УДК 519.8 doi: 10.15421/322015

**A.V. Siriak, V.A. Turchyna**

*Oles Honchar Dnipro National University*

# ZERO-KNOWLEDGE PROOF OF KNOWLEDGE OF MULTIVARIATE POLYNOMIAL'S ZERO

**The problem of constructing a NIZK proof of knowledge of a multivariate polynomial's zero is considered. A previously developed method for SNARK construction based on Quadratic Arithmetic Programs and Pinocchio protocol is adapted to solve the stated problem efficiently.**

**Keywords: proof of knowledge, non-interactive zero-knowledge proof, polynomial, algebraic equation, Pinocchio protocol, Quadratic Arithmetic Program, arithmetic circuit.**

**А.В. Сіряк, В.А. Турчина**

*Дніпровський національний університет імені Олеся Гончара*

# ДОВЕДЕННЯ З НУЛЬОВИМ РОЗГОЛОШЕННЯМ ЗНАННЯ НУЛЯ МНОГОЧЛЕНА БАГАТЬОХ ЗМІННИХ

**В даній статті розглядається актуальний в наш час підхід до побудови неінтерактивного доведення з нульовим розголошенням (англ. Non-Interactive Zero-Knowledge Proof). Він відноситься до некласичних методик доведення. Даний підхід використовується для побудови неінтерактивного доведення з нульовим розголошенням знання нуля многочлена багатьох змінних, тобто визначення n-вимірного числового вектору, що перетворює значення наперед заданої поліноміальної функції в нуль. Наводиться огляд деяких відомих на даний момент підходів до розв'язання задачі побудови неінтерактивних доведень з нульовим розголошенням. Такий підхід представляє як теоретичний, так і практичний інтерес оскільки доповнює теорію доведення новою схемою, яка також застосовується при реалізації розподілених обчислювальних систем. Наводяться прикладні сфери, в яких може бути застосований вказаний метод довенення. Дана формальна постановка задачі, пов'язаної з побудовою неінтерактивного доведення з нульовим розголошенням. Запропоновано підхід до її розв'язання, що базується на відомому методі побудови коротких неінтерактивних аргументів знання (англ. Succinct Non-Interactive Argument of Knowledge). Це один з підходів до побудови неінтерактивних доведень з нульовим розголошення. В цьому методі використовується один з багатьох добре розвинених, досліджених та зручних способів представлення обчислювальних задач, а саме, так звані квадратичні арифметичні програми. Також використовується протокол Піноккіо, адаптований для ефективного розв'язання задачі побудови неінтерактивного доведення з нульовим розголошенням знання нуля поліноміальної функції. Протокол Піноккіо дозволяє ефективно перевіряти узагальнені обчислення, базуючись лише на криптографічних припущеннях щодо складності розв'язання задачі знаходження дискретного логарифму.**

**Ключові слова: доведення знання, неінтерактивне доведення з нульовим пізнанням, многочлен, алгебраїчне рівняння, протокол Піноккіо, квадратична арифметична програма, арифметична схема.**

**А.В. Сиряк, В.А. Турчина**
*Днипровский национальный университет имени Олеся Гончара*

# ДОКАЗАТЕЛЬСТВО С НУЛЕВЫМ РАЗГЛАШЕНИЕМ ЗНАНИЯ НУЛЯ МНОГОЧЛЕНА МНОГИХ ПЕРЕМЕННЫХ

**Рассмотрена проблема построения неинтерактивного доказательства с нулевым разглашением (NIZK) знания нуля многочлена многих переменных. Разработанный ранее метод построения кратких неинтерактивных аргументов знания (SNARK) на основе квадратичных арифметических программ и протокола Пиноккио адаптирован для эффективного решения поставленной задачи.**

**Ключевые слова: доказательство знания, неинтерактивное доказательство с нулевым разглашением, многочлен, алгебраическое уравнение, протокол Пиноккио, квадратическая арифметическая программа, арифметическая схема.**

**Introduction.** One of many fields where researchers have made a lot of progress recently is a field of verifiable computations. The main problem under consideration in those papers is a problem of verifiable computations, namely outsourcing computations for one or another reason with the ability to verify the correctness of results as they arrive. One may want to do this for many reasons, some of which are: inability to perform computations on low power and/or low memory devices (e.g. cellphones, watches, laptops, tablets, etc.) because of speed and/or memory constraints and volunteer computing [1] (e.g. Berkeley Open Infrastructure for Network Computing, a distributed infrastructure for network computing based on a centralized server that distributes tasks and thus coordinates volunteer computer resources).

