Olha KOROL[1], Alla HAVRYLOVA[2]

Supervisor: Serhii YEVSEIEV[3]

PRAKTYCZNE ALGORYTMY UMAC Z WYKORZYSTANIEM KRYPTO-KODÓW

Streszczenie: W artykule przeprowadzono rozważania o zakresie użytkowania ulepszonego algorytmu UMAC w kryptografii postkwantowej. Rozważany algorytm oparty o formowanie podłoża (substrate) na trzeciej warstwie generowania kodu haszowania za pomocą systemu 'McElise crypto-code' (systemu krypto-kodowania) opartego na kodach eliptycznych. W artykule omawiany jest praktyczny algorytm generowania kodów typu „hash" (hash-kodów) z zastosowaniem kaskadowego hash-kodu UMAC, gdzie dodatkowo zastosowano krypto-kod McElise opartego na kodach eliptycznych. Zastosowanie krypto-kodów umożliwia zachowanie uniwersalności hash-kodu na wyjściu algorytmu, a to z kolei, umożliwia zastosowanie tego podejścia w wielkich bazach danych jako identyfikatora. Dodatkowo, biorąc pod uwagę zastosowanie wielkoskalowych komputerów kwantowych, eksperci organizacji US NIST uważają systemy krypto-kodów jako jedne z najbardziej efektywnych postkwantowych algorytmów kryptograficznych. Takie ujęcia metod kodowania umożliwia wprowadzenie modyfikowanego UMAC do różnorodnych modyfikacji struktur krypto-kodów oraz zapewnia autentyfikację profilów o różnej ważności (sile) oraz długości.

Słowa kluczowe: algorytm haszowania UMAC, budowa krypto-kodów McElice, kody eliptyczne

PRACTICAL UMAC ALGORITHMS BASED ON CRYPTO CODE DESIGNS

Summary: A study was carried out on the use of an improved UMAC algorithm in post-quantum cryptography based on the formation of a substrate on the third layer of the hash code generation by the McElise crypto-code system on elliptic codes. The paper considers a practical algorithm for generating a hash code based on an example implementation of a cascading UMAC hash algorithm with the McElise crypto-code construction on elliptic codes. Using a crypto-code design allows you to save the universality of the hash code at the output of the algorithm, which allows its use in large databases as an identifier. In addition, in the context of

[1] Associate Prof., PhD, Simon Kuznets Kharkiv National University of Economics, a Associate Professor of Department of Cyber Security and Information Technology olha.korol@hneu.net
[2] Simon Kuznets Kharkiv National University of Economics, a Senior Lecturer of Department of Cyber Security and Information Technology Alla.Gavrylova@hneu.net
[3] Senior Research. D.Sc., Simon Kuznets Kharkiv National University of Economics, a head of Department of Cyber Security and Information Technology serhii.yevseiev@hneu.net

the implementation of a full-scale quantum computer, US NIST experts consider crypto-code systems as one of the effective post-quantum cryptography algorithms. This approach allows you to implement the UMAC modification on various modifications of crypto-code structures and to ensure the formation of authentication profiles of different strength and length

Keywords: UMAC hashing algorithm, McElice crypto code constructions, elliptic codes

1. Introduction

An important direction in the development of post-quantum cryptography today is crypto-code systems (designs) (CCS). Their formation is based on the use of algebraic codes disguised as the so-called random code [1], [2]. CCS allow integrally to implement fast cryptographic data conversion and ensure the reliability of the transmitted data based on noise-resistant coding [3], [4]. Despite the advantages, their use in modern software and hardware is hampered by their practical implementation with the required level of cryptographic strength, and withstanding the attack of V.M. Sidelnikov on the basis of linear-fractional transformations allowing to open a private key (generating and / or verification matrix, depending on the McEliece or Niederreiter crypto-code system) [5]. At the same time, according to experts from NIST USA, these crypto-code designs can provide the required level of protection and are able to withstand modern threats. This is confirmed by the participation of the McEliece crypto code construction in the NIST contest for post-quantum cryptography algorithms.

