UDC 681.3.06.

MATHEMATICAL METHODS FOR CONSTRUCTING HASH FUNCTIONS BASED ON THE THEORY OF FINITE FIELDS IN INFORMATION TECHNOLOGY SYSTEMS

G. Vostrov, O. Ponomarenko
Odessa national polytechnic university

Abstract. *In this paper a method for constructing hash functions based on irreducible polynomials in finite fields was considered. The problem of finding irreducible polynomials was considered. A computer simulation of hash functions was performed using irreducible polynomials. The results of using various irreducible polynomials and their analysis are given. The problem of the occurrence of collisions depending on the length of the hash, and the probability of collisions is estimated depending on the number of enumeration operations were considered.*

Key words: *hash function, finite field, irreducible polynomial, collision, cyclic redundant code, polynomial arithmetic, "birthdays" paradox.*

Introduction

With an increase in the amount of information, problems associated with large amounts of data, which later require the implementation of the processes of storage, transmission or processing, are aggravated. Working with large volumes of files significantly complicates the specified processes, in connection with which there is a need for the existence of algorithms that allow to compress the volumes of files to the required size acceptable for their effective processing.

An important role in the process of interacting with files is played by hash functions. The use of hash functions implies a transformation of the original data according to a certain algorithm in a sequence of fixed length. This allows you to significantly speed up the search among a large variety of files to view, change or delete, to make comparisons between files, to check for immutability, in cases where the data should not be changed by outside persons. Thus, hashing is used in all areas where it builds the question of storing, transferring or processing data in the form of files, namely in cryptography, computer graphics, when organizing data on a computer and on the Internet.

At this stage of development of the theory of hashing, there is no clear definition of the concept of "hash function", as well as a clear classification of hashing methods. According to the definition of D.E. Knut, the hash function converts any possible key K (the original data) into a list number $h(K)$ in the range from 1 to m. This is true for data structures that store key values and its number [1], [2].

In cryptography, hash functions are designed to convert raw data into a sequence of fixed length in order to check for data immutability in the processes of storage or transfer [3]. An example of such data may be securities or passwords that should not be changed and accessible to a limited number of persons.

© G. Vostrov, O. Ponomarenko, 2018

Regardless of the scope of the hash functions, the same problem arises - the occurrence of collisions. A collision is a case where several files have the same hash, the result of using the hash function. The occurrence of collisions is due to the fact that in most cases the power of the set of various data to be hashed exceeds the power of the set of all kinds of hashes. For hash functions that take a variable-length input and return a hash of constant length, collisions will arise, because for at least one hash function value the corresponding set of input data will be infinite - and any two data sets from this set form a collision . When working with a large variety of files, collisions make it difficult to find data, and when transferring or storing valuable data, they allow the evil intent to replace them.

The requirements that cryptographic hash functions must satisfy (functions designed to ensure that an attacker protects files from access and change them) is irreversibility (it should be impossible to create an algorithm with polynomial computational complexity that restores the original data in real time) collision resistance, fast calculation speed and the presence of an avalanche effect (with a small change in the source data, the result should change significantly). Currently, 256-bit or more is considered an acceptable hash length in cryptography. However, with the advent of more powerful computers, this length will not be enough. The main problem with the use of hash functions is that the existence of irreversible functions, which exclude the possibility of collisions, has not been proven. In addition, there are no universal methods of hashing and they should be selected on the basis of their field of application.

In practice, functions are used for which the theoretical probability of occurrence of collisions is close to zero, but with the emergence of more powerful computational tools, the search for collisions may not be such an indescribable task. For this reason, existing algorithms require constant improvement. A special role for this has a complexity-theoretic problem, namely, algebraic number theory [4]. At present, number theory is widely used in cryptography; however, this is not sufficient when effective factorization algorithms appear. For this reason, it becomes necessary to use the theory of finite fields. Using irreducible polynomials over finite fields can be a more effective means of protection. It should be borne in mind that in the formation of hashing methods in finite fields, a number of mathematical problems may arise. One of these problems is the search for irreducible polynomials of a given degree over a field F_p or $GF(p)$ that can be used to find message hash codes. Since the data must be converted to a sequence of fixed length, it is convenient to use calculations in finite fields, since the result will be an element of the selected field, so the hash will take values from a finite set of elements.

