Practical UMAC Algorithm on Hybrid Crypto-Code Constructions of McElise on Shortened Mec

Alla Havrylova

March 22, 2020

4th International Congress on 3D Printing (Additive Manufacturing) Technologies and Digital Industry 2019
11 - 14 APRİL, 2019, ANTALYA, TR

PRACTICAL UMAC ALGORITHM ON HYBRID CRYPTO-CODE CONSTRUCTIONS OF McELISE ON SHORTENED MEC

A. HAVRYLOVA[a*]

[a*]Simon Kuznets Kharkiv National University of Economics, Department of Cyber Security and Information Technology, UKRAINE

[*]*Corresponding Author:* alla.gavrylova@hneu.net

ABSTRACT

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 hybrid 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 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 hybrid crypto-code structures and to ensure the formation of authentication profiles of different strength and length.

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

1. INTRODUCTION

An important direction in the development of post-quantum cryptography today is crypto-code systems (constructions) (CCC). Their formation is based on the use of algebraic codes disguised as the so-called random code [1], [2]. CCC allow integrated 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 stability, 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 crypto-code system of McElice or Niederreiter) [5]. At the same time, according to experts of 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 McElice crypto code construction in the NIST contest for post-quantum cryptography algorithms. It seems interesting to explore the possibilities of sharing the already known cryptographic coding systems for transmitting information.

2. LITERATURE REVIEW

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 speed 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 cryptographic transformations in the conditions of post-quantum cryptography. An urgent task is to increase the speed of cryptocurrencies while ensuring the required level of cryptographic stability of this algorithm. In [3], [4], practical algorithms for crypto-code constructions are considered that 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 hybrid McElice crypto-code constructions with shortened flawed elliptic codes based on a practical example in the UMAC algorithm.

3. CONSTRUCTION OF A MODIFIED UMAC ALGORITHM USING HCCC ON THE BASIS OF MKKS McELICE FOR A SHORTED MEC

In works [9], [10], a mathematical model and a structural diagram of the hash code generation in the UMAC algorithm were considered using, as an algorithm, a substrate (pseudo-random sequence that ensures the hash code cryptographic stability) of the McElise crypto code design using elliptic codes (EC) (modified elliptical codes (MEC), flawed codes).
The use of various algebraic and multi-channel cryptography codes will allow the formation of various hash code lengths and provide the required level of its cryptographic strength. The basic steps of creating a hash code are considered in the work [10].
Consider the practical implementation of the modified UMAC algorithm using the McElice HCCC in the EC using an example. The input to the calculations is:

Y_{L1I}	universal hash value (UHASH-hash) of the first level of hashing
Y_{L3I}	hash value (Carter-Wegman-hash) of the third level of 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 level of hashing, consisting of subkeys $K_1, K_2, \ldots, K_n$
K_{L3I}	second-level hash secret key consisting of keys K_{L31} (subkeys $K_1, K_2, \ldots, K_n$) and K_{L32} (subkeys $K_1, K_2, \ldots, K_n$)
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
$x^3+y^2z+yz^2=0$	algebraic curve over the field *GF (22)*
e=00000200	secret weight error vector $w_h(\mathrm{e}) \le \mathrm{t} = \left[\frac{d-1}{2}\right]$
$X = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$	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
Taglen	the length of the integrity control code (authenticity) *PadCx* (4 bytes)
Nonce	unique number for input message *I* (8 bytes)
Numbyte	subkey length (equal to *Keylen*)
Index	subkey number (0)
Cx=23023322	cryptogram
$P^{-1}=\begin{bmatrix}0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&1&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&1&0&0\end{bmatrix}$	matrix inverse to the permutation matrix (since its determinant is 1, then $P^{-1}=P^T$)
$D^{-1}=\begin{bmatrix}1&0&0&0&0&0&0&0\\0&3&0&0&0&0&0&0\\0&0&2&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&3&0&0&0\\0&0&0&0&0&2&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}$	the inverse of the diagonal matrix D – is a unipotent matrix (a square matrix, all eigenvalues are 1), which preserves the Hamming weight of the vector *e*
$X^{-1}=\begin{bmatrix}0&2\\3&1\end{bmatrix}$	matrix inverse of a non-degenerate matrix X

Algebraic curve points:

	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

3.1. Hash code generation in the algorithm UMAC

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. According to the block diagram of the iterative formation of Y, Pad, and Tag for an open message from the sender using the UMAC algorithm [9], [10], we distinguish the following calculation steps.

