DOI: https://doi.org/10.26565/2304-6201-2024-63-01

УДК (UDC) 004.056.55, 004.051

Theoretical implementation and testing of the first stage of the ZK-STARK protocol “Arithmetisation”

Averkov O. Y. *Master’s student, Department of Security information Systems and Technologies, Faculty of Computer Science Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine,61022*e-mail: xa12850341@student.karazin.ua;

Kuznetsov O. O. *Doctor of technical science; Professor of Department of Security of Information Systems and Technologies, Faculty of Computer Science V. N. Karazin Kharkiv National University, Svobody Sq 4, Kharkiv Ukraine, 61022;*e-mail: kuznetsov@karazin.ua;

Lysytska I. V. *Doctor of technical science; Professor of Department of Security of Information Systems and Technologies, Faculty of Computer Science V. N. Karazin Kharkiv National University, Svobody Sq 4, Kharkiv Ukraine, 61022;*e-mail: ivlisitska@karazin.ua;

Relevance: Starting with the invention of the Internet, the world began to change rapidly, and the pace of change is increasing, so the problem of data storage and processing is becoming more and more relevant. The ZK-STARK protocol is a new cryptographic zero-knowledge proof protocol that is not yet widely used in practice and allows you to check a message or a transaction on the blockchain network for authenticity without reproducing it completely. At the moment, gaps and problems related to this protocol are identified: computational complexity, possible poor compatibility with other protocols, and resistance to attacks from quantum computers. Therefore, the paper aims to supplement the coverage of the problem associated with computational complexity and to propose solutions to this problem.

Purpose: on the basis of the theoretical implementation of the first stage named Arithmetization of the ZK-STARK protocol, to test its software implementation in order to provide recommendations on its most computationally efficient version.

Research methods: mathematical statements on interpolation theory, group theory, number theory; information on Fibonacci numbers; information on the Euler function; generating element of a group; cyclic groups; Lagrange interpolation polynomial and the sequence of calculations of Arithmetization; Visual Studio 2022 programming environment, C^{++} programming language, NTL library, Microsoft Excel.

Results of work: The result of the work is the theoretical implementation of the first stage of the ZK-STARK protocol and the effectiveness testing of the first stage, and providing recommendations for its most effective version.

Conclusion: Testing has shown that the practical implementation of the Arithmetization based on the inverse fast Fourier transform has a time complexity $O(n*\log_2(n))$, that is in $\frac{n^3}{n*\log_2(n)}$ times less than the time complexity of the Arithmetization based on inverse matrices method and Gaussian method for interpolation, that speeds up the work of Arithmetization of the ZK-STARK protocol.

***Keywords: protocol**, ZK-STARK, ARETHMETIZATION, modeling, program, implementation, C++ programming language, testing, blockchain, efficiency.*

Як цитувати: Averkov O.Y., Kuznetsov O.O., Lisitska I.V., Theoretical implementation and testing of the first stage of the ZK-STARK protocol “Arithmetisation”. Вісник Харківського національного університету імені В. Н. Каразіна, серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». 2024. вип. 63. С. 6-16. https://doi.org/10.26565/2304-6201-2024-63-01

How to quote: Averkov O.Y., Kuznetsov O.O., Lisitska I.V., “Theoretical implementation and testing of the first stage of the ZK-STARK protocol “Arithmetisation””. Bulletin of V.N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems, vol. 63, pp.6-16, 2024. https://doi.org/10.26565/2304-6201-2024-63-01

1. Introduction

For modern distributed blockchain networks (e.g. cryptocurrencies) that store and process large amounts of data, it is very important that information always retains such properties as confidentiality,

integrity and availability (especially after the invention and production of a mass quantum computer) and each transaction must undergo an authentication procedure. Today, the transaction authentication procedure is a computationally complex and slow procedure because it requires recreating the entire transaction, which causes additional losses of resources of a network.

In this regard, it is relevant to develop and implement transaction authentication protocols in the blockchain network that would allow transactions to have not only the properties of confidentiality, integrity and availability, but also save network resources (including the time of the network user) during the transaction authentication procedure. Today, there is a promising solution to save network resources by using the modern post-quantum-resistant ZK-STARK protocol for the authentication procedure.

Thus, mathematical modelling and testing of the ZK-STARK protocol implementation software determine the relevance of this work.