Analysis of the last researches and publications. The development of computing capabilities in recent years, and in the first place, the creation of full-scale quantum computers, has jeopardized the use of classical mechanisms of not only symmetric cryptography, public key cryptography (including algorithms using the theory of elliptic curves), but also algorithms for providing authenticity services based on MDC and MAC codes, specialized hash functions [1], [3], [6], [7]. In the face of modern threats and the use of cryptanalysis algorithms using full-scale quantum computers, the use of the SHA-3 algorithm and the winning algorithms of the NESSIE European cryptographic contest in authentication and digital signature algorithms is questioned because of the possibility of hacking. Under such conditions, an increase in the level of cryptographic stability can lead to an increase in the length of key sequences and a decrease in the rate of cryptographic transformations. The use of the UMAC algorithm with the formation of the substrate of the third layer based on MASH-2 leads to an increase in the level of stability, collisions, but also to a decrease in the conversion speed [8], which is an indirect confirmation of the possibility of reducing the speed of crypto conversions in the conditions of post-quantum cryptography. An important task is to increase the speed of cryptocjnversions while ensuring the required level of cryptographic stability of this algorithm. In [3], [4], practical algorithms for crypto-code constructions are considered, which provide their practical implementation by reducing the power of the alphabet. Their application in the UMAC algorithm will not only provide the required level of cryptographic stability of the generated hash code, but also preserve its versatility.

Research problem – investigation of the possibility of using McEliece crypto-code constructions with elliptic codes (modified, flawed codes) based on a practical example in the UMAC algorithm.

2. Constructing a modified UMAC algorithm using the McEleice CCC

In [9], [10], a mathematical model and a structural diagram of the hash code generation in the UMAC algorithm were considered using, as an algorithm for forming a substrate (a pseudorandom sequence that provides the hash code cryptographic stability), the McEliese crypto code construction using elliptic codes (EC) (modified elliptical codes (MEC), flawed codes). Figure 1 shows a structural diagram of the formation of a modified UMAC algorithm using the McEliece CCC on various algebraic-geometric codes EC, MEC.

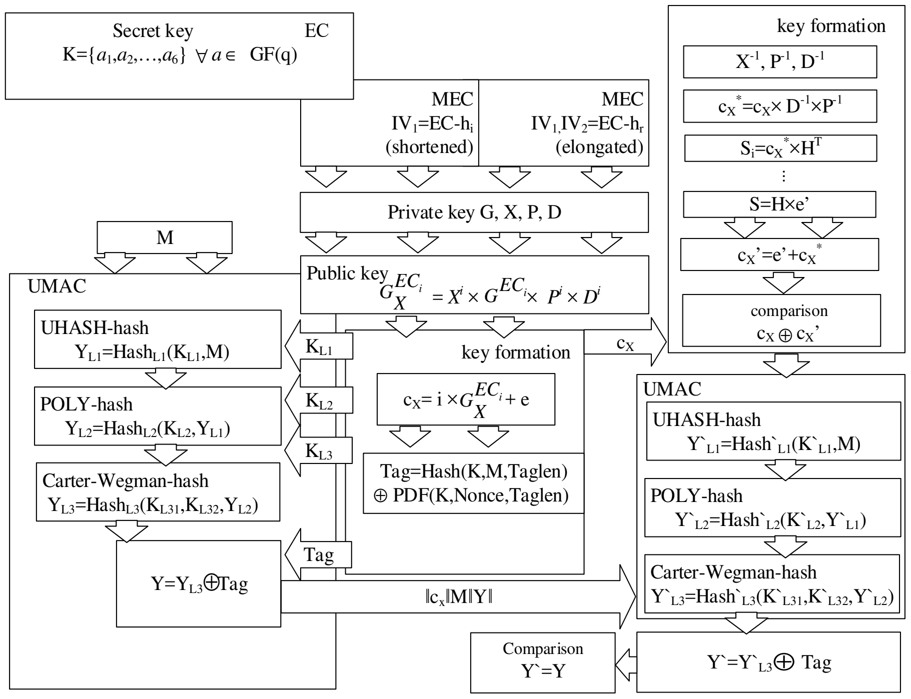

Figure 1. The scheme of transmitting a message from the sender to the recipient and checking the integrity of the received through a comparison of the codograms and hash codes using the CCC McEliece at MEC

The use of various algebraic-geometric and multichannel cryptography codes will allow the formation of various lengths hash code and provide the required level of its cryptographic strength. The main steps for generating a hash code are considered in [10].