This project allows to outsource computations, but they are performed multiple times to ensure correctness, which is, obviously, less than optimal) or lack of resources to perform computations in-house. In any of the cases discussed above, computations can be outsourced to mainframes, personal computers or smartphones depending on the desired outcome and goal in mind.

Though this paper is not about verifiable computations in their pure form, it takes a closer look at zero-knowledge proofs in the context of verifiable computations. Namely, this paper's main concern is to prove that one knows a zero of a multivariate polynomial without exposing the zero. It's easy to see a link between zero-knowledge proofs and verifiable computations and, indeed, those fields are tightly coupled and very often techniques invented for use in one of them bring new developments in another. This paper is not an exception, but a confirmation to the rule as it dives into both of the fields to gain inspiration and state-of-the-art tooling to deal with problems in hand. It takes what's good and synthesizes a new method based on existing approaches.

**Formal problem statement.** Let $F_p = \{0,1,...,p-1\}$ be a finite field of integers modulo p with usually defined operations of addition and multiplication modulo p, namely:

$$a+b \equiv (a+b) \bmod p;$$
$$ab \equiv (ab) \bmod p.$$

Let us define the following field:

$$F_p^n = \left\{ (x_1, x_2, ..., x_n), x_i \in F_p, i = \overline{1,n} \right\}.$$

In $F_p^n$ operations are defined component-wise as follows:

$$(a+b)_i \equiv (a_i + b_i) \bmod p;$$
$$(ab)_i \equiv (a_i b_i) \bmod p, i = \overline{1,n}.$$

Let us also define the following map (1) that is a polynomial function:

$$P: F_p^n \to F_p; \tag{1}$$

$$P(x) \equiv \sum_{p_1=0}^{m_1} \sum_{p_2=0}^{m_2} \dots \sum_{p_n=0}^{m_n} a_{p_1 p_2 \dots p_n} x_1^{p_1} x_2^{p_2} \dots x_n^{p_n}, \tag{2}$$
$$x \in F_p^n, \quad x = (x_1, x_2, ..., x_n), x_i \in F_p, i = \overline{1,n}.$$

Two parties, Peggy and Victor, are required and it is supposed that Peggy knows a zero of polynomial P, i.e. Peggy knows a solution $x^0 = (x_1^0, x_2^0, ..., x_n^0)$ to the following algebraic equation:

$$P(x) = 0.$$

As function P is a function of multiple variables, finding an extremum of it doesn't pose any difficulties, namely one needs to check all the points where $\frac{dP(x)}{dx} = 0$, by which we mean $\nabla P(x) = \bar{0}$, as $x$ is a vector.

On the other hand, finding zeroes of P poses quite challenging, as those points don't have any special properties that may help to find them. Taking that into account, knowledge of $x^0 = (x_1^0, x_2^0, ..., x_n^0)$ such that $P(x^0) = 0$ is valuable and worth proving.

So, the following conditions are given:

1. Peggy and Victor both know some (but same for both) P from (2).
2. Peggy knows $x^0 : P(x^0) = 0$
3. Peggy wants to prove Victor her knowledge of $x^0$ without revealing any information regarding $x^0$.

**Proposed algorithm.** Here are all the steps of an algorithm that proves the fact that $x^0 = (x_1^0, x_2^0, ..., x_n^0)$ is indeed a zero of polynomial P without revealing $x^0$ or any other additional knowledge. Such an approach was first used for SNARK construction in [4], but, as we shall see soon, it can be adapted to solve our problem in hand.

Algorithm steps:

Convert polynomial P to an arithmetic circuit.

Reduce the arithmetic circuit to a quadratic arithmetic program.

Using Pinocchio protocol, prove Victor that Peggy knows a satisfying assignment to the QAP such that the last element of the assignment is 0.

Now let's take a closer look at each step of the algorithm and see how existing methods from multiple disciplines interplay to help us solve this, at first glance, unsolvable problem.

**Convert polynomial P to an arithmetic circuit.** First, let's state the following. **Definition 1** [8]. An arithmetic circuit is a rooted, directed acyclic graph whose leaves are numeric constants or variables, and whose interior nodes, which are called gates, are addition and multiplication operations. The value of the function for an input tuple is computed by setting the variable leaves to the corresponding values and computing the value of each node from the values of its children, starting at the leaves.

We are interested only in arithmetic circuits that include multiplication and addition, as it's enough to represent evaluation of any polynomial.