The main goal of this paper is to analyze hash functions based on irreducible polynomials in finite fields. Next, the paper considers the problem of finding irreducible polynomials, and also describes a hashing method based on calculating the remainder of dividing by a irreducible polynomial.

1. Arithmetic of finite fields

Such sections of algebra as the theory of finite fields and the theory of polynomials over finite fields increasingly influence the construction of various systems for protecting information, encoding and decoding information. In particular, the algorithms of cyclic redundant codes appeared [5], which use polynomials over the field F_p. Cyclic redundancy codes can be used as hash functions for detecting errors and checking data integrity.

To understand the concept of a finite field, you must enter the definition of a ring and body. A ring (associative and c 1) is a system $(S,0,1,-,+,\cdot)$, where $(S,0,-,+)$ is an Abelian group (in which the commutativity law holds), $(S,1,\cdot)$ is a semigroup, with 1, satisfying the laws of distributivity. A ring is a body $0 \neq 1 \& \forall a \exists b \quad (ab = 1 \& ba = 1)$ in which, while a commutator body is called a field [6].Due to the fact that a finite field is a set with a finite number of elements, the operations of addition, subtraction, multiplication, and division can be performed in it in accordance with the field axioms [7]. Since the finite fields are closed with respect to the above operations, i.e. for any two field elements $a,b \in F_p$, when performing any of the operations $a \bullet b = c$, the result will be the element belonging to this field $c \in F_p$. It should be borne in mind that all calculations in finite fields are performed modulo p which is a characteristic of a finite field and is a prime number.

The simplest example of finite-A is the residue class ring $Z/(p)$ modulo a prime number p, which can be identified with the Galois field $F_p = GF(p)$ of order p [7]. According to the theorem on the existence and uniqueness of finite fields, for every prime number p and natural number n there is a finite field of p^n elements [7]. To build a field F_{p^n}, it is necessary to find a polynomial $P(x)$ of degree n irreducible over a field F_p. Such a field is represented by polynomials over F_p a degree not higher $n-1$. An irreducible polynomial is not decomposable into nontrivial polynomials and is an analog of prime numbers in the natural series. A feature of irreducible polynomials is that, being irreducible in one field, the polynomial will be reducible in another field, which has found application in coding theory and information protection systems.

The search for irreducible polynomials is a difficult task, especially over fields of large dimension. The search for irreducible polynomials requires efficient algorithms and large computational resources, as in the case of searching for primes, which is the main problem for building effective hashing algorithms based on them. At present, there are no effective algorithms for searching for irreducible polynomials; there are only irreducibility criteria and methods for testing irreducibility. The search is carried out by enumeration of polynomials and checks of each individual for irreducibility. To test a polynomial $P(x)$ degree $n \geq 2$ for irreducibility over a field of characteristic p, the following algorithm exists [8]:

1. The initial value of some polynomial is initialized $G_0(x) = x$.

2. The following value is calculated. $G_1(x) = G_0(x)^p \bmod P(x)$.

3. Calculate the greatest common factor (GCD) between $P(x)$ and $(G_1(x) - x)$. If the GCD is not equal to one, then the given polynomial is reducible. Otherwise, the next value is calculated by the recurrence formula

$G_i(x) = G_{i-1}(x)^p \bmod P(x), i = \overline{1, \lfloor n/2 \rfloor}$ where $\lfloor \ \rfloor$- is the operation of taking the integer part of the number.

4. If the GCD $P(x)$ and each of $(G_i(x) - x)$ is equal to one, then t polynomial $P(x)$ is irreducible.

The disadvantage of such an algorithm is the low computation speed with sufficiently large values n, since at each step the operation of raising to the power level and finding the GCD is performed. For this reason, we already consider irreducible polynomials over an extension of the Galois field $GF(2)$.

Polynomial arithmetic is used for calculations in finite fields. Adding to the field F_{p^n} corresponds to the usual addition of polynomials modulo p. Multiplication is performed in two stages - first, as usual multiplication of polynomials, and then the remainder of division by an irreducible polynomial is calculated, with which the field F_{p^n} is constructed. For example, fields of the same dimension can be constructed in different ways, depending on the choice of an irreducible many-member. They have the same order and is-morphs to each other. This follows from the fact that there are several irreducible polynomials of degree n for the characteristic field p. Examples of irreducible polynomials for fields F_2 are listed in Table 1.