Elie Ben-Sasson, Iddo Bentov, Yinon Horeshi, and Mikhail Ryabzev wrote the first articles describing the STARK protocol back in 2018 [1]. At the moment, there is only one known ZK-STARK utility, which is being developed by StarkWare, a company created by the ZK-STARK designers. The goal is to develop a test layer that will allow the technology to be used on the blockchain, decentralised exchanges, and much more. As of today, the STARK protocol has not been widely tested, studied, and has not yet been widely used in any real production system, even in the world of cryptocurrencies [2, 3]. However, StarkWare has already launched an alpha version of the Layer 2 Rollup Network blockchain, which uses ZK-STARK for the Ethereum cryptocurrency [1, 4].

In general, the main aim of using of knowledge verification systems, such as ZK-STARK, is focused on the creation of highly secure and private systems. The systems where there is complete decentralisation of information, and access to it is possible only under a number of clearly defined conditions, that are also difficult to achieve by unconventional means, such as hacking [2].

Systems with decentralised information include systems such as cryptocurrencies, where the use of cryptocurrencies not only ensures network security but also protects users, providing them with privacy and anonymity, depending on the situation [2]. And it is in the latter case that ZK-SNARK stands out from other similar protocols because it is well suited to ensure privacy and anonymity without revealing the information in any way, but at the same time leaving the tool for confirming the transaction unambiguous and deterministic. In other words, ZK-STARK does not reveal the information it encrypts, but you can always verify its authenticity no matter what [2].

A possible application of STARK is to increase the scalability of the blockchain by allowing cryptographic tests to take up less space. In cryptocurrencies like Bitcoin, where block size limits the number of transactions that can be processed per second, this is vital. With smaller cryptographic test sizes, transactions also take up less space, and more transactions can fit into each block. The effect becomes stronger when applied to thousands of transactions, and scalability improves with it [2]. However, this is only part of the solution to the scalability problem, as the most minor cryptographic tests will not lead to a dramatic increase in blockchain performance.

Another possible application of this type of system could be, for example, a fully encrypted and secure streaming system. This would not require current encryption systems, which are mainly based on symmetric cryptography. Electronic voting systems also benefit greatly from this type of systems. This is because they allow a voter to cast a vote, that can be verified, but we have no way of knowing who cast it [2]. The potential of ZK-STARK is enormous.

The ZK-STARK protocol is a new technology that is not yet widely used in practice. However, at the moment, some gaps and problems related to its use are identified: prevalence, scalability, proof size, computational complexity (resource requirements), possible poor compatibility with other protocols, resistance to quantum attacks, and others [4-10]. Therefore, in our work, we decided to supplement the coverage of the problem associated with computational complexity and propose solutions to this problem.

In this paper, we will model and test low-degree cryptographic zero-knowledge proof protocols by testing a software implementation of Arithmetization of the ZK-STARK protocol based on the Fast Fourier Transform, Gauss method and inverse matrix method, and recommending the most efficient version of Arithmetization.

2. Formulation of a problem

The transaction authentication procedure of the ZK-STARK protocol takes place after the generation of the execution trace and polynomial constraints for the user-generated transaction. During the authentication procedure, these two objects are transformed into a single low-degree polynomial [11], which will be a low-degree polynomial only if the execution trace is correct and, accordingly, the data on which the computation trace was generated is correct [12].

Taking into account that the Arithmetisation stage involves the use of 'Error Correction Codes', the plan for this stage may look like this [13]:

1) reformulate the execution trace into a polynomial form

2) extend the execution trace to a larger domain

3) transform the execution trace, using polynomial constraints, into another polynomial that is guaranteed to have a low degree if and only if the execution trace is correct

Based on the available information, we can model the operation of the first stage of the protocol using a specific example.

Let's say that according to the task we are given:

A finite field $Z_{13} = \{0,1,2,3,4,5,6,7,8,9,10,11,12\}$, which has $|Z_{13}| = 13$ numbers with addition and multiplication modulo 13. This field has a multiplicative subgroup G with length $|G| = 6$ and a generator g = 4. The existence of such a subgroup is guaranteed since 6 divides the size of this group (which is 12) without a remainder [13].

Suppose that the statement about the need to verify the computational integrity of a transaction sounds like this: 'The verifier has a sequence of 6 numbers, all of which are Fibonacci numbers.

This sequence of Fibonacci numbers must be verified by reading significantly fewer than 6 numbers [13].