Lets consider the practical implementation of the modified UMAC algorithm using the McEliece CCC in the EC using an example.

2.1. Hash code generation in the UMAC algorithm

The creation of a hash for an open message is carried out in parallel with the formation of the codogram, but we will describe the computational transformations according to these actions in sequence.

Table 1. Input data

Y_{L1I}	universal hash-function value (UHASH-hash) of first level hashing
Y_{L3I}	hash-function value (Carter-Wegman-hash) of third level hashing
T	data block
Blocklen	data block length (bytes)
K	secret key
Keylen	secret key length (32 bytes)
Tag	integrity and authenticity control code
K_{L1I}	secret key of the first hashing level, consisting of subkeys K_1, K_2, …, K_n
Y_{L3I}	second-level hash secret key consisting of keys K_{L31} (subkeys K_1, K_2, …, K_n) and K_{L32} (subkeys K_1, K_2, …, K_n)
M	length of the transmitted plaintext array *i*
K'	pseudo random key sequence
Numbyte	pseudo-random key sequence length (number of subkeys)
Index	subkey number
I=11	transmitted plaintext (*k*- bit information vector over *GF*(*q*))
Xor ($\oplus$)	bitwise summation

Implementation:
According to the structural scheme of iterative formation *Y*, *Pad* and *Tag* for an open sender message using an UMAC algorithm [9], [10] we distinguish the following calculation steps:

2.1.1. 1st layer formation

UHASH-hash function value of the first level hashing Y_{L1M} we will calculate by the formula:

$$Y_{L1I} = Hash_{L1}\left(K_{L1I}, I\right) \tag{1}$$

To form K_{L1I}, we will imagine it as a key sequence of four-byte subunits:

$$K_{L1I} = K_{1I} \parallel K_{2I} \parallel \ldots \parallel K_{nI}, \tag{2}$$

where $\parallel$ – concatenation (joining) of strings corresponding to subkeys.
The amount of subkey data depends on the values *Numbyte* and *Blocklen:*

$$n = \left\lceil \frac{Numbyte}{Blocklen} \right\rceil = \frac{1024 + 16 \times 3}{32} = \frac{1072}{32} = 33,5 \approx 33 \Rightarrow i = 1, 2, \ldots, 33.$$

Insofar as $T_i = Index \| i$, then for the first layer *Index* =1, => T_i:

T_1 = 1 || 1 = 00000001 000000001=>K_{1I}
T_2 = 1 || 2 = 00000001 000000010=> K_{2I}
T_3 = 1 || 3 = 00000001 000000011=> K_{3I}
T_4 = 1 || 4 = 00000001 000000100=> K_{4I}
T_5 = 1 || 5 = 00000001 000000101=> K_{5I}
T_6 = 1 || 6 = 00000001 000000110=> K_{6I}
T_7= 1 || 7 = 00000001 000000111=> K_{7I}
T_8 = 1 || 8 = 00000001 000001000=> K_{8I}
T_9 = 1 || 9 = 00000001 000001001=> K_{9I}
T_{10} = 1 || 10 = 00000001 00001010=> K_{10I}
T_{11} = 1 || 11 = 00000001 00001011=> K_{11I}
T_{12} = 1 || 12 = 00000001 00001100=> K_{12I}
T_{13} = 1 || 13 = 00000001 00001101=> K_{13I}
T_{14} = 1 || 14 = 00000001 00001110=> K_{14I}
T_{15} = 1 || 15 = 00000001 00001111=> K_{15I}
T_{16} = 1 || 16 = 00000001 00010000=> K_{16I}
T_{17} = 1 || 17 = 00000001 00010001=>K_{17I}
T_{18} = 1 || 18 = 00000001 00010010=> K_{18I}
T_{19} = 1 || 19 = 00000001 00010011=> K_{19I}
T_{20} = 1 || 20 = 00000001 00010100=> K_{20I}
T_{21} = 1 || 21 = 00000001 00010101=> K_{21I}
T_{22} = 1 || 22 = 00000001 00010110 => K_{22I}
T_{23} = 1 || 23 = 00000001 00010111 => K_{23I}
T_{24} = 1 || 24 = 00000001 00011000 => K_{24I}
T_{25} = 1 || 25 = 00000001 00011001=> K_{25I}
T_{26} = 1 || 26 = 00000001 00011010 => K_{26I}
T_{27} = 1 || 27 = 00000001 00011011=> K_{27I}
T_{28} = 1 || 28 = 00000001 00011100 => K_{28I}
T_{29} = 1 || 29 = 00000001 00011101=> K_{29I}
T_{30} = 1 || 30 = 00000001 00011110=> K_{30I}
T_{31} = 1 || 31 = 00000001 00011111 => K_{31I}
T_{32}= 1 || 32 = 00000001 00100000=> K_{32I}
T_{33} = 1 || 33 = 00000001 00100001=> K_{33I}