Table 1

Irreducible polynomials over a field F_2

Degree of polynomial	Irreducible polynomials
2	$(x^2 + x + 1)$
3	$(x^3 + x^2 + 1)$, $(x^3 + x + 1)$
4	$(x^4 + x^3 + x^2 + x + 1)$, $(x^4 + x^3 + 1)$, $(x^4 + x + 1)$
5	$(x^5 + x^2 + 1)$, $(x^5 + x^3 + x^2 + x + 1)$, $(x^5 + x^4 + x^3 + x + 1)$, $(x^5 + x^4 + x^3 + x^2 + 1)$, $(x^5 + x^4 + x^2 + x + 1)$

There is an infinite number of irreducible polynomials over each field. The existence of efficient algorithms for the search for such polynomials will make it possible to find polynomials of large degrees — 256 degrees or more, which allows using hash functions based on the theory of finite fields in cryptography.

2. Hashing by irreducible polynomials

A possible way to construct a hash function in finite fields is to use modulo division of an irreducible polynomial [4]. Such an operation is similar to the division of polynomials in a column. For the efficiency of computer implementation, it is convenient to use calculations in the Galois fields F_{2^g} to characterize the number 2 to a power (with extension) corresponding to the hash length. This allows you to perform calculations on the data in the form of a sequence of bits. The search for the remainder of the division is implemented using per-bit shifts and an exclusive or (XOR) operation. However, it would be more efficient to use fields with characteristics of large primes.

To apply this method of hashing, the data is encoded in some chosen way into a sequence $a_1, a_2, ..., a_k$ of zeros and ones $a_i = \{0,1\}$, which corresponds to a certain polynomial $A(x) = a_1 x^{k-1} + a_2 x^{k-2} + ... + a_{k-1} x + a_k$ and the hash code will be a sequence of bits obtained by dividing by irreducible polynomial $P(x) = p_n x^n + p_{n-1} x^{n-1} + ... + p_1 x + p_0$ and is calculated by the formulas [9]:

$$B(x) = A(x) \bmod P(x) \qquad (1)$$

$$h(a_1, a_2, ..., a_k) = b_{n-1} b_{n-2} .. b_1 b_0 \qquad (2)$$

In formulas (1) and (2) $b_{n-1} b_{n-2} .. b_1 b_0$ are the coefficients of a polynomial $B(x)$ obtained as a remainder of dividing a polynomial $A(x) = a_1 x^{k-1} + a_2 x^{k-2} + ... + a_{k-1} x + a_k$ by a irreducible polynomial $P(x) = p_n x^n + p_{n-1} x^{n-1} + ... + p_1 x + p_0$ of degree n.

Such a function is resistant to the restoration of the original data, since even knowing the dimension of the field and the irreducible polynomial used, it is difficult to decipher the data, especially for large degrees of irreducible polynomial. Irreducible polynomials should be chosen on the basis of the area of application of hash functions, since the length of the convolution is equal to the degree of the polynomial. So, for use in information security systems, at the moment the optimal length is at least 128 bits and at most 512 bits. The use of a polynomial of sufficient dimension plays a significant role. If a selected pol-

ynomial degree is l, then the set of all possible values that the convolution function can take is equal to 2^l. For example, by using an irreducible polynomial $P(x) = x^4 + x + 1$, the number of various convolutions will be equal to 16 and the search for messages with the same convolutions will not be difficult.

A computer simulation of hash functions based on irreducible polynomials of degree 32 with a different number of monomials was carried out, as a result of which hashing efficiency was analyzed using each of the polynomials. An important factor when choosing an irreducible polynomial is the number of monomials in it. For greater computational speed, it is desirable to find polynomials with a minimum number of monomials. For comparison of the results of hashing, polynomials with 5, 12, and 18 monomials were chosen. The speeds of calculating hash convolutions of the input data of different lengths and using different irreducible polynomials of the same degree are given in Table 2.

Table 2

The conditional speed of calculating convolutions with the use of irreducible polynomials of degree 32.

№	**Irreducible polynomial**	**Calculation speed depending on the message length (in bits)**		
		64	**256**	**512**
1	$x^{32} + x^{22} + x^2 + x + 1$	0,22	0,37	0,41
2	$x^{32} + x^{31} + x^{24} + x^{22} + x^{16} + x^{14} + + x^8 + x^7 + x^5 + x^3 + x + 1$	0,39	0,43	0,51
3	$x^{32} + x^{30} + x^{29} + x^{28} + x^{26} + x^{20} + x^{19} + x^{17} + x^{16} + + x^{15} + x^{11} + x^{10} + x^7 + x^6 + x^4 + x^2 + x + 1$	0,43	0,73	1

At first glance, the difference in the speed of computations is small, however, when processing data volumes from 1MB and more, the difference in computations between functions can reach one-hour or more. This method of hashing is effective for use over small amounts of data within a few kilobytes.