This problem has the following solution:

The Fibonacci sequence is formally defined as follows:

$$a_0 = 1$$
$$a_1 = 1$$
$$a_{n+2} = (a_{n+1} + a_n) \bmod 13$$

You can create an execution trace by simply writing down all 6 Fibonacci numbers in a row: 1, 1, 2, 3, 5, 8. Then, the polynomial constraints can have the next form [13]:

$$\begin{Bmatrix} A_0 - 1 = 0 \; ma \; A_1 - 1 = 0, \\ \forall 0 \le i < 4 : A_{i+2} - A_{i+1} - A_i = 0, \\ A_5 - 8 = 0. \end{Bmatrix}$$

Now let's bring the polynomial constraints to a polynomial form:

The recurrent Fibonacci relation embodies a set of constraints on the entire execution trace, and can be alternatively interpreted as follows [13]:

$$\forall 0 \le i < 4 : f(g^{i+2}) - f(g^{i+1}) - f(g^i) = 0,$$

Now the Verifier can create a polynomial composition using the formula [24]:

$$q(x) = \frac{f(g^{i+2}) - f(g^{i+1}) - f(g^i)}{\prod_{i=0}^{3} (x - g^i)},$$

The calculation of this expression for the special case when the degrees of g form a subgroup can be performed as follows [13]:

$$x^{|G|} - 1 = \prod_{g \in G} (x - g),$$

This equality is correct because both sides are polynomials of degree $|G|$, whose roots are exactly elements of G [13].

And the actual denominator of the considered composite Fibonacci polynomial can be obtained by rewriting it in the form [13]:

$$\frac{f(g^{i+2})-f(g^{i+1})-f(g^{i})}{\prod_{i=0}^{3}(x-g^{i})}=\frac{(\omega-g^{4})*(\omega-g^{5})*[f(g^{2}*\omega)-f(g^{1}*\omega)-f(\omega)]}{\omega^{6}-1}, \quad (1)$$

where $\omega\in\{1,g^{1},g^{2},g^{3},g^{4},g^{5}\}$.

3. Research methods

To solve equation (1), we must first calculate the coefficients of the interpolation polynomial $f(g^{0}*x)$. For this purpose, it may be used the following formula to find the coefficients of the Lagrange interpolation polynomial [14], but provided that the calculations are carried out in a specific field.

$$L_{n}(x)=\sum_{i=0}^{n}f(x_{i})*\frac{\omega_{n}(x)}{(x-x_{i})*\omega_{n}^{|}(x_{i})}, \quad (2)$$

where, x – is the argument of the interpolation polynomial and the function f(x), $\omega_{n}(x)=(x-x_{0})*(x-x_{1})\ldots(x-x_{n})$.

Using the notation of the previous section, the arguments of the interpolation polynomial $f(g^{0}*x)$ are $g^{i}=\{1,g^{1},g^{2},g^{3},g^{4},g^{5}\}$, where $i=0,1\ldots n$, where n – the number of Fibonacci numbers that are the results of the function $f(g^{0}*x)$ in the domain $x\in\{1,g^{1},g^{2},g^{3},g^{4},g^{5}\}$.

After finding the coefficients of the interpolation polynomial $f(g^{0}*x)$ you need to calculate the found polynomial also for the cases $f(g^{1}*x)$ and $f(g^{2}*x)$.

The next step in the calculations is to authenticate the blockchain transaction using the right-hand side of formula (1).

This paper will consider three variants of the method of finding the coefficients of the interpolation polynomial:

1) Using the Gaussian method with the selection of the main element by column.

2) Using the inverse matrix method with the search for the inverse matrix using the Gaussian method (used to convert the matrix to the upper triangular form).

3) Using the Fast Fourier Transform.

The first variant [15] is equivalent to applying the conventional Gaussian method (G) to a system in which the equations are renumbered accordingly at each elimination step.

$$\left|a^{(k)}{}_{rk}\right|=\max_{i}\left|a^{(k)}{}_{ik}\right|,\ k\le i\le n.$$

However, the main element can be selected among all the elements of the untransformed part of the matrix (Fig. 1, b):

$$\left|a^{(k)}{}_{rs}\right|=\max_{i,j}\left|a^{(k)}{}_{ij}\right|,\ k\le i,\quad j<n.$$