Based on the length *M* of input message (*M*=3 bytes), amount of blocks *T*=1, therefore, the number of subkeys on this layer is the same. Wherein $K_{L1I} = T_1 = 0000000100000001$.

The hash-code values of this layer are calculated using the following formula:

$$Y_{L1I} = (I + K_{L1I}) \bmod 32 \quad (3)$$

$$Y_{L1I} = (0100110+10000001)\text{mod}32 = 111$$

2.1.2. 2-nd layer formation

Since the length of *M* is less than 1,024 bytes, this level of hashing will not be performed, and we will perform calculations using the hash code of the third level.

2.1.3. 3-rd layer formation

Number of subkeys for K_{L31} and K_{L32} also depends on the values $Numbyte$ and *Blocklen.*

Formation of K_{L31I}

Number of subkeys for K_{L31I} :

$$n = \left\lceil \frac{Numbyte}{Blocklen} \right\rceil = \frac{64\times 4}{32} = 8 \;\; \Rightarrow i = 1,2,3,4,5,6,7,8$$

Therefore, to form K_{L31I} we will imagine it as a key sequence of eight four-byte subunits:

$$K_{L31I} = K_{1I} \,\|\, K_{2I} \,\|\, K_{3I} \,\|\, K_{4I} \,\|\, K_{5I} \,\|\, K_{6I} \,\|\, K_{7I} \,\|\, K_{8I} \quad (4)$$

For the third layer at *Index* =3, => T_i:

T_1 = 3 || 1 = 00000011 00000001 => K_{1I}
T_2 = 3 || 2 = 00000011 00000010 => K_{2I}
T_3 = 3 || 3 = 00000011 00000011 => K_{3I}
T_4 = 3 || 4 = 00000011 00000100 => K_{4I}
T_5 = 3 || 5 = 00000011 00000101 => K_{5I}
T_6 = 3 || 6 = 00000011 00000110 => K_{6I}
T_7 = 3 || 7 = 00000011 00000111 => K_{7I}
T_8 = 3 || 8 = 00000011 00001000 => K_{8I}

Formation of K_{L32I}

Number of subkeys for K_{L32I}:

$$n = \left\lceil \frac{Numbyte}{Blocklen} \right\rceil = \frac{4 \times 4}{32} = 0,5 \approx 1 \Rightarrow i = 1$$

To form K_{L32I} we will imagine it as a key sequence of 1 four-byte subunit:

$$K_{L32I} = K_{1I} \tag{5}$$

For the third layer at *Index*=4, => T_i:

T_i = 4 || 1 = 00000100 00000001 => K_{1I}

The hash-code value of the third layer is calculated using the following formula:

$$Y_{L3I} = ((Y_{L1I} \bmod (2^{36} - 5)) \bmod 2^{32}) xor Y_{L32I} =$$
$$((I + K_{1I}) \bmod 32) \bmod (2^{36} - 5)) \bmod 2^{32}) xor Y_{L32I} \tag{6}$$

$$Y_{L3I} = ((11 \bmod (2^{36} - 5)) \bmod 2^{32}) xor 00000100\ 00000001 = 10000000010 \tag{7}$$

