V. A. Luzhetskiy, Dr. Sc. (Eng.), Prof.; Yu. V. Baryshev

HASH CONSTRUCTIONS THAT ARE SECURE AGAINST MULTICOLLISIONS

The paper presents analysis of the attacks, based on finding multicollisions, and methods to counteract them. A generalized multi-pipe hash construction is proposed. Using this construction, the known methods of increasing hash constructions infeasibility are generalized and improved. A new approach to multicollision-resistant parallelized hashing design, formalized in the form of constructions, is proposed. Estimations of the proposed hash computing duration are presented.

Key words: *hash construction, multicollision, Joux attack, infeasible multi-pipe hash constructions.*

Introduction

Recently the field of cryptography that deals with hashing has faced a vital problem of providing resistance to multicollisions that are used in Joux attack. The problem originated from the birthday attack, which was counteracted by duplication of the resultant hash value length. This solution wasn't adequate to the available computational resources of the society and hashing time limitations. To reduce hashing duration, cascading described in [1] was proposed, i.e. parallelization of small-length hash values computation using different hash functions and concatenation of these values after the last iteration is completed. Cascading had been supposed to be efficient counteraction to the birthday attack till 2004, when Joux published his work [2]. Having found the so-called multicollisions in one of the cascaded hash functions, for which parallel computations are used, Joux showed that infeasibility of hash value, obtained by cascading, is not much greater than that of a single hash function being cascaded (on the condition that these hash functions have equal infeasibility). Therefore, the problem of hashing infeasibility and duration arose again.

Approaches proposed by Joux in his attack were generalized for an arbitrary quantity of hash functions and improved for quicker finding of multicollisions in works [3 – 6]. There were several attempts to propose mathematical models of hashing (traditionally called hash constructions in literature), which are secure against this kind of attack. The most famous constructions are HAIFA [7] and Lucks' double-pipe construction [8]. The former uses some specific arguments, but it has no theoretical proof of increasing the construction infeasibility, and the latter solves the problem by duplicating the computations and the length of intermediate hash values.

The goal of the research is the development of hash constructions, which provide parallelization of computations and counteract multicollision design.

To achieve the goal the following tasks must be solved:

- analysis of the known multicollision design methods;
- analysis of the known constructions that were proposed to increase multicollision-resistance;
- development of a generalized hash construction;
- determination of such properties of the generalized hash construction, that resist a cryptanalyst's attempts to design a multicollision.

Analysis of multicollision design methods

The first attack, where multicollisions were used, was proposed by Joux in work [2]. The attack is proposed for hash functions, which use cascading described in work [1] and are based on Merkle-Damgaard strengthened construction. According to the construction informational message $\mathbf{M}$ is to be padded to the length that is the multiple of the data block size, split into l parts and padded by additional block m_{l+1}, that has the length of original message $\mathbf{M}$, and hashing is performed according to the formula:

$$h_i = f\left(m_i, h_{i-1}\right), \tag{1}$$

where h_i – the intermediate hash value computed at the i-th step; $f(\cdot)$ – the compression function, which provides a fixed length of the result.

Let us consider two compression functions $f^{(1)}(\cdot)$ and $f^{(2)}(\cdot)$, that implement hashing according to formula (1). The resultant hash value is formed by concatenation of small-length hash values $h_{l+1}^{(1)}$ and $h_{l+1}^{(2)}$ computed using $f^{(1)}(\cdot)$ and $f^{(2)}(\cdot)$ respectively. The manner of hashing makes it possible to perform parallel computation of functions $f^{(1)}(\cdot)$ and $f^{(2)}(\cdot)$, and, consequently, to decrease hashing time. However, as Joux proved in paper [2], this decreases infeasibility of the resultant hash value cracking as compared with the one computed using a single function with the resultant hash value length being double that of one of the functions $f^{(1)}(\cdot)$ and $f^{(2)}(\cdot)$.

Joux implemented his attack by finding a multicollision for one function. Then, such collisions are found among the formed ones, which would cause a collision in the second function. The multicollision was obtained by Joux through finding for each of the i-th data block m_i another block m_i^* such that satisfied the following condition:

$$f^{(1)}\left(m_i, h_{i-1}\right) = f^{(1)}\left(m_i^*, h_{i-1}\right). \tag{2}$$

Fig. 1 shows schematically Joux's way of finding multicollisions for hash functions, that are based on the strengthened Merkle-Damgaard construction or an analogous one, described in work [2].

