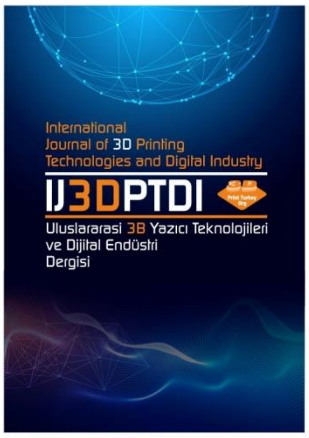
International
Journal of 3D Printing
Technologies and Digital Industry
IJ3DPTDI
Uluslararası 3B Yazıcı Teknolojileri
ve Dijital Endüstri
Dergisi

ULUSLARARASI 3B YAZICI TEKNOLOJİLERİ
VE DİJİTAL ENDÜSTRİ DERGİSİ

INTERNATIONAL JOURNAL OF 3D PRINTING
TECHNOLOGIES AND DIGITAL INDUSTRY

ISSN:2602-3350 (Online)

URL: https://dergipark.org.tr/ij3dptdi

FORMATION OF HASH CODES BASED ON THE UMAC ALGORITHM ON HYBRID CRYPTO-CODE CONSTRUCTIONS OF McELICE ON DAMAGED CODES

Yazarlar (Authors): Alla HAVRYLOVA*

Bu makaleye şu şekilde atıfta bulunabilirsiniz (To cite to this article): *Havrylova A.“ Formation Of Hash Codes Based On The Umac Algorithm On Hybrid Crypto-Code Constructions Of Mcelice On Damaged Codes”, Int. J. of 3D Printing Tech. Dig. Ind., 4(2): 106-115, (2020).*

DOI:10.46519/ij3dptdi.744251

Araştırma Makale/ Research Article

Erişim Linki: (To link to this article): https://dergipark.org.tr/en/pub/ij3dptdi/archive

FORMATION OF HASH CODES BASED ON THE UMAC ALGORITHM ON HYBRID CRYPTO-CODE CONSTRUCTIONS OF McELICE ON DAMAGED CODES

Alla HAVRYLOVA[a]*

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

*Corresponding Author: alla.gavrylova@hneu.net

(**Received:** 28.05.2020; **Revised:** 08.06.2020; **Accepted:** 24.07.2020)

ABSTRACT

A practical implementation of the improved UMAC algorithm on hybrid crypto-code constructions of McEliece with damage was carried out in order to increase the cryptographic strength of hash codes in post-quantum cryptography. The algorithm is based on the formation of a pseudo-random substrate at the third level of hash code generation. The use of hybrid crypto-code constructions allows maintaining the universality of the hash code. This contributes to an increase in the speed of the hash code generation, which will lead to a quick search for information in large databases by hash identifiers. The proposed approach makes it possible to generate MAC codes with various modifications in hybrid crypto-code constructions, while providing the formation of authentication profiles of various strengths and lengths.

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

1. INTRODUCTION

An important direction in the development of post-quantum cryptography is the use of crypto-code constructions [1, 2]. Their formation is based on the use of algebraic codes disguised as the so-called random code [3]. They allow realizing fast cryptographic data transformations and ensuring the reliability of transmitted data based on resistant to interference coding [4, 5]. Despite all the advantages, the use of crypto-code constructions in modern software and hardware is hampered by their practical implementation with a given level of cryptographic stability and resistance to attacks. At the same time, according to the experts of NIST (USA), the use of these constructions when generating the message authentication code provides the necessary level of protection and increases the level of cryptographic strength [6]. Therefore, the expediency of approaches to the formation of a message authentication code in the form of a crypto-code structure using an algorithm that increases the collision properties of hash codes can be justified by the results of calculations and a comparison of the generated messages by the sending and receiving parties.

2. MATERIAL AND METHOD

The development of computational capabilities, namely full-scale quantum computers, has jeopardized the use of classical mechanisms not only for symmetric cryptography, public key cryptography (including algorithms using the theory of elliptic curves), but also algorithms for providing authentication services based on MDC and MAC codes, specialized hash functions [7, 8]. In 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.

In articles [9] and [10] practical algorithms for crypto-code constructions are considered, which provide their practical implementation by reducing the power of the alphabet. Their use in the UMAC algorithm provides the required level of cryptographic stability of the generated hash code and preserves the universality of the entire message authentication code.