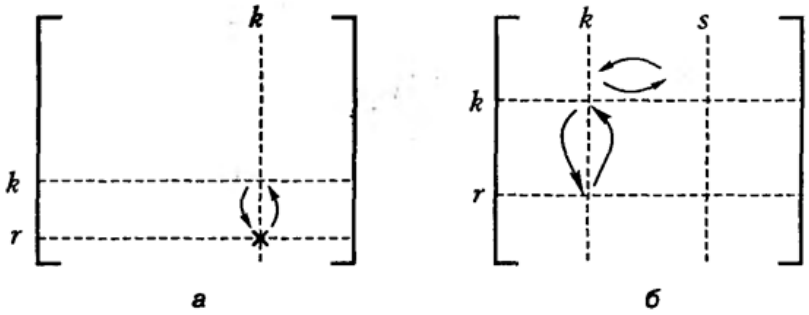

Figure 1 – Selection of the main element: a - among the elements of the matrix column; b - among the elements of the untransformed part of the matrix..

The complexity of the algorithm (the number of arithmetic operations) is estimated by $f_A(n) = \frac{n^3}{3} + n^2 = O(n^3)$

The algorithm is modified for use in modular arithmetic.

Consider the second variant, in which the inverse matrix method is used to find the coefficients of the interpolation polynomial with the search for the inverse matrix using the Gaussian method (IM) [16]. It is used to convert the matrix to the upper triangular form.

Let the given system:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1,$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n,$$

Or in matrix form

$$\mathrm{A} * \mathrm{X} = \mathrm{B}, \quad (3)$$

where $\mathrm{A} = (a_{ij})$ - is a matrix with coefficients at the unknown variables, B and X are a column vector consisting of free terms and unknown variables.

If the matrix A is nondegenerate, i.e., the determinant of the system is $\Delta = |A| \neq 0$, then multiplying both parts of equation (3) by the matrix A^{-1} on the left, we obtain the solution of the system in matrix form: $X = B * A^{-1}$.

The time complexity of the method is $O(n^3)$

The third variant for finding the coefficients of the interpolation polynomial is to use the Fast Fourier Transform. The interpolation is performed using the Cooley-Tukey method to calculate the Inverse Fast Fourier Transform (IFFT).

Let's say we have a sequence of numbers $v = (v_0, v_1 \ldots v_{n-1})$ and we want to compute its DFT $V = (V_0, V_1 \ldots V_{n-1})$. Instead of directly using the DFT algorithm, the Cooley-Tukey algorithm splits it into two parts, one for even indices and one for odd indices:

$V_j = E_j + e^{-1*(\frac{2*\pi*i}{n})*j} * O_j$, where E_j – DFT for even indices, a O_j – DFT for odd indices. Now, instead of calculating one DFT of size n, we calculate two DFTs of size $\frac{n}{2}$. This process can be repeated recursively, breaking the DFT into smaller parts until the size becomes small enough for direct calculation. Also, thanks to the properties of complex numbers, it is possible to reuse the results of calculations E_j and O_j for calculating V_j. This leads to a significant reduction in computational costs [17].

In order to use this method to find the coefficients of the interpolation polynomial, one must change the direction of traversal of the complex plane by multiplying the degree of the number e by minus one and multiplying the resulting number $e^{-1*(\frac{2*\pi*i}{n})}$ on $\frac{1}{n}$, where n – is the size of the input sequence. Compared to the other two, this method looks the most promising because its time complexity is equal to $n * \log_2(n)$ [18].

This method will be used for calculations in the field: the algorithm for calculations in the field of complex numbers differs from the algorithm for calculations in the field by modulo only in that the number $e^{-1*(\frac{2*\pi*i}{n})}$ is replaced by the element α^i of a group, which forms a cyclic subgroup of the multiplicative group [19].

4. Results

4.1 Theoretical implementation of the ZK-STARK protocol

To solve a simple example, let's use formula (2) directly and get the numerical solution of the problem as follows:

1) we find the interpolation Lagrange polynomial for the six Fibonacci numbers.

The polynomial has the form:

$$f(x) = 7*x^5 + 10*x^4 + 8*x^3 + 6*x^2 + 10*x + 12.$$

The check shows that the polynomial is calculated correctly:

With $x = g^0 = 1$:

$$f(x) = 7*1^5 + 10*1^4 + 8*1^3 + 6*1^2 + 10*1 + 12 = 1 \bmod(13).$$

Corresponds to the first Fibonacci number.

With $x = g^1 = 4$:

$$f(x) = 7*4^5 + 10*4^4 + 8*4^3 + 6*4^2 + 10*4 + 12 = 1 \bmod(13).$$

Corresponds to the second Fibonacci number.