Table 3 shows the results of hashing by the irreducible polynomials from Table 2 for an arbitrary 64-bit string and for the same line with small changes to check for the presence of the avalanche effect and mixing. For this, the 19th and 20th bits of the input sequence are swapped.

Table 3

The results of applying hash functions based on irreducible polynomials of degree 32

Input bit sequence	Hash results by irreducible polynomials		
	1	2	3
0110001100111001.. 0001110000011111.. 1110000011001000.. 0110110100011111	11010011.. 11101101.. 01000000.. 00001110	01000110.. 00001100.. 01101101.. 01100110	01010111.. 11011111.. 00101111.. 01001100
0110001100111001.. 0010110000011111.. 1110000011001000.. 0110110100011111	11010000.. 11101101.. 11010000.. 00101010	01010010.. 00000101.. 00100010.. 10110110	11110010.. 00010011.. 11101010.. 01001010

Functions implemented on the basis of the considered polynomials have the property of mixing. This means that no connection can be established between the convolution and the source data. Since with a small change in the source data, the result of the hashing should change significantly, the 19th and 20th bits in the initial sequence of bits were changed to check for compliance with this property. Based on the results given in Table 3, functions based on polynomials with a large number of monomials have the best avalanche effect. It is worth noting that, despite the small degree of the above polynomials, hash functions based on them have some resistance to the occurrence of collisions. When iterating over convolutions obtained during processing of data in the volumes 1000, 5000, 10000 and 30000, no collisions were found, although this cannot guarantee their absence on large volumes.

In the process of computer simulation of the hash function, no collisions were found on samples from the generated sequences, however, it is theoretically possible to estimate the probability of finding collisions depending on the number of all-possible hashes and the generated sequences. To evaluate the hash functions, para-dox "birthdays" is used [10]. According to paradox-sous, the probability of coincidence of convolutions of two sequences from one sample (collisions of the second kind) is calculated by the formula (3):

$$p(k) \approx 1 - e^{\frac{-2n^2}{2M}}, \qquad (3)$$

where k is the sample size, M is the number of all possible hashes. The estimate of the probability that a hash of a certain preselected sequence coincides with any hash value from a sample of the generated sequences is calculated as follows:

$$p(k) = 1 - \left(\frac{M-1}{M}\right)^n \qquad (4)$$

Thus, it is not difficult to show that the probability (3) of finding collisions of the second kind increases significantly with increasing sample size, as shown in Figure 1. Figure 2 shows the dependence of the probability of finding collisions of the first kind on the volume of the sample. Graphs are constructed for a hash function with a number $M = 2^{32}$ of various values.

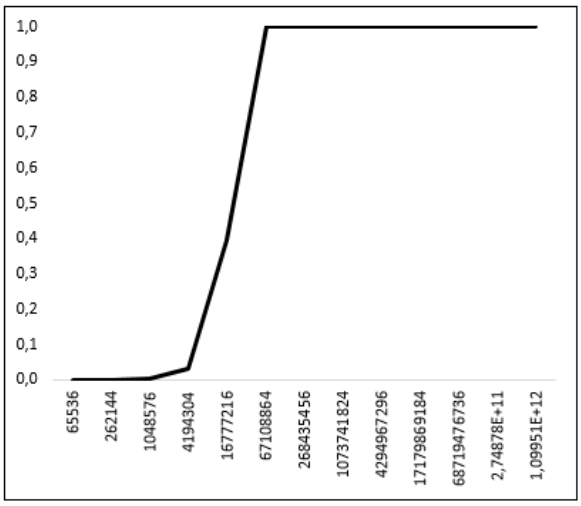

Figure 1. Probability of finding collisions of the second kind.