Any method of evaluation can be used here, but schemas that minimize the total number of operations that need to be performed and, hence, the size of the resulting circuit, are preferred.

Though the main reason to use arithmetic circuits in this paper is because of ability to transform multivariate polynomial evaluation into a polynomial of one variable using Quadratic Arithmetic Programs later, they can be utilized in a multitude of ways, some of which are identity testing, degree computation, and polynomial equivalence testing. Those and other algorithms on arithmetic circuits are discussed and analyzed in depth in paper [7].

Suppose the following polynomial function is given:

$$P(x) = 3x_1 + x_1x_2 .$$

Then, the corresponding arithmetic circuit is shown on fig. 1.

Reduce the arithmetic circuit to a quadratic arithmetic program.

In [9] an idea of using polynomials next to arithmetic circuits was first proposed. More than 20 years later, it received further development and in [6] a method of conversion of an arithmetic circuit into a Quadratic Arithmetic Program, which is used here, was proposed. This method makes use of polynomials and their properties to transform computations represented by an arithmetic circuit into polynomial with some very special properties.

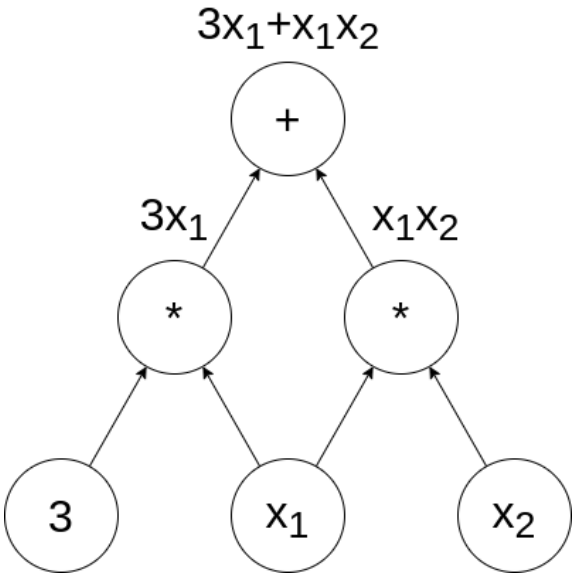


**Fig. 1. An example of an arithmetic circuit**

That polynomial is used later in the Quadratic Arithmetic Program to prove specific statements about the computations used to create the arithmetic circuit in the first place.

To use the proposed method, we first need to name wires and gates in our arithmetic circuit in a particular way. Almost all wires and almost all gates are associated with a moniker that is used later when we need to refer to a wire or a gate. The general schema will be explained in great detail later, but for now, let's take a look at fig. 2, which depicts our example arithmetic circuit with all required wires and gates labeled properly.

By $w_i, i = \overline{1,5}$ we mean "wires", which are values that travel through our circuit, $g_i, i = \overline{1,2}$ – multiplication gates, we don't label addition gates and think of values that come there as just passing through them to the next gate.

We also assume that all gates have no more and no less than 2 input wires, namely left and right wires.

After having done all the labeling, we can begin the reduction process. For that we first need to associate each multiplication gate with a field element and after that define sets of left $L_i$, right $R_i$ and output $O_i$ wire polynomials, $i = \overline{1,5}$.

An element that a multiplication gate is associated with is called a target point. We need to define our left, right and output wire polynomials in a way such that they are equal to zero on all target points except those associated with the corresponding multiplication gate.

Let's get back to our neat example. We will associate $g_1$ with $1 \in F_p$ and $g_2$ with $2 \in F_p$. We could have chosen any elements from field $F_p$, not necessarily 1 and 2, they only need to be different from one another.

Because $w_1$, $w_2$ and $w_4$ are left, right and output wires of the gate $g_1$ correspondingly, we define left, right and output polynomials of that gate in the following fashion: $L_1 = R_2 = O_4 = 2 - x$.

In this way $L_1(1) = R_2(1) = O_4(1) = 1 \neq 0$ and $L_1(2) = R_2(2) = O_4(2) = 0$.