3.1.1. 1st layer formation

The value of the first level hash function UHASH-hash Y_{L1I} we will calculate by the formula:

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

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

$$K_{L1I} = K_{1I} \,\|\, K_{2I} \,\|\, ... \,\|\, K_{nI},$$

where || – is the concatenation (joining) of the strings corresponding to the subkeys.

The amount of subkey data depends on the values *Numbyte* and *Blocklen*:

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

Because the $T_i = Index \| i$, 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_{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_{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_{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 the input message (*M* = 3 bytes), the number of blocks is *T* = 1, therefore, the number of subkeys on this layer is the same. Wherein $K_{L1I} = T_1 = 0000000100000001$.
The hash values of this layer are calculated using the following formula:

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

3.1.2. 2nd layer formation.
Since the length of *M* is less than 1024 bytes, this level of hashing will not be performed, and we will perform calculations using the hash code of the third level.

3.1.3. 3rd layer formation.
Number of subkeys for K_{L31} and K_{L32} also depends on the values *Numbyte* and *Blocklen*.
Number of subkeys for K_{L31I}:

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

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

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

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}

Number of subkeys for K_{L32I}:

$$n = \left[\frac{Numbyte}{Blocklen}\right] = \frac{4 \times 4}{32} = 0,5 \approx 1 => i = 1$$

To form K_{L32I} imagine it as a key sequence of 1 four-byte sub-block:

$$K_{L32I} = K_{1I}$$

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

$$T_i = 4 \parallel 1 = 00000100\ 00000001 \Rightarrow K_{1I}$$

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

$$Y_{L3I} = ((Y_{L1I} \bmod (2^{36} - 5)) \bmod 2^{32}) xor Y_{L32I} =$$
$$((I + K_{1I}) \bmod 32) \bmod (2^{36} - 5)) \bmod 2^{32}) xor Y_{L32I}$$
$$Y_{L3I} = ((11 \bmod (2^{36} - 5)) \bmod 2^{32}) xor 00000100\ 00000001 = 10000000010$$

3.2.1. Pad Shaping

1) The recipient generates a public key, which in the McElice cryptosystem is the matrix [3]:

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

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

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

2) The cryptogram (codogram) formed from the information message *I* is a vector of length n, which is calculated by the following formula:

$$C_X^* = I \times G_X^{MEC} \oplus e,$$

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

$$C_X^* = 11 \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \oplus 00000200 = 23023322$$

3) We form the initialization vector IV = 00100000 for the recipient and sender. This vector shows the location of the code sequence reduction.:

$$C_X^* = 2323322$$

4) Damage to the initial text based on the conversion Table 1.

Table 1. Damage

Word (shuffled)	**Residue length**	***C(x)***	***F(x)***
000	2	00	1
001	2	01	1
010	2	10	1
011	2	11	1
100	2	00	0
101	2	01	0
110	2	10	0
111	2	11	0

Initial text (word): $C_x^* = 2323322_{10}$ = 010 011 000 010 011 011 010 010_2.

5) Sending damage (flag) by the first channel to the recipient, sending the flawed code (balance) by the second channel to the recipient.

We get $C_x^* = 1011001011111010_2$

Convert to decimal notation: 545750_{10} – enters the first channel.

Flags received $F(x)=111111111_2$.

When converted to decimal, we get: 777_{10} – enters the second channel.

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

To ensure the cryptographic stability of the UMAC algorithm at the level of stability of the used cryptographic algorithm, we form a *PadCx* pseudo-random pad for I using the function PDF:

$$Pad = PDF(K, Nonce, Taglen)$$

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

$$K' = KDF(K, Index, Numbyte)$$

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

Pseudo-random lining Pad will have the form:

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

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

4. HASH CODE VERIFICATION AT THE RECEPTION SIDE USING AN ALGORITHM UMAC

4.1. Generating a validity code for a received message

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

$$Tag = UMAC\left(K, I, Nonce, Taglen\right) = Hash(K, I, Taglen) \oplus$$
$$\oplus PDF(K, Nonce, Taglen) = Y_{L3M} \oplus Pad$$
$$Tag = 10000000010 \oplus 1101010 = 10001101100$$

To generate a summary code of the reliability of the transmitted text, we will use the found value of the hash code Y_{L3M} and code authentication code Tag plaintext sender:

$$Y = Y_{L3M} \oplus Tag$$
$$Y = 10000000010 \oplus 10001101100 = 1101110 = 110_{10}$$