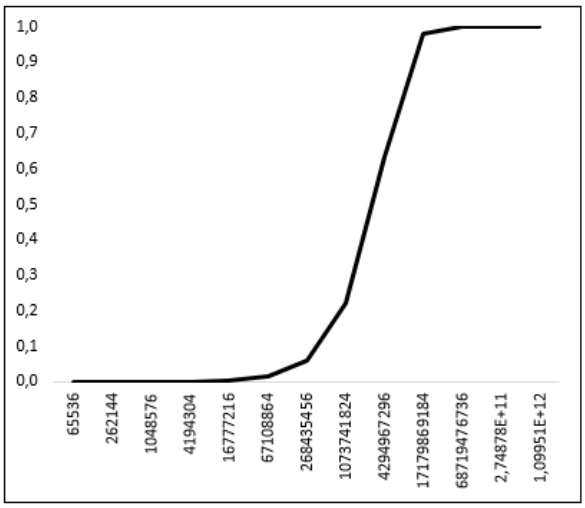

Figure 2. Probability of finding collisions of the first kind

Based on the results obtained, in order for the probability of finding the collisions of the first kind to be equal $p(k) = 1$, the sample k size must satisfy $k > 2^{34}$, and in order to find the collisions of the second kind - $k > 2^{18}$.

The considered hashing method is suitable for sequences of bits that can be represented by a polynomial to a greater degree than the degree of the irreducible polynomial chosen. Less long sequences must be supplemented with the help of some function. In most existing hashing algorithms, the addition to the required length is performed by adding one single bit and zero-bit bits to the sequence. In addition, the sequence is desirable to add its original length, which will reduce the likelihood of collisions after the addition. Using the remainder of dividing by an irreducible multi-term can serve as a separate hash function, and can be used in conjunction with other algorithms to improve certain properties. The same hash found by this method can be used as a cryptographic salt.

Conclusions

The theory of finite fields can be applied to the construction of hash functions, however, along with its application, problems arise that require a separate study for further solution. One of these problems is finding irreducible polynomials with certain properties. For the fields of the characteristic of high-digit prime numbers, the task of searching for irreducible polynomials of certain degrees is much more complicated and requires large computational costs. It was shown that it is advisable to use irreducible polynomials of sufficiently large degrees. Polynomials consisting of a small number of monomials allow finding convolutions for fewer operations. However, polynomials with a large number of multi-member enhance the avalanche effect of the hash function. Both those and others have the same resistance to collisions. An irreducible polynomial must be chosen on the basis of the required properties of the hash function. To enhance collision resistance and improve the avalanche effect, one should choose irreducible polynomials of degree 256 and higher with the maximum possible number of one-members. In cases where the hash function is used in systems that require high speed calculations, it is advisable to use irreducible polynomials with a minimum number of monomials.

The main disadvantage of hash functions based on irreducible polynomials is the low computation speed for large amounts of data. In addition to ex-

tensions of the considered field, in order to increase the resistance to collisions, it is necessary to consider fields of large characteristics, which is a task for a further solution. This will allow hashing data into elements from a larger field, but it should be borne in mind that this will complicate computational operations on the computer.

References

1. Graham, R., Knuth, D. (1998), Concrete mathematics. A foundation for computer science – 703 p.

2. Sedgewick, R. (2001), Fundamental algorithms on C++, - Kyiv: Publishing house "DiaSoft"– 688 p.

3. Schneier, B. (1996), Applied Cryptography, Second Edition, John Wiley & Sons, 816 p. ISBN 0-471-11709-9.

4. Khomich, E. A. (2017), Irreducible polynomials over finite fields and relation with cryptography. – pp.19–24.

5. Warren, Jr., Henry, S. (2002). Hacker's Delight (1 ed.). Addison Wesley. ISBN 978-0-201-91465-8. –512 p.

6. Lambek, J. (1966), Lectures on rings and modules. Waltham, Massachusetts : Blaisdell. – 283 p.

7. Lidl, R., Niederreiter, G.(1988), Finite fields: In 2 volumes, Transl. from English, - Moscow: Publishing house "Mir", 430 p.

8. Crandall, R., Pomerance, K. (2011), Prime numbers: cryptographic and computational aspects, Transl. from English / Ed. and with a preface by V. Chubarikova, - Moscow: URSS: Book House "LIBROKOM", 664 p.

9. Rabin, M. O., Karp, R. M. (1987). Efficient randomized pattern-matching algorithms – IBM, 1987. – № 2. – pp.. 249–260.