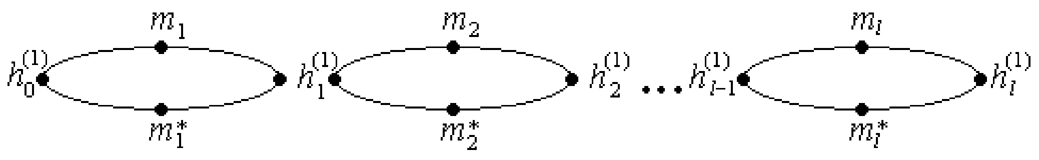

Fig. 1. The scheme of Joux's multicollision

Using l-multicollision, shown in fig. 1, it is possible to build 2^l different messages, that cause a collision of the first function $f^{(1)}(\cdot)$, and then, there exists a strong probability to find at least one message among them that would cause a collision in the second function $f^{(2)}(\cdot)$. In the case of using Merkle-Damgaard strengthened construction, the last hashed data block m_{l+1} is not an object of search for collisions, because it contains the length of the original message. Therefore, its forgery would be noticed immediately. It is clear that the greater the number of data blocks l is, the stronger probability of finding a collision will be. Thus, the complexity of finding collisions for n-bit hash value functions $f^{(1)}(\cdot)$ and $f^{(2)}(\cdot)$ is evaluated as $O\left(l \cdot 2^{n/2}\right)$ operations instead of the expected $O\left(2^{2n/2}\right)$.

In work [2] Joux proposed the way of finding collisions for the case when computation of two hash functions are parallelized and data blocks are processed only at one iteration. Work [3] was focused on finding multicollisions in the case when data blocks are processed by the compression function twice. Hoch and Shamir explicated and generalized this result in work [4]. They considered the case, when hash functions are expanded, i.e. input blocks for functions $f^{(1)}(\cdot)$ and $f^{(2)}(\cdot)$ can be inputted into the pipes, using the terminology of Lucks' paper [8], in random order. E.g., the intermediate hash value could be computed using function $f^{(1)}(\cdot)$ for input blocks with even indexes and then with odd ones, and using function $f^{(2)}(\cdot)$ – on the contrary, or any other way of input block permutation could be used. Moreover, in work [4] the term of "expanded" hash function concerns those ones, which process input blocks several times, e.g. inputting data from m_1 to m_l, and then in the inverse order. These ideas were explicated in paper [5], where the authors described hashing and multicollision design in terms of automata theory.

In work [6] an approach to designing multicollisions, based on finding expandable messages, is proposed. The messages are used in the same way as m_i^* in formula (2), but unlike in the method proposed by Joux, expandable messages can be formed using various quantities of data blocks. Thus, several messages of different length, which cause a collision and use the same intermediate value as the initial one, should be found. Besides, the authors of paper [6] showed the way of increasing efficiency of the attack in the case, when the compression function that forms the basis of hashing has fixed points, which is characteristic of Davis-Meyer construction. This approach has made it possible to achieve better results for long messages, than in work [2].

Analysis of the known hash constructions

The construction HAIFA (HAsh Iterative FrAmework) was proposed by the authors of paper [7] to counteract multicollision attacks. The construction provides for the extension of arguments of the iterative function by increasing their quantity in construction (1):

$$h_i = f\left(m_i, h_{i-1}, \#bits_i, r\right), \qquad (3)$$

where $\#bits_i$ – the quantity of the message bits that have been already hashed; r – a pseudorandom number.

In paper [7] a pseudorandom number r is proposed to be used as an identifier of the data exchange session or as a secret number in the case of hash function being used for digital signature. Introduction of additional arguments into the compression function allows a cryptanalyst to complicate the task of performing a pre-computed attack, but still the construction (3) doesn't make the task infeasible. Using the counter of the already hashed bits forces the cryptanalyst to find collisions taking into account a relative position of the data block in the message, but it doesn't counteract Joux attack. The main advantage of construction (3) consists in the following: till an intruder doesn't know the pseudorandom number r and at least a part of the message being hashed, he cannot start his attack.