2.2.1. Formation of the Pad substrate

Input data:

$x^3+y^2z+yz^2=0$	algebraic curve over a field $GF(2^2)$
K	secret key
Keylen	secret key length (32 bytes)
e=00100200	secret error vector weight $w_h(e) \le t = \left\lceil \frac{d-1}{2} \right\rceil$
$X = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$	nondegenerate $k \times k$ matrix
$P = \begin{bmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \end{bmatrix}$	permutation matrix of size $n \times n$

$D=\begin{bmatrix}1&0&0&0&0&0&0&0\\0&2&0&0&0&0&0&0\\0&0&3&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&2&0&0&0\\0&0&0&0&0&3&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}$	diagonal matrix equal 1
$G=\begin{bmatrix}2&2&3&0&1&3&0&1\\3&3&2&1&0&2&1&0\end{bmatrix}$	generating matrix
I=11	transmitted plaintext (*k*- bit information vector over $GF(q)$)

Points of an algebraic curve:

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
X	0	0	0	1	2	3	1	2	3
Y	1	0	1	2	2	2	3	3	3
Z	0	1	1	1	1	1	1	1	1

The formation of the substrate on the CCC:

1) Find the public key, which in the McEliece cryptosystem is the matrix [3]:

$$G_X^{EC}=X\times G\times P\times D \tag{8}$$

$$G_X^{EC}=\begin{bmatrix}1&2\\3&0\end{bmatrix}\times\begin{bmatrix}2&2&3&0&1&3&0&1\\3&3&2&1&0&2&1&0\end{bmatrix}\times\begin{bmatrix}0&1&0&0&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0\end{bmatrix}\times$$

$$\times\begin{bmatrix}1&0&0&0&0&0&0&0\\0&2&0&0&0&0&0&0\\0&0&3&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&2&0&0&0\\0&0&0&0&0&3&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}=\begin{bmatrix}2&1&3&0&1&1&1&0\\0&2&2&2&2&0&3&2\end{bmatrix}$$

2) A cryptogram (codogram) is a *n*-length vector, which is calculated by the following formula:

$$C_X = I \times G_X^{EC} + e \, , \tag{9}$$

where vector $I \times G_X^{EC}$ is the code word of masked code, i.e. belongs to (*n*,*k*,*d*)-code with generating matrix G_X^{EC} ; vector *e* is a one-time session secret key.

$$C_X = 11 \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \oplus 00100200 = 23023322$$

3) Cipher of generated cryptogram 23023322 is transmitted to the transmission channel to the recipient.

2.2.2. The formation of a pseudo-random lining (substrate) using the PDF function

To ensure the cryptographic stability of the UMAC algorithm at the level of stability of the used cryptographic algorithm, we form a pseudo-random lining Pad_{Cx} for *I* using *PDF* function:

$$Pad = PDF(K, Nonce, Taglen) \tag{10}$$

Input data:

K=0106	secret key
Keylen	secret key length (4 bytes)
Taglen	the length of the integrity control code (authenticity) Pad_{Cx} (4 bytes)
Nonce	unique number for input message *I* (8 bytes)
Numbyte	subkey length (equal to *Keylen)*
Index	subkey number (0)
Cx=23023322	cryptogram

According to the pseudo-random *Pad* lining formation procedure for *I*, it is necessary to form the following subkey, presented as a function *KDF* [8–10]:

$$K' = KDF(K, Index, Numbyte) \tag{11}$$

$$K' = KDF(0106, 0, 4)$$

Pseudo-random *Pad* lining will look like:

$$Pad = PDF(0106, 8, 4) = 1101010$$

As a result of the substrate formation, various parts of it can be used as an additional initialization vector.