10. Goldberg, S. (1976), A Direct Attack on a Birthday Problem – Mathematical Mathematics Magazine, – pp. 130–132.

МАТЕМАТИЧНІ МЕТОДИ ПОБУДОВИ ХЕШ-ФУНКЦІЙ НА ОСНОВІ ТЕОРІЇ КІНЦЕВИХ ПОЛІВ В СИСТЕМАХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Востров Г. М., Пономаренко О. Ю.
Одеський національний політехнічний університет

Анотація. *У даній статті описана проблема визначення та класифікації хеш-функцій, в залежності від області застосування. Розглянуто метод побудови хеш-функцій на основі незвідних многочленів в кінцевих полях. Аналізуються характеристики незвідних многочленів, та результати їх використання. Розглянуто проблему пошуку незвідних многочленів високих степенів та недоліки алгоритмів їх пошуку, пов'язаних з необхідністю повного перебору. Було виконано комп'ютерне моделювання хеш-функцій з використанням незвідних многочленів з різною кількістю одночленів. Наведено результати використання різних незвідних многочленів та їх аналіз. Показано вплив обраного незвідного многочлену на результат хешування, зокрема на властивості «лавинного ефекту» та швидкість перетворення даних у хеш. Показано, що наведений метод побудови хеш-функцій достатньо стійкий до виникнення колізій, однак потребує використання обчислень в полях більших характеристик для використання в області криптографії. Розглянута проблема виникнення колізій в залежності від довжини хешу і оцінена ймовірність виникнення колізій в залежності від кількості операцій перебору. Показано, що можливість знаходження колізій є проблемою, яка виникає з появою швидкісних обчислювальних машин. Було виявлено, що метод потребує модифікації у випадках, якщо дані, переведені в послідовність біт, які підлягають обробці хеш-функцією, мають довжину, меншу за степінь незвідного многочлену. Зазначено, що даний метод може бути використаний як криптографічна сіль в сукупності з іншими методами хешування. Показано, що задля збільшення стійкості хеш-функцій до виникнення колізій, необхідним є використання многочленів більш високих порядків та перехід до обчислень в полях більших характеристик простих чисел. Показано, що розглянутий спосіб побудови хеш-функцій потребує розробки більш ефективних методів пошуку незвідних многочленів для знаходження многочленів достатньо високих степенів..*

Ключові слова: *хеш-функція, кінцеве поле, незвідний многочлен, колізія, циклічний надлишкових код, поліноміальна арифметика, парадокс «днів народження».*

МАТЕМАТИЧЕСКИЕ МЕТОДЫ ПОСТРОЕНИЯ ХЕШ-ФУНКЦИЙ НА ОСНОВЕ ТЕОРИИ КОНЕЧНЫХ ПОЛЕЙ В СИСТЕМАХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Востров Г. Н., Пономаренко Е. Ю.
Одесский национальный политехнический университет

Аннотация. *Рассмотрен метод построения хеш-функций на основе неприводимых многочленов в конечных полях. Рассмотрена проблема поиска неприводимых многочленов. Выполнено компьютерное моделирование хеш-функций с использованием неприводимых многочленов. Приведены результаты использования различных многочленов. Рассмотрена проблема возникновения коллизий.*

Ключевые слова: *хеш-функция, конечное поле, неприводимый многочлен, коллизия, циклический избыточных код, полиномиальная арифметика, парадокс «дней рождения».*

Received 03.12.2018

George Vostrov, Ph. D. of Technical Sciences, Associate Professor of the Department of Applied Mathematics and Information Technologies, Odessa National Polytechnic University. Shevchenko ave., 1, Odessa, Ukraine.
E-mail: vostrov@gmail.com, mob. +380503168776

Востров Георгій Миколайович, кандидат технічних наук, доцент кафедри прикладної математики та інформаційних технологій Одеського національного політехнічного університету. Проспект Шевченко, 1, Одеса, Україна.
E-mail: vostrov@gmail.com, тел. +380503168776

ORCID ID: 0000-0003-3856-5392

Olena Ponomarenko, Student of the Department of Applied Mathematics and Information Technologies, Odessa National Polytechnic University. Shevchenko ave., 1, Odessa, Ukraine.
E-mail: ponomarenkoelena1997@gmail.com, mob. +380934321669

Пономаренко Олена Юріївна, студент кафедри прикладної математики та інформаційних технологій, Одеського національного політехнічного університету. Проспект Шевченко, 1, Одеса, Україна.
E-mail: ponomarenkoelena1997@gmail.com, тел. +380934321669

ORCID ID: 0000-0003-1585-4706