Lucks' double-pipe construction [8] is the most infeasible construction for multicollision attacks, that is why it has the biggest potential for parallelization as compared with other constructions. The construction proposed by Lucks has resulted from the obvious response to Joux attack. If in the case of parallelization the complexity of n-bits hash cracking is $O\left(2^{n/2}\right)$, then it is necessary to duplicate the length of intermediate hash values [8]. So the wide-pipe construction evolved, but such increase of intermediate hash would obviously have a negative influence on hashing speed and computational resources. To avoid this, the double-pipe construction was proposed by Lucks. It provides two computational tools, i.e. pipes for finding n-bit intermediate hash value, informational data being split into k-bit blocks ($k \geq n$) [8]:

$$\begin{cases} h_i^{(1)} = f\left(h_{i-1}^{(1)}, h_{i-1}^{(2)} \,\|\, m_i\right) \\ h_i^{(2)} = f\left(h_{i-1}^{(2)}, h_{i-1}^{(1)} \,\|\, m_i\right) \end{cases}, \qquad (4)$$

where $h_i^{(1)}$ and $h_i^{(2)}$ – intermediate hash values determined by the first and the second pipes respectively;

After the last iteration an additional hash round is proposed to reduce the obtained hash value to n bits [8]:

$$h = f\left(h^{\circ}, h_l^{(1)} \,\|\, h_l^{(2)} \,\|\, 0^{k-n}\right), \qquad (5)$$

where h° is the initial vector analogous to $h_0^{(1)}$ and $h_0^{(2)}$; 0^{k-n} – padding to the complete $(k+n)$-bit block.

Main flaws of the double-pipe construction are double hardware overhead for hashing implementation as compared with other constructions; a large number of initialization vectors – $3n$, which complicates the development of the key hashing variant based on the construction. It is necessary to note that the last round decreases infeasibility of the entire construction and frustrates uniformity of the hashing process. Moreover, Lucks construction remains vulnerable to attacks that use pre-computation of the cryptanalyst.

Let us generalize the known constructions and determine the methods for multicollision design counteraction.

A generalized hash construction

We will consider a generalized iterative hash construction. Let us determine the basic arguments that could be used in the hash function. As their arguments, all known iterative hash constructions have a block of the message being hashed and the intermediate hash value received at the previous iteration as it is done in construction (1). Let us expand the notion taking into account that the intermediate hash value computed at the i-th iteration could directly depend on: all informative data blocks; i previous intermediate hash values (including initialization vector h_0); additional key data. As in most cases initialization vector h_0 is used as the key data, and usage of additional key data is determined by the specific features of certain tasks and doesn't influence finding multicollisions, we will not consider them separately in the generalized construction to avoid its growing bulky.

Therefore, the intermediate hash value to be received at the i-th iteration, can be computed according to the function of the form

$$h_i = f\left(h_0, h_1, ..., h_{i-1}, m_1, m_2, ..., m_l, m_{l+1}\right).$$

In general case strengthening of the construction may be performed taking into account the information bits that have been already hashed instead of padding the message by the original message length as it is done in construction (3):

$$h_i = f\left(h_0, h_1, ..., h_{i-1}, m_1, m_2, ..., m_l, length_i\right), \quad (6)$$

where $length_i$ – a bitwise length of the already hashed part of the message or its equivalent.

The arguments with their values changing pseudorandomly may be used in hashing. If a general case is considered, than the numbers of pseudorandom sequence can be used several times at different iterations. Moreover, several pseudorandom numbers may be used at one iteration. Taking this into account, formula (6) will have the following form:

$$h_i = f\left(h_0, h_1, ..., h_{i-1}, m_1, m_2, ..., m_l, r_1, r_{2,} ..., r_z, length_i\right), \quad (7)$$

where z – the quantity of pseudorandom numbers, used at one iteration; $r_1, r_{2,} ..., r_z$ – pseudorandom numbers.

Pseudorandom numbers may be published, determined just before the data exchange session or used as a secret parameter of a digital signature as it is proposed in work [7].