3. Hash code verification at the receiver using the UMAC algorithm

Generation of message authentication codes is possible according to the formula [9, 10]:

$$Tag \ = UMAC\left(K, I, Nonce, Taglen\right) = Hash(K, I, Taglen) \oplus$$
$$\oplus PDF(K, Nonce, Taglen) = Y_{L3M} \oplus Pad \quad (12)$$

$$Tag = 10000000010 \oplus 1101010 = 10001101100 \quad (13)$$

To generate a summary code of the reliability of the transmitted text, we will use the found value of the hash code Y_{L3M} (7) and authentication code for codogram *Tag* (13) sender's plaintext:

$$Y = Y_{L3M} \oplus Tag \quad (14)$$

$$Y = 10000000010 \oplus 10001101100 = 1101110 = 110_{10} \quad (15)$$

3.1. Finding the error vector from the C_X^* cryptogram

To verify the received message based on the verification of hash codes on the receiving side, the authorized user needs to calculate the sender's session key (error vector) to form a pseudo-substrate *Tag*.

Authorized user who knows the secret key *K*, getting a cryptogram C_X^*, starts decoding it. Input data:

C_X=*23023322*	cryptogram
K=0106	secret key
$P^{-1} = \begin{bmatrix} 0&0&0&1&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0 \end{bmatrix}$	matrix inverse to the permutation matrix P(since its determinant is equal to 1, $P^{-1} = P^T$)

$D^{-1}=\begin{bmatrix}1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0\\0&0&2&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&3&0&0&0\\0&0&0&0&0&2&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}$	matrix inverse to the diagonal matrix D – unipotent matrix (square matrix, all eigenvalues are equal to 1), which saves weight by Hamming of *e* vector
$X^{-1}=\begin{bmatrix}0&2\\3&1\end{bmatrix}$	matrix inverse to a non-degenerate matrix X

Implementation:

1) we construct a vector, which is a code word of a code with a generating matrix *G*, distorted by no more than *t* bits:

$$C_X^* = C_X \times D^{-1} \times P^{-1}, \tag{16}$$

$$C_X^* = 23023322\times\begin{bmatrix}1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0\\0&0&2&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&3&0&0&0\\0&0&0&0&0&2&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}\times\begin{bmatrix}0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&1&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&1&0&0\end{bmatrix}=22202221$$

2) we get an error syndrome:

$$S = C_X^* \times H^T \tag{17}$$

$$S = 22202221\times\begin{bmatrix}1&1&1&1&1&1&1&1\\0&0&1&2&3&1&2&3\\0&1&2&2&2&3&3&3\\0&0&1&3&2&1&3&2\\0&0&2&3&1&3&1&2\\0&1&3&3&3&2&2&2\end{bmatrix}$$

S_{00}= 1
S_{10}= 2+1+2+3+3=1
S_{01}= 2+3+3+1+1+3=1
S_{20}= 2+3+2+1+2=0
S_{11}= 3+2+1+2+2=0
S_{02}= 2+1+1+3+3+2=0
S(1,1,1,0,0,0)

3) we find the polynomial of the error locator ($\Lambda(x)$),based on its general presentation:

$$\Lambda(x) = a_{00} + a_{10}x + y = 0 \tag{18}$$

$$\begin{bmatrix}S_{00}&S_{10}\\S_{10}&S_{20}\end{bmatrix}\times\begin{bmatrix}a_{00}\\a_{01}\end{bmatrix}=\begin{bmatrix}S_{01}\\S_{11}\end{bmatrix}\ldots\begin{bmatrix}1&1&1\\1&0&0\end{bmatrix}\Rightarrow a_{00}=0;\ a_{10}=1$$

$\Lambda(\mathrm{xy}) = x + y = 0$ - polynomial error locators

4) we localize the error according to the Chen procedure:

$P_1(0,0,1)$ Λ (x,y) =0+0=0 – error
$P_2(0,1,1)$ Λ (x,y) =0+1=1
$P_3(1,2,1)$ Λ (x,y) =1+2=3
$P_4(2,2,1)$ Λ (x,y) =2+2=0 – error
$P_5(3,2,1)$ Λ (x,y) =3+2=1
$P_6(1,3,1)$ Λ (x,y) =1+3=2
$P_7(2,3,1)$ Λ (x,y) =2+3=1
$P_8(3,3,1)$ Λ (x,y) =3+3=0 – error