With $x = g^2 = 3$:

$$f(x) = 7*3^5 + 10*3^4 + 8*3^3 + 6*3^2 + 10*3 + 12 = 2 \bmod(13).$$

Corresponds to the third Fibonacci number.

With $x = g^3 = 12$:

$$f(x) = 7*12^5 + 10*12^4 + 8*12^3 + 6*12^2 + 10*12 + 12 = 3 \bmod(13).$$

Corresponds to the fourth Fibonacci number.

With $x = g^4 = 9$:

$$\text{f(x)} = \ f(x) = 7*9^5 + 10*9^4 + 8*9^3 + 6*9^2 + 10*9 + 12 = 5 \bmod(13).$$

Corresponds to the fifth Fibonacci number.

With $x = g^5 = 10$:

$$f(x) = 7*10^5 + 10*10^4 + 8*10^3 + 6*10^2 + 10*10 + 12 = 8 \bmod(13).$$

Corresponds to the sixth Fibonacci number.

Let's check the property of the recurrence relation, which has the form:

$$\forall\, 0 \in \{g^0, g^1, g^2, g^3\}:\ f(g^2 x) - f(g^1 x) - f(x) = 0.$$

To do this, in the interpolation polynomial for six numbers, we substitute the value of x for x, $g*x$, g^2*x in such a way that next polynomials are formed:

$$f(x) = 7*x^5 + 10*x^4 + 8*x^3 + 6*x^2 + 10*x + 12$$
$$f(g*x) = 5*x^5 + 12*x^4 + 5*x^3 + 5*x^2 + 1*x + 12 \quad .$$
$$f(g^2*x) = 11*x^5 + 4*x^4 + 8*x^3 + 2*x^2 + 4*x + 12$$

Then the composition polynomial looks like this:

$$q(x) = \frac{f(g^{i+2}) - f(g^{i+1}) - f(g^i)}{x^6 - 1} = \frac{12*x^5 + 8*x^4 + 8*x^3 + 4*x^2 + 6*x + 1}{x^6 - 1}.$$

Based on equality (1), we find the composition polynomial q(x)[3]:

$$q(x) = \frac{(x-9)*(x-10)*[f(g^{i+2}) - f(g^{i+1}) - f(g^i)]}{x^6 - 1} = \frac{(x-9)*(x-10)*[12*x^5 + 8*x^4 + 8*x^3 + 4*x^2 + 6*x + 1]}{x^6 - 1}$$

If the Verifier receives the correct data and creates the correct polynomial, than we have that the composition polynomial q(x) is a polynomial of degree less than two and looks like this:

$$q(x) = \frac{12*x^7 + x^6 + x + 12}{x^6 - 1} = 12*x + 1, \qquad (4)$$

Using this algorithm, the Verifier checks the transaction data for authenticity. Based on the existing algorithm of Verifier actions, the C++ program was tested on a different number of input Fibonacci numbers. By default, the Verifier always enters correct data.

5. Testing a C^{++} implementation of the Arithmetization

According to the test results, we have determined that the Inverse Fast Fourier Transform (IFFT) method is the most effective for solving a system of linear algebraic equations (SLAE) with a large number of unknowns and equations. This method is able to solve SLAE with the number of unknowns and equations that is almost in 16578 times bigger than the number of unknowns and equations in the Gaussian (G) and Inverse Matrix (IM) methods.

When using the IFFT method to perform the interpolation procedure, it is possible to give to the input in 16578 times more Fibonacci numbers (maximum 2^{24} numbers) than can be input when using the G and IM methods (maximum 1012) to solve SLAE, provided that the SLAE are solved in a 'reasonable time'.

The greater efficiency of the IFFT method can be illustrated by the graph:

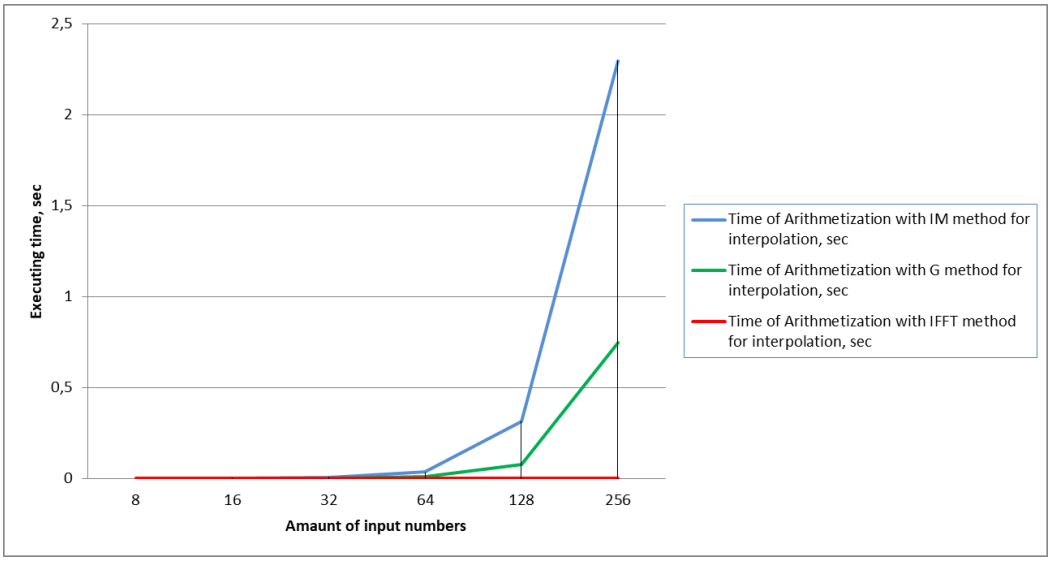

Figure 2 – depency of a time from amount of input numbers.

In the graph, the Ox axis means the average runtime of the program for 1000 measurements, the Oy axis means the number of Fibonacci numbers that are input in the program, the blue colour indicates the runtime of the Arithmetization variant that uses the IM method for the interpolation procedure, the green colour indicates the runtime of the Arithmetization variant that uses the G method for the interpolation procedure, and the red colour indicates the runtime of the Arithmetization variant that uses the IFFT method for the interpolation procedure. At the same time, calculations in the Arithmetization variants that use the G and IM methods for interpolation are performed in the GF field (257), and calculations in the Arithmetization variant that uses the IFFT method are performed in the Goldilocks field, i.e. in GF $(2^{64}-2^{32}+1)$.So, calculations using IFFT for 256 input Fibonacci numbers (256 unknowns and 256 equations) are at least 176 times faster, provided that the values of the unknown numbers in the Goldilocks field are greater in $\frac{2^{64}-2^{32}+1}{257}$ times than in the field GF (257).

It was determined from the tests that if the Arithmetization variant using the G or IM method is given the 8 Fibonacci number as an input, but the calculations are performed in the Goldilocks field, i.e. modulo P = $2^{64}-2^{32}+1$, than this option ran for 40 minutes without a result, which is 1 545 946 times longer than the IFFT option for interpolation.

Using the linear regression method, we have shown that the programmed version of Arithmetization, which uses the IFFT method for interpolation, does indeed have a time complexity $O(n*\log_2(n))$:

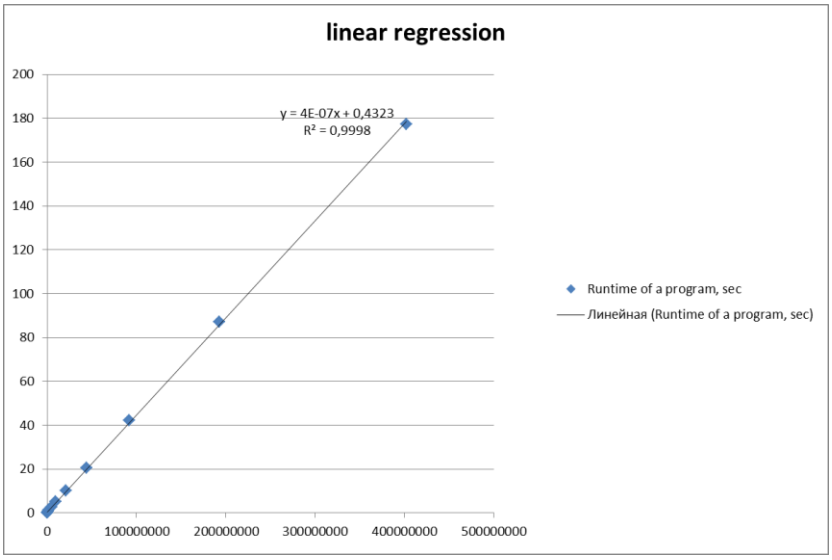

Figure 3 – Linear regression for IFFT method.