The usage of q pipes is required to parallelize hash computation. Construction (7) may be generalized in the following way:

$$\begin{cases} h_i^{(1)} = f\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(1)}, r_2^{(1)}, ..., r_{z_1}^{(1)}, length_i\right) \\ h_i^{(2)} = f\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(2)}, r_2^{(2)}, ..., r_{z_2}^{(2)}, length_i\right) \\ ... \\ h_i^{(q)} = f\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(q)}, r_2^{(q)}, ..., r_{z_q}^{(q)}, length_i\right) \end{cases}, \quad (8)$$

where z_j is the quantity of pseudorandom numbers, used in the j-th pipe ($\sum_{j=1}^{q} z_j = z$).

Moreover, hashing can be performed using different compression functions at different iterations in each pipe. So formula (8) will take the following form:

$$\begin{cases} h_i^{(1)} = f_i^{(1)}\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(1)}, r_2^{(1)}, ..., r_{z_1}^{(1)}, length_i\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(2)}, r_2^{(2)}, ..., r_{z_2}^{(2)}, length_i\right) \\ ... \\ h_i^{(q)} = f_i^{(q)}\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(q)}, r_2^{(q)}, ..., r_{z_q}^{(q)}, length_i\right) \end{cases}. \quad (9)$$

Let us introduce the following designation for multi-pipe hashing – $MPHq(k,d,r,z)$, where q is the quantity of pipes; k – the quantity of pipes, intermediate hash values of which directly influence the next intermediate hash value of the j-th $(j = 1,2,...,q)$ pipe; d – the quantity of data blocks, that take part in forming the next hash value in the j-th pipe; g – the grade of pseudorandomness that characterizes the pseudorandom numbers manipulating mode ($g = 0$ – pseudorandom numbers are not used, $g = 1$ – they are used as additional data, $g = 2$ – used as the indices of data blocks); z – the quantity of pseudorandom numbers, used at one iteration.

Thus, a generalized hash construction has been obtained. Let's consider its features that could counteract the multicollision design.

Determination of the methods for complicating multicollision design

One of the most obvious methods to complicate multicollision design is multiple processing of informational data blocks. However, the method could counteract only a classic Joux attack, but it cannot complicate the later attacks that have originated from it, e.g., the attack proposed in paper [4] is among them.

A more interesting method of multicollision design counteraction is simultaneous usage of several data blocks at each iteration, e.g.:

$$h_i = f\left(h_{i-1}, m_{i-a}, m_{i-b}\right), \quad (10)$$

where a, b are certain constants.

Construction (10) can be implemented in several ways: by sequential computation of two data blocks using construction (1) – in fact, those are two chaining iterations; by "conjunction" of m_{i-a} and m_{i-b} using a quickly performed operation. In the former case a hash function, vulnerability of which is proved in work [3], is obtained. The latter case is more interesting. Let operation that converts two operands into one be designated "$\circ$", then construction (10) can be rewritten in the following way:

$$\begin{cases} m_i' = m_{i-a} \circ m_{i-b} \\ h_i = f\left(h_{i-1}, m_i'\right) \end{cases}. \quad (11)$$

The obtained construction is more multicollision-resistant, than construction (1), but it apparently can be cracked by analogous actions. Thus, using Joux attack it is possible to find such $m_i'^{*} \neq m_i'$, that would satisfy the equation, analogous to (2), and then the multicollision is to be found. The main difference between constructions (1) and (11) is that in the case of construction (11) an intruder will have to solve the equation set to find m_{i-a}^{*} and m_{i-b}^{*}, which would not be a problem for a cryptanalyst.

The above-mentioned is not the evidence of the approach used in construction (10) being non-prospective, but its improvement is required. Let us change the grade of randomness and use pseudorandom numbers as hashed data blocks indexes to achieve this. Accordingly, introduction of randomness into the first equation of system (11) would counteract construction of the above equation set by an intruder. The following way of introducing randomness is proposed:

$$\begin{cases} r_i = rand(m_i) \\ m_i' = m_i \circ m_{i-r_i} \\ h_i = f(h_{i-1}, m_i') \end{cases}, \tag{12}$$

where $rand(\cdot)$ – a certain function with the uniform law of its values distribution.