The use of a pseudo-random underlay can help to increase resistance to collisions in the formation of hash codes to the level of strict universality [11]. The results of the study, considered in article [11], showed that the practical application of the formation of a pseudo-random substrate on elliptic curves according to the McEliece scheme contributes both to an increase in the cryptographic strength of the message authentication code implementation and to an increase in the speed of operations related to it. The use of symmetric code-theoretic schemes on modified elliptic codes is based on algebraic codes, which makes it possible to obtain analytical expressions connecting the parameters of modified elliptic codes and symmetric crypto schemes.

When implementing the UMAC algorithm to form a pseudo-random substrate, it was proposed to use the AES (Advanced Encryption Standard) block symmetric encryption algorithm [12]. But its use does not guarantee the preservation of the universality property of the message authentication result code. This leads to a deterioration in the collision properties of the UMAC algorithm. To eliminate this drawback, it was proposed to use modular transformations, which are implemented by the asymmetric RSA (Rivest, Shamir, Adleman) encryption algorithm [13]. This algorithm is based on elliptic curves and the computational complexity of the factorization problem for large numbers. Using a quantum computer, the discrete logarithm underlying the RSA algorithm can be computed in polynomial time. This will mean the practical unsuitability of the cryptosystems formed on its basis for long-term data protection. It was also proposed to form a pseudo-random substrate based on modular transformations using the MASH-1 keyless algorithm [14]. But this algorithm in practice had temporary resistance and was hacked. An alternative to it can be the MASH-2 keyless algorithm. Its use should lead to an increase in the level of collision resistance, but it reduces the rate of formation of a pseudo-random substrate in real time [15] due to the high computational complexity of the implementation of this algorithm. Therefore, the task of this study is to calculate by calculation the equality of the generated hash codes in the form of hybrid crypto-code constructions on the damaged McEliece codes when implementing the UMAC algorithm on the sender and receiver sides to confirm the authentication of the transmitted message.

3. EXPERIMENTAL RESULTS

3.1. Construction of a modified UMAC algorithm using McEliece's hybrid crypto-code constructs for a shortened modified elliptic code

In articles [3] and [2], the mathematical model and block diagram of the hash code generation in the UMAC algorithm were considered using a pseudo-random substrate that ensures the cryptographic stability of the hash code. The use of various algebraic and multichannel cryptographic codes will allow the formation of various lengths of the hash code and provide the required level of its cryptographic strength. The main stages of creating a hash code are discussed in the article [1].

Let us consider an example of a practical implementation of the modified UMAC algorithm using a hybrid crypto-code construction on harmed codes. Input data for calculations are as follows:

1) Y_{L1I} – universal hash value (UHASH-hash) of the first level of hashing;
2) Y_{L3I} – hash value (Carter-Wegman-hash) of the third level of hashing;
3) *T* – data block;
4) *Blocklen* – data block length (bytes);
5) *K* – secret key;
6) *Keylen* – secret key length (32 bytes);
7) *Tag* – integrity and authenticity control code;
8) K_{L1I} – secret key of the first level of hashing, consisting of subkeys $K_1, K_2, \ldots, K_n$;
9) K_{L3I} – second-level hash secret key consisting of keys K_{L31} (subkeys $K_1, K_2, \ldots, K_n$) and K_{L32} (subkeys $K_1, K_2, \ldots, K_n$);
10) M – length of the transmitted plaintext array *I;*
11) K' – pseudo random key sequence;
12) *Numbyte* – pseudo-random key sequence length (number of subkeys);
13) *Index* – subkey number;
14) *I=11* – transmitted plaintext (*k*-bit information vector over *GF (q)*);

15) *Xor* ($\oplus$) – bitwise summation;
16) $x^3+y^2z+yz^2=0$ – algebraic curve over the field *GF (22);*
17) *e=00000200* – secret weight error vector $w_h(\mathrm{e}) \le \mathrm{t} = \left[\frac{d-1}{2}\right]$;
18) $X = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$ – nondegenerate $k \times k$ matrix;
19) $P = \begin{bmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \end{bmatrix}$ – permutation matrix of size $n \times n$;
20) $D = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&2&0&0&0&0&0&0 \\ 0&0&3&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&2&0&0&0 \\ 0&0&0&0&0&3&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix}$ – diagonal matrix equal 1;
21) $G = \begin{bmatrix} 2&2&3&0&1&3&0&1 \\ 3&3&2&1&0&2&1&0 \end{bmatrix}$ – generating matrix;
22) *Taglen* – the length of the integrity control code (authenticity) *PadCx* (4 bytes);
23) *Nonce* – unique number for input message *I* (8 bytes);
24) *Numbyte* – subkey length (equal to *Keylen*);
25) *Index* – subkey number (0);
26) *Cx=23023322* – cryptogram;
27) $P^{-1} = \begin{bmatrix} 0&0&0&1&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0 \end{bmatrix}$ – matrix inverse to the permutation matrix (since its determinant is 1, then $P^{-1} = P^T$);

28) $D^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ – 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;*

29) $X^{-1} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$ – matrix inverse of a non-degenerate matrix X .

