D + S_L$, where S_D is some invertible $n \times n$ matrix and S_L is a random n-dimensional vector [8]. The map $S^{-1}(x)$, which is inverse to the initial one, can be represented as $(x - S_L) \cdot S_I$, where $S_I = S_D^{-1}$. If S_D is singular, the MATLAB function, which inverts matrices, will produce an error message.

```
>> tmp = [1 1 1 0 1 0,
          0 1 0 1 0 0,
          0 0 0 1 0 1,
          1 0 1 0 1 0,
          1 1 0 1 0 0,
          0 0 1 1 0 1]
>> Sd = gf(tmp)
>> Si = inv(Sd)
>> Sl = gf([1 0 0 1 1 0])
```

2. The final map T(x) can be created using the same approach as in the previous step [8]: $T(x) = x \cdot T_D + T_L$, $T^{-1}(x) = (x - T_L) \cdot T_I$, $T_I = T_D^{-1}$.

```
>> tmp = [1 0 0 1 0 1,
          0 1 0 0 1 0,
          0 0 1 0 0 0,
          0 1 0 1 0 0,
          0 0 0 0 1 0,
          0 0 0 1 0 1]
>> Td = gf(tmp)
>> Ti = inv(Td)
>> Tl = gf([0 1 0 0 1 1])
```

3. Representation of the map composition P, which equals $T \circ F \circ S$, in the form of the tuple $(p_0(x_0, ..., x_{n-1}), ..., p_{n-1}(x_0, ..., x_{n-1}))$, whose elements are quadratic polynomials over GF(q). The given map transforms $(x_0, ..., x_{n-1})$ into $(y_0, ..., y_{n-1})$ by the formula $y_i = p_i(x_0,...,x_{n-1})$ [8]. In this step, the symbolic variables, which represent the components of $(x_0, ..., x_{n-1})$, are substituted into the corresponding formulae to obtain the expressions for the components of the tuple representing P. Since GF(2) is a prime field, the aforementioned computations are performed using polynomials with integer coefficients and division modulo 2.

```
>> syms x0 x1 x2 x3 x4 x5
>> tmp = num2cell(mod([x0 x1 x2 x3 x4 x5] * Sd.x + Sl.x, 2))
>> [v0 v1 v2 v3 v4 v5] = tmp{:}
>> u5 = v3^2+v5*v4+v5*v2+v5^2+v2^2+v4^2+v1^2+v3*v2+v4*v1+v5*v1
u4 = v4*v2+v4*v0+v0*v1+v3*v2+v5^2+v5*v1+v5*v2+v5*v3+v2*v1+v5*v4+v4*v1
u3 = v3*v2+v2*v0+v4*v2+v3^2+v2*v1+v4*v1+v5*v4
u2 = v2^2+v2*v0+v5*v1+v4*v2+v5^2+v3*v0+v4*v3+v5*v0+v3*v2+v2*v1+v5*v3+v1^2+v3*v1
u1 = v4*v0+v5*v0+v5*v3+v3^2+v4*v2+v1^2+v5*v2+v3*v2
u0 = v3^2+v0*v1+v4*v3+v2*v0+v3*v0+v2*v1+v5^2+v0^2+v3*v2
>> tmp = num2cell(mod(expand([u0 u1 u2 u3 u4 u5] * Td.x + Tl.x), 2))
>> [p0 p1 p2 p3 p4 p5] = tmp{:}
```

4. The public key is the map composition P, which is represented in the form of the tuple $(p_0(x_0, ..., x_{n-1}), ..., p_{n-1}(x_0, ..., x_{n-1}))$ [8]. The private key, which describes S^{-1}, F^{-1} and T^{-1}, is the tuple $(S_i, S_v, T_i, T_v, b, g(z))$.