4.2. Decoding a received message

1) Recover received text

The resulting values from two channels are translated into a binary number system:

$$C_x^* = 545750_{10} = 1011001011111010_2$$
$$F(x) = 777_{10} = 111111111_2$$

2) Loss recovery

Using Table 1 we get the code word:

$$C_x^* = 010\ 011\ 000\ 010\ 011\ 011\ 010\ 010_2 = 2323322_{10}$$

3) To restore closed text, the recipient adds null information characters to the location indicated by the initialization vector *IV:*

$$C_x^* = 2323322 \rightarrow 23023322$$

4) With recovered closed text C_x remove the action of secret permutation and diagonal matrices:

$$C_x^* = C_x \times D^{-1} \times P^{-1}$$

$$C_x^* = 23123322 \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} =$$

$= 22102221$

$$C_x^* = 22102221$$

5) We find the syndrome and polynomial of error locators:

$$S = C_x^* \times H^{EC^T},$$

$$S = 22102221 \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 & 3 & 1 & 2 & 3 \\ 0 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 0 & 0 & 1 & 3 & 2 & 1 & 3 & 2 \\ 0 & 0 & 2 & 3 & 1 & 3 & 1 & 2 \\ 0 & 1 & 3 & 3 & 3 & 2 & 2 & 2 \end{bmatrix}$$

We find a syndrome:

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

$$S = (1,1,1,0,0,0);$$

Find the polynomial of error locators $\Lambda(x) = a_{00}+a_{10}x+y = 0$

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

$\Lambda(xy) = x+y= 0$ – error locator polynomial

6) We find error locators according to Chen's procedure:

$P_1(0,0,1)$ $\Lambda(x,y)$ =0+0=0 – error

$P_2(0,1,1)$ $\Lambda(x,y)$ =0+1=1

$P_3(1,2,1)$ $\Lambda(x,y)$ =1+2=3

$P_4(2,2,1)$ $\Lambda(x,y)$ =2+2=0 – error

$P_5(3,2,1)$ $\Lambda(x,y)$ =3+2=1

$P_6(1,3,1)$ $\Lambda(x,y)$ =1+3=2

$P_7(2,3,1)$ $\Lambda(x,y)$ =2+3=1

$P_8(3,3,1)$ $\Lambda(x,y)$ =3+3=0 – error

$$e^* = e_1 00 e_4 000 e_8$$

We find: $e^* \times H^{EC^T} = S$, solving the system of equations, we obtain: $e_1 = 0$, $e_4 = 2$, $e_8 = 3$

$$e^* = 00020003$$

We find $i^* = e^* + C_x^*$
$i^* = 00020003 \oplus 22102221 = 22$;

7) Find plain text:

$$i = i^* \times X^{-1},\ i = 22 \times \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} = 11.$$

4.3. Hash verification

The authorized user (recipient) generates in accordance with paragraph. 3.1 - 4.1 hash code. Verification is carried out by comparison, received from the sender and generated by the recipient of the hash codes. If they coincide, a decision is made that the plaintext received through the open channel is not modified.

5. CONCLUSION

As a result of the research, practical algorithms for generating a hash code and its verification based on the UMAC algorithm using the McElice hybrid crypto-code constructions on the MEC were developed. This mechanism of message authenticity can be used not only on defective shortened codes, but also on elongated ones. This approach can significantly increase the relative data transfer rate, which will positively affect the practical implementation of a fast hashing algorithm with a given level of strength in post-quantum cryptography.

REFERENCES

1. Black, J., Halevi, S., Krawczyk, H., Krovetz, T. and P. Rogaway, "UMAC: Fast and provably secure message authentication", Advances in Cryptology, CRYPTO '99, LNCS, Vol. 1666, Pages 216-233, 1999.

2. Krovetz, T. and Rogaway, P., "Fast universal hashing with small keys and no preprocessing, work in progress", http://www.cs.ucdavis.edu/~rogaway/umac, October 12, 2000.

3. Krovetz, T., Black, J., Halevi, S., Hevia, A., Krawczyk, H. and Rogaway, P., "UMAC -Message authentication code using universal hashing. IETF Internet Draft, draft-krovetz-umac-01.txt.", http://www.cs.ucdavis.edu/~rogaway/umac, November 15, 2000.