E.g., function $rand(\cdot)$ can be implemented as a pseudorandom numbers generator, the initial state of which will be data block m_i. Construction (12) is secure against multicollisions, because an intruder is unable to perform sequential data blocks substitution to design a multicollision. Really, if one collision m_i^* is found, then block m_{i-r_i} and block $m_{i-r_i^*}$ ($r_i^* = rand(m_i^*)$) cannot be substituted to obtain a multicollision. Therefore, if one keeps looking for collisions, then it will be impossible to transform l collisions into l-multicollision. The multi-pipe variant of construction (12) $MPHq(1,2,2,1)$ will have the form of

$$\begin{cases} r_i = rand(m_i) \\ h_i^{(1)} = f_i^{(1)}\left(h_{i-1}^{(1)}, m_i \circ m_{i-r_i}\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_{i-1}^{(2)}, m_i \circ m_{i-r_i}\right) \\ \dots \\ h_i^{(q)} = f_i^{(q)}\left(h_{i-1}^{(q)}, m_i \circ m_{i-r_i}\right) \end{cases}. \tag{13}$$

Usage of the data blocks permutation before hashing is proposed for different pipes to complicate the task of cryptanalysis. Let $\left\{m_1^{(j)}, m_2^{(j)}, ..., m_l^{(j)}\right\}$ be some permutation of message blocks $\mathbf{M} = \{m_1, m_2, ..., m_l\}$, then construction $MPHq(1,2,2,1)$, described by formula (13), could be improved to become $MPHq(1,2,2,q)$:

$$\begin{cases} r_i^{(1)} = rand\left(m_i^{(1)}\right) \\ r_i^{(2)} = rand\left(m_i^{(2)}\right) \\ \dots \\ r_i^{(q)} = rand\left(m_i^{(q)}\right) \\ h_i^{(1)} = f_i^{(1)}\left(h_{i-1}^{(1)}, m_i^{(1)} \circ m_{i-r_i^{(1)}}^{(1)}\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_{i-1}^{(2)}, m_i^{(2)} \circ m_{i-r_i^{(2)}}^{(2)}\right) \\ \dots \\ h_i^{(q)} = f_i^{(q)}\left(h_{i-1}^{(q)}, m_i^{(q)} \circ m_{i-r_i^{(q)}}^{(q)}\right) \end{cases}. \tag{14}$$

Except this approach, proposed to increase infeasibility, an approach, proposed by Lucks in paper [8], may be used. Construction $MPH2(2,1,0,0)$, described by formulas (4) and (5), can be generalized by construction $MPHq(q,1,0,0)$:

$$\begin{cases} h_i^{(1)} = f_i^{(1)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, ..., h_{i-1}^{(q)}, m_i\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, ..., h_{i-1}^{(q)}, m_i\right) \\ ... \\ h_i^{(q)} = f_i^{(q)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, ..., h_{i-1}^{(q)}, m_i\right) \end{cases}. \tag{15}$$

Along with maintaining the pipes linkage effect after each iteration, the variant of construction $MPHq(q,1,0,0)$, described by formula (15), could be simplified to obtain construction $MPHq(2,1,0,0)$ that preserves the analogous infeasibility of hashed messages consisting of blocks, their quantity being no less than $(q/2+1)$.

$$\begin{cases} h_i^{(1)} = f_i^{(1)}\left(h_{i-1}^{(q)}, h_{i-1}^{(1)}, m_i\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, m_i\right) \\ ... \\ h_i^{(q)} = f_i^{(q)}\left(h_{i-1}^{(q-1)}, h_{i-1}^{(q)}, m_i\right) \end{cases}. \tag{16}$$

The advantage of construction $MPHq(1,2,2,q)$, described by formula (14), compared with construction $MPHq(2,1,0,0)$, described by formula (16), consists in that computations of each pipe do not depend on the computations, performed in other pipes, which allows to implement independent hash value computation. Hence, duration of hash values computation using the construction $MPHq(2,1,0,0)$ makes:

$$t_{MPHq(2,1,0,0)} \approx \sum_{i=1}^{l} \max\left(t_i^{(1)}, t_i^{(2)}, ..., t_i^{(q)}\right), \tag{17}$$

where $t_i^{(j)}$ – duration of the i-th iteration in the j-th pipe.