Creation of a digital signature requires the following operations:

1. Hashing the message to be signed and converting its digest to the n-dimensional vector h over GF(q) [8]. In this task, h is randomly generated.

```
h = gf([1 1 0 1 0 1])
```

2. Applying T^{-1} to h and saving the result [8].

```
>> tmp = (h - Tl) * Ti
```

3. Applying F^{-1} to the result of the previous step [8]. This operation requires the integer representation of g(z). Since this polynomial is defined over the prime field GF(2) and equals $z^6 + z + 1$, the required value is $1 \cdot 2^6 + 1 \cdot 2^1 + 1$, i.e. 67.

```
>> n = 6, g = 67, b = 26
>> tmp = gf(bi2de(double(tmp.x)), n, g)^b
>> tmp = gf(de2bi(double(tmp.x), n))
```

4. The signature E is computed by applying S^{-1} to the result of the previous operation [8].

```
>> E =(tmp - Sl) * Si
```

Verification of a digital signature can be performed in the following way:

1. Applying P to E and saving the result to J [8]. In this step, the symbolic variables, which represent the components of $(x_0, ..., x_{n-1})$, are assigned the corresponding components of E. The result of this step is obtained as the numerical value of $(p_0(x_0,...,x_{n-1}),...,p_{n-1}(x_0,...,x_{n-1}))$ modulo 2, since the given computations are performed using polynomials with coefficients, which are not elements of GF(2), but integers.

```
>> tmp = num2cell(E.x)
>> [x0 x1 x2 x3 x4 x5] = tmp{:}
>> J = gf(double(mod(subs([p0 p1 p2 p3 p4 p5]),2)))
```

2. Comparison of J with the hash value of the signed message [8]. In this task, the message is not corrupted, so hashing of it yields h. Since J = h, the signature is considered to be valid.

```
>> isequal(J, h)
```

D. Assessment questions

1. What parameters represent the keys of the Matsumoto-Imai digital signature scheme?
2. How do the parameters of the central map determine the sizes of the hash value and digital signature in the Matsumoto-Imai cryptosystem?
3. What properties should the central map possess?
4. What approach can be used to represent a map composition as a tuple of polynomials?
5. How are addition and multiplication performed in GF(2)?

V. CONCLUSION

The proposed task suite is aimed at consolidating a basic understanding of code-based and multivariate cryptosystems through using MATLAB to perform the main steps of their work. These tasks are recommended to be used in conjunction with [8], which is a lecture suite dedicated to an introduction to post-quantum cryptography. Both of these suites of teaching

materials have been developed within the framework of the TEMPUS SEREIN project.

The current work can be extended by applying the proposed approach to teaching of the basics of lattice-based cryptography.

ACKNOWLEDGMENT

The author thanks the team of the TEMPUS SEREIN project (https://serein.eu.org).

REFERENCES

[1] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C, 20th Anniversary Edition*. John Wiley & Sons, 2015, 784 p.

[2] F. Maury, J.-R. Reinhard, O. Levillain, H. Gilbert, "Format Oracles on OpenPGP," in *Proceedings of the Cryptographer's Track at the RSA Conference 2015 (CT-RSA 2015)*, San Francisco, USA, April 20-24, 2015, pp. 220-236.

[3] S. Ghosh, A. Kate, "Post-Quantum Forward Secure Onion Routing (Future Anonymity in Today's Budget)," in *Proceedings of the 13th International Conference on Applied Cryptography and Network Security (ACNS 2015)*, New York, USA, June 2-5, 2015, pp. 263-286.

[4] F. Schillinger, C. Schindelhauer, "End-to-End Encryption Schemes for Online Social Networks," in *Proceedings of the 12th International Conference on Security, Privacy, and Anonymity in Computation, Communication, and Storage (SpaCCS 2019)*, Atlanta, USA, July 14-17, 2019, pp. 133-146.

[5] M. Campagna et al., *ETSI White Paper No. 8. Quantum Safe Cryptography and Security: An introduction, benefits, enablers and challenges*. European Telecommunications Standards Institute, 2015, 64 p.

[6] ETSI ISG QSC, *ETSI GR QSC 001: Quantum-Safe Cryptography (QSC); Quantum-safe algorithmic framework*. European Telecommunications Standards Institute, 2016, 42 p.

[7] G. Alagic et al., *Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process*. National Institute of Standards and Technology, 2019, 27 p.

[8] O. Vambol, "Post-Quantum Cryptoraphy," in V. Kharchenko (ed.), *Secure and resilient computing for industry and human domains. Vol. 1. Fundamentals of security and resilient computing*. Department of Education and Science of Ukraine, National Aerospace University named after N. E. Zhukovsky "KhAI", 2017, 596 p.

[9] J. Ding, "A New Variant of the Matsumoto-Imai Cryptosystem Through Perturbation," in *Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography*, Singapore, March 1-4, 2004, pp. 305-318.