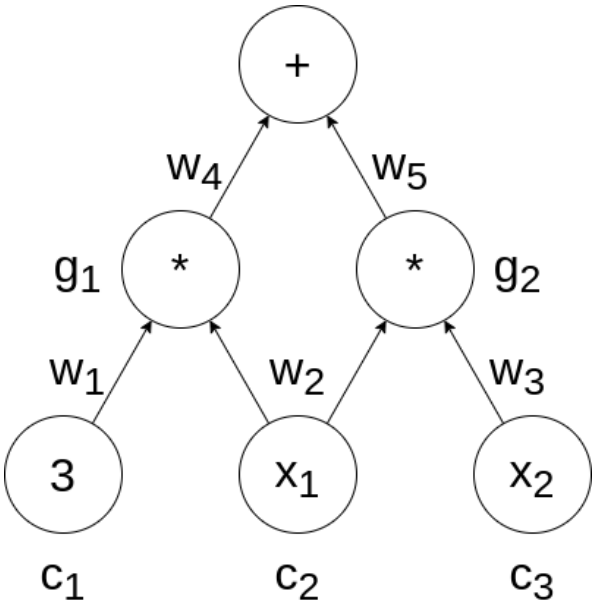


**Fig. 2. Labeled arithmetic circuit**

So, the polynomial we have chosen exhibits the required property of being not equal to zero only on the field element corresponding to its multiplication gate. In this particular case, polynomial $x - 2$ would also suffice, so we could have chosen it as well as the one we have chosen. This is an insight into the general form of this polynomial.

Let us construct required polynomials in a general form.

Let $G_L$ be a set of field elements that the wire $w$, the one we are considering, is connected to as a left wire.

Let $G_L = F_{p_L}$ be a set of all other field elements.

Then $L = \prod_{g \in G_L} (x - g)$ – left wire polynomial corresponding to the wire $w$.

As we can see, it's indeed zero on all field elements except those the wire $w$ is connected to as a left wire, and not zero on all other elements. Right and output wires are defined in a similar fashion.

**Definition 2** [6]. A tuple of numbers $(c_1, c_2, ..., c_n), c_i \in F_p, i = \overline{1,n}$ is called a legal assignment to an arithmetic circuit if after setting $w_i = c_i, i = \overline{1,n}$ for all gates $w_L * w_R = w_O$ where $w_L, w_R, w_O$ - left, right and output wires of the gate accordingly and * – gate operation.

After having finished constructing polynomials $L_i, R_i, O_i, i = \overline{1,n}$, where n is a number of wires in our circuit, let's define $L, R, O$ as follows:

$$L = \sum_{i=1}^{n} c_i L_i \tag{3}$$

$$R = \sum_{i=1}^{n} c_i R_i \tag{4}$$

$$O = \sum_{i=1}^{n} c_i O_i ; \tag{5}$$

$$P = LR - O, \tag{6}$$

where $(c_1, c_2, ..., c_n), c_i \in F_p, i = \overline{1,n}$ – some assignment to the arithmetic circuit in consideration.

Having defined this, we can state the following theorem.

**Theorem 1** [6]. A tuple $(c_1, c_2, ..., c_n), c_i \in F_p, i = \overline{1,n}$ is a legal assignment to the circuit if and only if P vanishes on all the target points.

**Definition 3** [6]. A Quadratic Arithmetic Program of degree d and size n consists of polynomials $L_i, R_i, O_i, i = \overline{1,n}$ and a target polynomial T of degree d.

**Definition 4** [6]. An assignment $(c_1, c_2, ..., c_n), c_i \in F_p, i = \overline{1,n}$ satisfies a Quadratic Arithmetic Program if target polynomial T divides polynomial P constructed by (6).

In this terminology, Peggy wants to prove to Victor that she knows a satisfying assignment $(c_1, c_2, ..., c_n)$ for the Quadratic Arithmetic Program such that the final value in arithmetic circuit evaluation is 0 (i.e. $c_n = 0$).

**Prove Victor that Peggy knows a satisfying assignment to the QAP using the Pinocchio protocol.** Let us introduce a function of a new kind:

$$E: F_p \to F_q .$$

Let E have the following properties:

$$\forall a \in F_p \forall b \in F_p : E(a+b) = E(a) + E(b);$$

$$\forall a \in F_p \forall b \in F_p : E(ab) = E(a)E(b);$$

$$\forall a \in F_p \forall b \in F_p : E(a) = E(b) \Rightarrow a = b .$$

There is a field that studies and may provide us with such functions. This field is concerned with homomorphic encryption. There is a lot of effort being put in developing such functions, for example, papers [2, 10, 13] are about that.

One more fact required here is DeMillo-Lipton-Schwartz-Zippel theorem – a tool often used in probabilistic polynomial identity testing.

**Theorem 2** (DeMillo-Lipton-Schwartz-Zippel) [5, 12, 14]. Let $P \in F[x_1, x_2, ..., x_n]$ be a non-zero polynomial of total degree d $\geq 0$ over a field F.