The time estimate for hash value computation using construction $MPHq(1,2,2,q)$ is:

$$t_{MPHq(1,2,2,q)} \approx \max\left(\sum_{i=1}^{l}\left(t_i^{(1)}\right), \sum_{i=1}^{l}\left(t_i^{(2)}\right), ... \sum_{i=1}^{l}\left(t_i^{(q)}\right)\right). \tag{18}$$

It is obvious, that always $t_{MPH(q,2,1,0)} \geq t_{MPH(q,1,2,2)}$.

Conclusions

Appearance of the birthday attack forced double increase of the hash value length, which caused substantial prolongation of computation time. This, in turn, made it impossible to use hashing in certain tasks. Therefore, parallelization of hashing must be performed to make computation time satisfy hashing duration requirements. However, multicollisions used in the Joux attack and its further generalization and explications, became an obstacle on the way of parallelization, their target being hash values obtained by parallelization of hash functions.

Analysis of the known methods of parallelization has shown that known constructions achieve multicollision resistance by double increase of computational recourses without decreasing computation time. The known Lucks approach was improved and generalized by decreasing computational recourses and by the ability to compute hash value using q parallel pipes instead of two, considered in paper [8], which makes it possible to decrease computation time. These approaches are formalized by constructions $MPHq(q,1,0,0)$ and $MPHq(2,1,0,0)$.

A new approach to multicollision-resistant hash design, formalized by the construction $MPHq(1,2,2,q)$, is proposed. The obtained estimations of hash values computation time indicate that the proposed approach, formalized by the construction $MPHq(1,2,2,q)$, will be faster than the improved one, formalized by the construction $MPHq(2,1,0,0)$, their implementation quality being the same.

REFERENCES

1. Preneel B. Analysis and Design of Cryptographic Hash Functions: PhD thesis / Bart Preneel. – Katholieke Universiteit Leuven, 1993. – 323 c. – Режим доступу до ресурсу : http://homes.esat.kuleuven.be/~preneel/phd_preneel_feb1993.pdf
2. Joux A. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions / Antoine Joux // Lecture Notes in Computer Science. – 2004. – № 3152. – С. 306-316.
3. Nandi M. Multicollision Attacks on Some Generalized Sequential Hash Functions / M. Nandi, D.R. Stinton // Scientific Literature Digital Library. – Режим доступу до ресурсу : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.5370&rep=rep1&type=pdf
4. Hoch J.J. Breaking the ICE – Finding Multicollisions in Iterative Concatenated and Expanded (ICE) Hash Functions / J.J. Hoch, A. Shamir. – 2006. – Режим доступу до ресурсу : http://www.wisdom.weizmann.ac.il/~yaakovh/papers/hashpaper_submission.pdf
5. Halunen K. An Automata-Theoretic Interpretation of Iterated Hash Functions – Application to Multicollisions / Kimmo Halunen, Juha Kortelainen, Tuomas Kortelainen // Cryptology ePrint Archive. – 2009. – 13 c. – Режим доступу до ресурсу : http://eprint.iacr.org/2009/456.pdf
6. Kelsey J. Second preimages on *n*-bit Hash Function for Less than 2^n Work. / J. Kelsey, B. Schneier // Cryptology ePrint Archive. – 2004. – 15 c. – Режим доступу до ресурсу : http://eprint.iacr.org/2004/304.pdf
7. Biham E. A Framework for Iterative Hash Functions: HAIFA. / Eli Biham, Orr Dunkelman // Second cryptographic hash workshop. – 2006. – 9 c. – Режим доступу до ресурсу : http://csrc.nist.gov/pki/HashWorkshop/2006/Papers/DUNKELMAN_NIST3.pdf
8. Lucks S. Design Principles for Iterated Hash Functions / S. Lucks // Cryptology ePrint Archive. – 2004. – 22 c. Режим доступу до ресурсу : http://eprint.iacr.org/2004/253.pdf

Volodymyr Luzhetskiy – Dc. Sc. (Eng.), prof., head of the Information protection department.

Yuriy Baryshev – Post graduate student of the Information protection department. E-mail: yuriy.baryshev@gmail.com

Vinnytsia National Technical University.