4. Krovetz T., "UMAC-Message authentication code using universal hashing. IETF Internet Draft, draft-krovetz-umac-02.txt.", http://www.cs.ucdavis.edu/~rogaway/umac, February 2, 2004.

5. "Final report of European project number IST-1999-12324, named New European Schemes for Signatures, Integrity and Encryption", Version 0.15 (beta), Springer-Verlag, April 19, 2004.

6. Krovetz T., "UMAC-Message authentication code using universal hashing", http://www.cs.ucdavis.edu/~rogaway/umac, June 23, 2006.

7. Krovetz T., "Software-Optimized Universal Hashing and Message Authentication. Dissertation submitted in partial satisfaction of the requirements for the degree of doctor of philosophy", University Of California Davis, California, September 2000.

8. Carter, J. L. and Wegman, M. N., "Universal classes of hash functions", Computer and System Scince, No. 18, Pages 143–154, 1979.

9. Wegman, M. N. and Carter, J. L., "New hash functions and their use in authentication and set equality", Computer and System Scince, No. 22, Pages 265–279, 1981.

10. Koro, Olha, Havrylova, Alla and Yevseiev Serhii “Practical UMAC algorithms based on crypto code designs”, Przetwarzanie, transmisja I bezpieczenstwo informacji. Bielsko-Biala: Wydawnictwo naukowe Akademii Techniczno-Humanistycznej w Bielsku-Bialej, Tom 2, Pages 221-232, 2019.

11. Korol, O. G. and Yevseiev, S. P., “The method of universal hashing on the basis of modular transformations, Information processing systems”, Information Technology and Computer Engineering, No. 7(97), Pages 131–132, 2011.

12. Korol, O. G., Yevseiev, S. P. and Dorokhov, A. V., “Mechanisms and protocols for protecting information in computer networks and systems”, Scientific Journal of the Ministry of Defense of Republic of Serbia. Military Technical Gazette, Belgrade, No. 4, Pages 15–30, 2011.

13. Korol, O.G. and Yevseiev, S. P., “Results of the statistical test security hash algorithms-candidates tender to select standard hash algorithm SHA-3”, News of higher technical educational institutions of Azerbaijan, No. 2, Pages 73–78, 2012.

14. Regenscheid, Andrew, Perlner, Ray, Chang, Shu-jen, Kelsey, John, Nandi, Mridul and Paul, Souradyuti, “Status Report on the First Round of the SHA-3 Cryptographic Hash Algorithm Competition”, http://www.nist.gov/index.html, March 3, 2005.

15. Chung-Wei Phan Raphael, “Mini Advanced Encryption Standard (Mini-AES): A testbed for Cryptanalysis Students”, Cryptologia, XXVI (4), Pages 283–306, 2002.

16. A Description of Baby Rijndael, ISU CprE/Math 533; NTU ST765-U, 2003.

17. Lisitskaya, I. V., Grinenko, T. A. and Bessonov, S. Yu., “Analysis of the differential and linear properties of ciphers rijndael, serpent, threefish with 16-bit inputs and outputs”, East European Journal of Advanced Technologies, Pages 50-54, 2015.

18. Yevseiev, S. P., Ostapov, S. E. and Korolev, R. V., “Use of mini-versions for evaluation of the stability of block-symmetric ciphers”, Scientific and Technical Journal “Information Security”, Vol.23, No. 2, Pages 100–108, 2017.

19. Yevseiev, S. P., Yokhov, O. Y. and Korol, O. G., “Data Gaining in Information Systems: monograph”. pub. KhNUE, Kharkiv, 2013.

20. Yevseiev, S., Rzayev, H. and Tsyganenko, A., “Analysis of the software implementation of direct and inverse transformations using the non-binary balanced coding method”, Science and Technology Journal “Security Without Information”, Vol. 22, No. 2, Pages 196–203, 2016.

21. Yeseiev, S., “The use of flawed codes in crypto-code systems”, Information processing systems, No. 5 (151), Pages 109–121, 2017.

22. Yevseiev, S. and Bilodid, I., “The use of unprofitable codes in hybrid crypto-code designs”, Fifth International Scientific and Technical Conference “Problems of Informatization”, Cherkasy – Baku – Bielsko-Biala – Poltava, Page 11, 2017.

23. Hryshchuk, R., Yevseiev, S. and Shmatko, A., “Construction methodology of information security system of banking information in automated banking systems: monograph”, Pages 134–156, Premier Publishing s. r. o., Vienna, 2018.