Let S be a finite subset of F and let $r_1,r_2,...,r_n$ be selected at random independently and uniformly from S. Then:

$$Pr\left[P\left(r_1,r_2,...,r_n\right)=0\right]\le\frac{d}{|S|}.$$

That's a lot of gear, but now we finally have everything to construct an algorithm that would allow Peggy to prove Victor the fact that she has a satisfying assignment.

1. Peggy chooses polynomials $L,R,O,H$ of degree not more than d.
2. Victor chooses a random point $s\in F_p$ and computes $E\left(T\left(s\right)\right)$.
3. Peggy sends Victor $E\left(L\left(s\right)\right),E\left(R\left(s\right)\right),E\left(O\left(s\right)\right),E\left(H\left(s\right)\right)$.
4. Victor checks that $E\left(L\left(s\right)R\left(s\right)-O\left(s\right)\right)=E\left(T\left(s\right)H\left(s\right)\right)$.

Here is a catch in that we have to make sure Peggy chooses according to an assignment and not just random polynomials of degree not more than d.

Let us combine polynomials $L,R,O$ in such a way:

$$F=L+x^{d+1}R+x^{2(d+1)}O.$$

Coefficients of F are coefficients of $L,R,O$ because $L,R,O$ have degree not more than d and so every addendum in polynomial $x^{d+1}R$ has a degree more than d and not more than $2d+1$, which means that coefficients don't mix up when we are constructing $F=L+x^{d+1}R+x^{2(d+1)}O$.

Let us introduce a new set of functions $F_i$:

$$F_i=L_i+x^{d+1}R_i+x^{2(d+1)}O_i.$$

Because of the way we have constructed polynomials $L,R,O$ in (3-5), one can easily see that in case if F was indeed produced knowing a legal assignment, the following equality holds:

$$F=\sum_{i=1}^{n}c_iF_i.$$

Therefore, to check if Peggy used a legal assignment when producing polynomials, Victor asks her to prove that F is a linear combination of $F_i$. This is done in a way we are about to describe.

Victor chooses a random $\beta\in F_p$ and sends Peggy encrypted values $E\left(\beta F_1\left(s\right)\right),E\left(\beta F_2\left(s\right)\right),...,E\left(\beta F_n\left(s\right)\right)$. Then Victor asks Peggy to send him $E\left(\beta F\left(s\right)\right)$ if she sends correct value, by Knowledge of Exponent Assumption (it's discussed in great depth in work [3]) Victor knows that she knows how to write F as a linear combination of $F_i$.

Peggy has proven that she knows a zero of polynomial to Victor, but it's not a Zero-Knowledge Proof yet. For example, in case Victor has some satisfying assignment $(\tilde{c}_1,\tilde{c}_2,...,\tilde{c}_n)$ he can compute corresponding $\tilde{L},\tilde{R},\tilde{O},\tilde{H}$ and $E(\tilde{L}),E(\tilde{R}),E(\tilde{O}),E(\tilde{H})$.

If it turns out that $E(\tilde{L})=E(L)$, $E(\tilde{R})=E(R)$, $E(\tilde{O})=E(O)$ and $E(\tilde{H})=E(H)$, then Victor knows that $(\tilde{c}_1,\tilde{c}_2,...,\tilde{c}_n)=(c_1,c_2,...,c_n)$, otherwise, he knows that $(\tilde{c}_1,\tilde{c}_2,...,\tilde{c}_n)\neq(c_1,c_2,...,c_n)$. So, he gains additional information.

To conceal that, we will add a random "T-shift" to our polynomials. To do that, we introduce new versions $L_z,R_z,O_z,H_z$ of already known to us polynomials $L,R,O,H$. Let $\delta_1,\delta_2,\delta_3\in F_p$ and define:

$$L_z = L+\delta_1 T\,;\, R_z = R+\delta_2 T\,;\, O_z = O+\delta_3 T\,.$$

As we've added a multiple of T to each polynomial, T shall still divide $L_zR_z-O_z$. Let's verify that:

$$\begin{aligned} L_zR_z-O_z &= (L+\delta_1 T)(R+\delta_2 T)-(O+\delta_3 T)= \\ &= T(H+\delta_1 R+\delta_2 L+\delta_1\delta_2 T-\delta_3). \end{aligned} \tag{7}$$

So, defining:

$$H_z = H+\delta_1 R+\delta_2 L+\delta_1\delta_2 T-\delta_3\,. \tag{8}$$

Substituting (8) into (7), we get:

$$L_zR_z-O_z = TH_z\,.$$

Therefore, Peggy can use our brand new $L_z,R_z,O_z,H_z$ instead of old $L,R,O,H$.