The numbers on the Ox axis are $n*\log_2(n)$, and the Oy axis shows the total time of the Arithmetization. In other words, testing confirms that using the IFFT method speeds up the Arithmetization stage compared to options that use the G and IM methods for interpolation. That is, Arithmetisation with IFFT has a time complexity of almost $O(n*\log_2(n))$, in contrast to the G and IM methods, which have time complexity $O(n^3)$.

Testing has shown that the maximum number of Fibonacci numbers that can be given into the input is 2^{24} numbers, so, it is possible to solve SLAEs of size 2^{24} unknowns and 2^{24} equations in an average time of 177.45 seconds (average time over 100 measurements).

With the help of the IFFT, it is possible to reduce the time of converting the input data into a polynomial. Also it is possible to reduce the percentage of time of converting the input data into a polynomial from the time of the entire first stage. Thanks to the IFFT, the percentage of time decreased from 99.99 % (for 1012 Fibonacci numbers) to 58.56 % (for 2^{24} Fibonacci numbers).

Using the IFFT method to speed up Prover's work in Arithmetisation, you can speed up in $\frac{n^3}{n*\log_2(n)}$ times the time it takes to check for the correctness of a blockchain network transaction and save time and hardware resources.

6. Conclusions

The paper carries out mathematical modelling and testing of the programmed first stage of the ZK-STARK Arithmetisation protocol, which is promising and relevant, which necessitates the study of its functioning and ways of implementing it in the blockchain network.

1) It is determined the peculiarities of the functioning of the first stage of the ZK-STARK protocol - the Arithmetisation stage: it begins with the transformation of input data into a polynomial form using the interpolation polynomial. This polynomial is then tested for low degree. If the polynomial has a degree of less than two, it passes the test and is considered reliable and accepted by Verifier.

2) A theoretical implementation of the first stage of the ZK-STARK protocol was created and three software versions were tested (the first stage of the protocol was programmed using the interpolation polynomial and the procedure for dividing two polynomials. The search for the coefficients of the interpolation polynomial was programmed using three methods with different speeds: the inverse matrix method, the Gaussian method, and the fast inverse Fourier transform) of the first stage of the protocol. Tests showed that the inverse fast Fourier transform was the most effective method for finding the coefficients of the interpolation polynomial. With the help of the IFFT, it is possible to solve SLAEs in which the maximum number of unknowns is 2^{24} and the equations is 2^{24} (in 16578 times more than

the IM and G methods) in 177.45 seconds in the Goldilocks field, which is in $\frac{2^{64}-2^{32}+1}{257}$ times bigger than the fields, on which Arithmetization based on IM and G methods was tested. The IM and G methods in the Goldilocks field failed to find the coefficients of the interpolation polynomial faster than the IFFT method.

3) Testing has shown that the practical implementation of Arithmetization based on IFFT has time complexity $O(n*\log_2(n))$. This is in $\frac{n^3}{n*\log_2(n)}$ times less than the time complexity of Arithmetization based on IM and G. IFFT speeds up the Arithmetization.

4) The most promising method among the three tested was the IFFT method, so it is recommended to use this method at the Arithmetization stage of the ZK-STARK protocol, as it will increase the effectiveness of the protocol and will speed up the authentication procedure, of which the Arithmetization stage is a part.

5) Solving a SLAEs with a large number of unknowns and equations is a very complex computational task. Therefore, if the activity of the ZK-STARK protocol leads to the solution of a SLAEs of size 2^{24}, or more, it makes sense to distribute the computations during the ZK-STARK authentication procedure among the computers of the blockchain computer network to speed up the first stage of the protocol, or to increase the capacity of individual network nodes, or to perform computations on a quantum computer, since the amount of data that computer networks need to process is growing every year.

Given the fact that the IFFT has reduced the percentage of time of converting input data into a polynomial, as a further step in the study of the promising modern ZK-STARK protocol, is the optimization of polynomial division (acceleration of polynomial division) can be used to further speed up the Verifier's work of the first stage of the ZK-STARK protocol and a software implementation of an even faster method than the IFFT is for finding the roots of an interpolation polynomial more faster.