5) imagine the error in the form of a vector with indications of erroneous positions:

$$e' = e_1 00\, e_6 000 e_8$$

6) finding the multiplicities e_1, e_4 and e_8, solving a system of equations by the formula:

$$S' = H \times e' \qquad (19)$$

$$e' = 00020003 \qquad (20)$$

7) as a result, we get a cryptogram $C_X^{'}$ taking into account the error vector (20):

$$C_X^{'} = C_X^{*} + e' = 00020003 \oplus 22202221 = 22222224 \qquad (21)$$

This codeword is used as the basis for the formation of the substrate according to the UMAC algorithm.

3.2. Hash-codes verification

An authorized user (recipient) generates a hash code using expressions (1) - (15). Verification is carried out by comparison, received from the sender and the generated hash codes. If they coincide, a decision is made that the plaintext received through the open channel is not modified.

4. Conclusions

As a result of the research, practical algorithms for generating a hash code and its verification based on the UMAC algorithm using McEliece crypto code constructions on the EC were developed. This mechanism of message authenticity can be used not only on elliptic codes, but also on modified (shortened, and / or elongated) elliptic codes, as well as on flawed codes using hybrid crypto-code constructions. This approach allows the practical implementation of a fast hashing algorithm with a level of strength in post-quantum cryptography.

REFERENCES

1. BERNSTEIN D., BUCHMANN J., DAHMEN E.: Post-Quantum Cryptography. – 2009, Springer-Verlag, Berlin-Heidleberg. – 245 p.
2. KUZNETSOV A. A., PUSHKAREV A. I., SVATOVSKIY I. I., SHEVTSOV A. V.: Несимметричные криптосистемы на алгебраических кодах для постквантового периода / A. A. Kuznetsov, A. I. Pushkarev, I. I. Svatovskiy, A. V. Shevtsov // Radiotehnika. – 2016. – Vyp. 186. – P. 70 – 90.
3. HRYSHCHUK R., YEVSEIEV, S. SHMATKO A.: Construction methodology of information security system of banking information in automated banking systems: monograph, 284 p., Vienna.: Premier Publishing s. r. o., 2018.
4. YEVSEIEV S.: The development of the method of multifactor authentication based on hybrid crypto-code constructions on defective codes / S Yevseiev, H Kots, S Minukhin [and other] //. East European Advanced Technology Journal, – 2017, T. 5/9(89). – P. 19 – 35.
5. СИДЕЛЬНИКОВ В.М.: Криптография и теория кодирования. Материалы конференции «Московский университет и развитие криптографии в России», МГУ. – 2002. – 22 с.
6. Final report of European project number IST-1999-12324, named New European Schemes for Signatures, Integrity, and Encryption, April 19, 2004 – Version 0.15 (beta), Springer-Verlag
7. Status Report on the First Round of the SHA-3 Cryptographic Hash Algorithm Competitionhttp Andrew Regenscheid, Ray Perlner, Shu-jen Chang, John Kelsey
8. KOROL O., PARHUTS L., YEVSEIEV S.: Razrabotka modeli i metoda kaskadnogo formirovaniya MAC s ispolzovaniyem modelyarnyh preobrazovaniy / Olha Korol, Lubomir Parhuts, Serhii Yevseiev // Zahyst informacii. – T. 15. – №3. – lypen-veresen 2013. – P. 186 – 196.
9. YEVSEIEV S., OLHA K., HAVRYLOVA A.: Development of authentication codes of messages on the basis of UMAC with crypto-code McEliece’s scheme on elliptical codes / Serhii Yevseiev, Olha Korol Alla Havrylova // Materials of VIIth International Scientific and Technical Conference "Information protection and information systems security": report theses, May 30 –31, 2019. – Lviv: Lviv Polytechnic Publishing House, 2019. – 1 electron. opt. disk (DVD). – C. 86 – 87.
10. HAVRYLOVA A., OLHA K., YEVSEIEV S.: Development of authentication codes of messages on the basis of UMAC with crypto-code McEliece’s scheme. International Journal of 3D printing technologies and digital industry 3:2 (2019). P.153–170.