The points of the algebraic curve are shown in Table 1.

Table 1. 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

3.1.1. Generating a hash code in the UMAC algorithm

The creation of a hash code for an open message is performed in parallel with the generation of the codogram, but we will describe the computational transformations in accordance with these steps in order. In accordance with the block diagram of the iterative formation of *Y*, *Pad* and *Tag* for an open message from the sender using the UMAC algorithm [16, 17], we distinguish the following calculation stages.

1st layer formation

The value of the hash function of the first level UHASH-hash Y_{L1I} is calculated by the formula Eq. (1):

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

To form it K_{L1I} we represent it as a sequence of keys from four-byte blocks is calculated by Eq. (2-4):

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

where || - concatenation (connection) of strings corresponding to subkeys.

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

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

Since $T_i = Index\|i$, then for the first level *Index* =1, => T_i :

T_1 = 1 || 1 = 00000001 000000001=>K_{1I}	T_{17} = 1 || 17 = 00000001 00010001=>K_{17I}
T_2 = 1 || 2 = 00000001 000000010=> K_{2I}	T_{18} = 1 || 18 = 00000001 00010010=> K_{18I}
T_3 = 1 || 3 = 00000001 000000011=> K_{3I}	T_{19} = 1 || 19 = 00000001 00010011=> K_{19I}
T_4 = 1 || 4 = 00000001 000000100=> K_{4I}	T_{20} = 1 || 20 = 00000001 00010100=> K_{20I}
T_5 = 1 || 5 = 00000001 000000101=> K_{5I}	T_{21} = 1 || 21 = 00000001 00010101=> K_{21I}
T_6 = 1 || 6 = 00000001 000000110=> K_{6I}	T_{22} = 1 || 22 = 00000001 00010110 => K_{22I}
T_7 = 1 || 7 = 00000001 000000111=> K_{7I}	T_{23} = 1 || 23 = 00000001 00010111 => K_{23I}
T_8 = 1 || 8 = 00000001 000001000=> K_{8I}	T_{24} = 1 || 24 = 00000001 00011000 => K_{24I}
T_9 = 1 || 9 = 00000001 000001001=> K_{9I}	T_{25} = 1 || 25 = 00000001 00011001=> K_{25I} (4)

T_{10} = 1 || 10 = 00000001 00001010=> K_{10I}	T_{26} = 1 || 26 = 00000001 00011010 => K_{26I}
T_{11} = 1 || 11 = 00000001 00001011=> K_{11I}	T_{27} = 1 || 27 = 00000001 00011011=> K_{27I}
T_{12} = 1 || 12 = 00000001 00001100=> K_{12I}	T_{28} = 1 || 28 = 00000001 00011100 => K_{28I}
T_{13} = 1 || 13 = 00000001 00001101=> K_{13I}	T_{29} = 1 || 29 = 00000001 00011101=> K_{29I}
T_{14} = 1 || 14 = 00000001 00001110=> K_{14I}	T_{30} = 1 || 30 = 00000001 00011110=> K_{30I}
T_{15} = 1 || 15 = 00000001 00001111=> K_{15I}	T_{31} = 1 || 31 = 00000001 00011111 => K_{31I}
T_{16} = 1 || 16 = 00000001 00010000=> K_{16I}	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, so the number of subkeys at this level is the same, and $K_{L1I} = T_1 = 0000000100000001$.
The hash values for this layer are calculated using the following formula Eq. (5):

$$Y_{L1I} = (I \;\; + K_{L1I}) \bmod 32 = (0100110+10000001) \bmod 32 = 111 \tag{5}$$

2nd layer formation
Since the length of *M* is less than 1024 bytes, this level of hashing will not be performed, but calculations will need to be performed using a third level hash code.