Those polynomials evaluated at any point will not reveal any information about a legal assignment $(c_1,c_2,...,c_n)$ because they are masked by random shifts.

So, we have developed a framework that allows proving that assignment satisfies our Quadratic Arithmetic Program and the only thing left is that the result of arithmetic circuit evaluation should be zero (i.e. $c_n = 0$), but, fortunately, this problem was already solved in [11] using Pinocchio protocol, which produces NIZK proof for us as well, and we won't delve into details here.

**Conclusion.** The problem of constructing a NIZK proof of knowledge of a multivariate polynomial's zero is considered in the paper.

A previously developed by Ben-Sasson et al. method based on Quadratic Arithmetic Programs and Pinocchio protocol is adapted to solve the problem in consideration. It, for given polynomial and its zero, first transforms it into an arithmetic circuit and after that, reduces obtained arithmetic circuit into a Quadratic Arithmetic Program.

After that, a framework is developed to prove that one knows a satisfying assignment to the Quadratic Arithmetic Program without revealing any additional information using the Pinocchio protocol and DeMillo-Lipton-Schwartz-Zippel theorem.

The prospects for further research are to test the proposed methods on a wider set of applied problems and to develop frameworks for solving other problems in fields of verifiable computations and zero-knowledge proofs.

**References**


1. **Anderson, D.** SETI@home: An Experiment in Public-Resource Computing [Text] / D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, D. Werthimer // Commun. ACM. – 2002. – pp. 56-61.
2. **Armknecht, F.** A Guide to Fully Homomorphic Encryption [Text] / F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C.A. Reuter, M. Strand // IACR Cryptology ePrint Archive. – 2015. – pp. 1192-1226.
3. **Bellare, M.** The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols [Text] / M. Bellare, A. Palacio // CRYPTO 2004: Advances in Cryptology. – 2004. – pp. 273-289.
4. **Ben-Sasson, E.** Succinct NonInter active Zero Knowledge for a von Neumann Architecture [Text] / E. Ben-Sasson, A. Chiesa, E. Tromer, M. Virza // USENIX Association. – 2014. – pp. 781-796.
5. **DeMillo, R.** A probabilistic remark on algebraic program testing [Text] / R. DeMillo, R. Lipton // Information Processing Letters. – 1978. – pp. 193-195.
6. **Gennaro, R.** Quadratic Span Programs and Succinct NIZKs without PCPs [Text] / R. Gennaro, C. Gentry, B. Parno, M. Raykova // EUROCRYPT 2013. Lecture Notes in Computer Science, vol 7881. – 2013. – pp. 215-274.
7. **Kayal, N.** Algorithms for Arithmetic Circuits. [Text] / N. Kayal // Electronic Colloquium on Computational Complexity (ECCC). – 2010. – pp. 53-77.
8. **Lowd, D.** Learning Arithmetic Circuits [Text] / D. Lowd, P. Domingos // Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence. – 2012. – pp. 383-392.
9. **Lund, C.** Algebraic methods for interactive proof systems [Text] / C. Lund, L. Fortnow, H.J. Karloff, N. Nisan // J. ACM. – 1992. – pp. 859-868.
10. **Ogburn, M.** Homomorphic Encryption [Text] / M. Ogburn, C. Turner, P. Dahal // Procedia Computer Science, 20. – 2013. – pp. 502-509.
11. **Parno, B.** Pinocchio: Nearly Practical Verifiable Computation. Proceedings [Text] / B. Parno, J. Howell, C. Gentry, M. Raykova // IEEE Symposium on Security and Privacy. – 2013. – pp. 238-252.
12. **Schwartz, J.** Fast Probabilistic Algorithms for Verification of Polynomial Identities [Text] / J. Schwartz // Journal of the ACM (JACM). – 1980. – pp. 701-717.
13. **Sen, J.** Homomorphic Encryption: Theory & Applications [Text] / J. Sen // Theory and Practice of Cryptography and Network Security Protocols and Technologies. – 2013. – pp. 1-31.
14. **Zippel, R.** Probabilistic algorithms for sparse polynomials. [Text] / R. Zippel // EUROSAM 1979: Symbolic and Algebraic Computation. – 1979. – pp. 216-226.


*Received 25.05. 2020.*