3rd layer formation
The number of subkeys K_{L31} and K_{L32} also depends on the values *Numbyte* and *Blocklen*.
The number of subkeys for K_{L31I} (Eq. (6-11))

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

Therefore, to form it K_{L31I}, we represent it as a sequence of keys of eight four-byte blocks:

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

For the third level *Index* =3, => T_i:

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

The number of subkeys for K_{L32I}:

$$n = \left\lceil \frac{Numbyte}{Blocklen} \right\rceil = \frac{4 \times 4}{32} = 0,5 \approx 1 => i = 1 \tag{9}$$

To form it K_{L32I}, we represent it as a sequence of keys from 1 four-byte subblock, ($K_{L32I} = K_{1I}$)

For the third level *Index*=4, => T_i:

$$T_i \;\; = 4 \,||\, 1 = 00000100\ 00000001 => K_{1I} \tag{10}$$

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

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

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

3.1.2. Formation of a cryptogram of an open message

1) The recipient generates a public key, which in the McEliece cryptosystem is matrices (Eq. (12-16)):

$$G_X^{MEC} = X \times G^{EC} \times P \times D \tag{12}$$

$$G_X^{MEC} = \begin{bmatrix} 2 & 1 & 3 & 0 & 1 & 1 & 1 & 0 \\ 0 & 2 & 2 & 2 & 2 & 0 & 3 & 2 \end{bmatrix} \tag{13}$$

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

$$C_X^* = I \times G_X^{MEC} \oplus e\,, \tag{14}$$

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

$$C_X^* = 23023322 \tag{15}$$

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

$$C_X^* = 2323322 \tag{16}$$

4) Damage to the original text based on its transformation is shown in Table 2.

Table 2. 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

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

5) The message with damage (flag) will be sent via the first channel to the recipient, the damaged code (remainder) will be sent via the second channel to the recipient.

Got, that $C_x^* = 1011001011111010_2$

Convert the given value to decimal representation: 45818_{10} – we send to the first channel.

Got, that checkbox *F(x)*=11111111_2.
Convert the given value to decimal representation: 255_{10} – we send to the second channel.

3.1.3. Generating a pseudo-random pad (*Pad*) using the PDF function

To ensure the cryptographic stability of the UMAC algorithm at the stability level of the used cryptographic algorithm, we form a pseudo-random block *Pad* from *Cx* to *I* using the *PDF* function (Eq. (17-19)):

$$Pad = PDF(K, Nonce, Taglen) \tag{17}$$

In accordance with the procedure for forming a pseudo-random pad Pad for *I*, it is necessary to generate the next subkey, represented as the *KDF* function:

$$K' = KDF(K, Index, Numbyte) \tag{18}$$
$$K' = KDF(0106, 0, 4)$$

Pseudo-random pad Pad will take the form:

$$Pad = PDF(0106, 8, 4) = 1101010 \tag{19}$$

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

3.1.4. Generating a validity code for a forwarded message

Generation of authentication codes of the received message is possible by the Eq. (20-23):

$$\begin{aligned} Tag &= UMAC\left(K, I, Nonce, Taglen\right) = Hash(K, I, Taglen) \oplus \\ &\oplus PDF(K, Nonce, Taglen) = Y_{L3M} \oplus Pad \end{aligned} \tag{20}$$
$$Tag = 10000000010 \oplus 1101010 = 10001101100 \tag{21}$$

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

$$Y = Y_{L3M} \oplus Tag \tag{22}$$
$$Y = 10000000010 \oplus 10001101100 = 1101110 = 110_{10} \tag{23}$$

3.2. Decrypting a received message

1) Recovering the received text
The values obtained from the two channels are converted into a binary number system by the Eq. (24-36):

$$C_x^* = 45818_{10} = 1011001011111010_2 \tag{24}$$
$$F(x) = 255_{10} = 11111111_2 \tag{25}$$

2) Damage recovery
Using Table 2, we get the codeword:

$$C_x^* = 010\ 011\ 000\ 010\ 011\ 011\ 010\ 010_2 = 2323322_{10} \tag{26}$$

3) To recover the closed text, the receiver adds null information characters at the location specified by the initialization vector IV*:*

$$C_x^* = 2323322 \rightarrow 23023322 \tag{27}$$

4) Remove the action of the secret permutation and diagonal matrices from the restored closed text *Cx*:

$$C_x^* = C_x \times D^{-1} \times P^{-1} \tag{28}$$

$$C_x^* = 22102221$$

5) Find the syndrome and the error locator polynomial by the formula:

$$S = C_x^* \times H^{EC^T} \tag{29}$$

We get the 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 (30)
S_{11}= 3+2+1+2+2=0
S_{02}= 2+1+1+3+3+2=0
$S = (1,1,1,0,0,0);$

Finding the error locator polynomial $\Lambda(x) = a_{00}+a_{10}x+y = 0$

$$\begin{bmatrix} S_{00} & S_{10} \\ S_{10} & S_{20} \end{bmatrix} \times \begin{bmatrix} a_{00} \\ a_{01} \end{bmatrix} = \begin{bmatrix} S_{01} \\ S_{11} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad a_{00}=0; \quad a_{10}=1; \tag{31}$$

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

6) Finding 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 (32)
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

$e^*= e_1 00 e_4 000 e_8$, knowing that: $e^* \times H^{EC^T} = S$, and solving the system of equations, we get: e_1 =0, e_4 = 2, e_8 =3

$$e^*=00020003 \tag{33}$$

Find $i^* = e^* + C_x^*$ (34)

$i^* = 00020003 \oplus 22102221 = 22;$ (35)

7) Find the transmitted message:

$i = i^* \times X^{-1}$, $i = 11$. (36)

Hence, the messages that were sent and received are the same. This means that no changes were made to them during the transfer. You also need to check if the sender's address is correct. To do this, we proceed to the implementation of paragraph. 3.3.

3.3. Checking the hash code on the receiving side using the UMAC algorithm

The next stage of the research is to verify the verification of the hash codes generated by the recipient and the sender of the authentication code of the sent message. For this, the authorized user (recipient) generates in accordance with paragraphs 3.1.1 – 3.1.4 a hash code from the message authentication code received from the sender. Verification is carried out by comparing received from the sender and generated by the recipient hash codes. If they match, the decision is made that the plaintext received through the open channel has not changed.

4. CONCLUSION

According to the results of the study, the cryptographic layer of the formation of message authentication codes both at the level of the transmitted cryptogram and at the level of the pseudo-random substrate satisfies the properties of universal hashing. The probability of a collision of the generated hash images does not exceed a given value. However, the provision of these requirements for the formation of hash codes is possible in the case of using the keyless modular transformation algorithm MASH-2. Therefore, the use of algorithms for the formation of hash codes and their verification on hybrid crypto-code constructions of McEliece with causing damage makes it possible to fulfill the universality condition, first of all, by increasing the level of cryptographic strength of the generated hash code of an open message transmitted over open telecommunication channels. Therefore, in the future, it is necessary to analyze the software implementation of the proposed algorithm to determine the speed level of the resulting hash codes and the probability of collisions.

REFERENCES

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

2. Король, Ольга и Гаврилова, Алла, "Реализация алгоритма UMAC на крипто-кодовых конструкциях", M.Sc. thesis, [Implementation of the UMAC algorithm on crypto-code constructions], [Thesis in Ukraine], Національний авіаційний університет, Kiev, 2020.

3. Гаврилова, А.А., "Применение алгоритма UMAC на крипто-кодовых конструкциях в блокчейн-технологиях", [Application of the UMAC algorithm on crypto-code constructions in blockchain technologies], [article in Ukraine], Науковий журнал "ScienceRise", Issue 12(65), Pages 20 – 23, 2019.

4. 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, Issue 4, Pages 15 – 30, 2011.

5. Yevseiev, Serhii and Havrylova, Alla, "Improved UMAC algorithm with crypto-code mceliece's scheme", Modern Problems Of Computer Science And IT-Education: collective monograph, Pages 79 – 92, Premier Publishing s.r.o., Vienna, 2020.

6. 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.

7. Wegman, M.N. and Carter, J.L., "New hash functions and their use in authentication and set equality", Journal of Computer and System Scince, Issue 22, Pages 265 – 279, 1981.

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

9. 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, Issue 7(97), Pages 131 – 132, 2011.

10. Chung-Wei Phan Raphael, "Mini Advanced Encryption Standard (Mini-AES): A testbed for Cryptanalysis Students", Cryptologia, Issue 26(4), Pages 283 – 306, 2002.

11. 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, Issue 2, Pages 73 – 78, 2012.

12. Yeseiev, S., "The use of flawed codes in crypto-code systems", Information processing systems, Issue 5(151), Pages 109 – 121, 2017.

13. Yevseiev, S.P., Korol, O.H., and Ogurtsov, V.V., "Усовершенствованный алгоритм UMAC на основе модулярных преобразований" [Enhanced UMAC Modular Transformation Algorithm], Восточно-европейский журнал передовых технологий, Issue 1/9 (67), Pages 16 – 23, 2014.

14. 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, Issue 2, Pages 100 – 108, 2017.

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

16. Yevseiev, S.P., Yokhov, O.Y. and Korol, O.G., "Data Gaining in Information Systems: monograph". pub. KhNUE, Pages 213, Kharkiv, 2013.

17. 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, Issue 2, Pages 196 – 203, 2016.