СПИСОК ЛІТЕРАТУРИ

1. CONSENSYS. Zero-Knowledge Proofs: STARKs vs SNARKs / BLOCKCHAIN EXPLAINED: веб-сайт. – URL: https://consensys.net/blog/blockchain-explained/zero-knowledge-proofs-starks-vs-snarks/ (дата звернення: 30.04.2023).
2. Bit2Me ACADEMY. What are zk-STARKs?: веб-сайт. URL: https://academy.bit2me.com/en/what-are-zk-stark/ (дата звернення: 30.04.2023).
3. Chainlink. Understanding the Difference Between zk-SNARKs and zk-STARKS: веб-сайт. URL: https://chain.link/education-hub/zk-snarks-vs-zk-starks (дата звернення: 25.04.2024).
4. Ramses F. STARKs vs SNARKs [Електронний ресурс] // Medium: веб-сайт. URL: https://medium.com/@ramsesfv/starks-vs-snarks-d2e09c4e6069 (дата звернення: 25.04.2024).
5. Throne of ZK: SNARK vs STARK [Електронний ресурс] // Medium: веб-сайт. URL: https://medium.com/nonce-classic/throne-of-zk-snark-vs-stark-e449984d5c36 (дата звернення: 25.04.2024).
6. Techopedia. Zero-Knowledge STARK (zkSTARK): веб-сайт. URL: https://www.techopedia.com/definition/zero-knowledge-stark-zkstark (дата звернення: 25.04.2024).
7. Changelly Blog. New Technology Explained: Zero-Knowledge Proof – Security Enhancing Protocol: веб-сайт. URL: https://changelly.com/blog/zero-knowledge-proof-explained/ (дата звернення: 25.04.2024).
8. Hacken. Comparing ZK-SNARKs & ZK-STARKs: Key Distinctions In Blockchain Privacy Protocols: веб-сайт. URL: https://hacken.io/discover/zk-snark-vs-zk-stark/ (дата звернення: 25.04.2024).
9. Binance Academy. zk-SNARKs and zk-STARKs Explained: веб-сайт. URL: https://academy.binance.com/en/articles/zk-snarks-and-zk-starks-explained (дата звернення: 30.04.2023).
10. Binance Academy. zk-STARKs: веб-сайт. URL: https://academy.binance.com/en/glossary/zk-starks (дата звернення: 25.04.2024).

11. StarkWare. Arithmetization I // Medium: веб-сайт. URL: https://medium.com/starkware/arithmetization-i-15c046390862 (дата звернення: 25.04.2024).
12. Polygon Labs. Introducing Plonky2 // Polygon: веб-сайт. URL: https://polygon.technology/blog/introducing-plonky2 (дата звернення: 25.04.2024).
13. StarkWare Team. ethSTARK Documentation Version 1.1 // IACR: веб-сайт. URL: https://eprint.iacr.org/2021/582.pdf (дата звернення: 25.04.2024).
14. H. P. William, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes in C++. The Art of Scientific Computing, 2nd ed: Cambridge University Press, Cambridge, 2003, 1004 с.
15. Державний університет "Житомирська політехніка" // Освітній портал: веб-сайт. URL: https://learn.ztu.edu.ua (дата звернення: 25.04.2024).
16. Літнарович Р.М. Алгебра матриць: Курс лекцій. – Рівне: МЕГУ, 2007. – 112 с. веб-сайт. URL: https://core.ac.uk/download/pdf/14034615.pdf (дата звернення: 25.04.2024).
17. Кузнецов О.О. Конспект лекцій по математичним основам криптографічних доказів з нульовим розголошенням. Лекція №22: Дискретне перетворення Фур'є: PROXIMA LABS, 1501 Larkin Street, Suite 300, San Francisco, USA, 13 с.
18. The Fast Fourier Transform (FFT): Most Ingenious Algorithm Ever? // Youtube: веб-сайт. URL: https://www.youtube.com/watch?v=h7apO7q16V0 (дата звернення: 25.04.2024).
19. Кузнецов О.О. Конспект лекцій по математичним основам криптографічних доказів з нульовим розголошенням. Лекція №23: Дискретне перетворення Фур'є в кінцевих полях: PROXIMA LABS, 1501 Larkin Street, Suite 300, San Francisco, USA, 13 с.

REFERENCES

1. CONSENSYS, "Zero-Knowledge Proofs: STARKs vs SNARKs," BLOCKCHAIN EXPLAINED, [Online]. Available: https://consensys.net/blog/blockchain-explained/zero-knowledge-proofs-starks-vs-snarks/. [Accessed: 30-Apr-2023].
2. Bit2Me ACADEMY, "What are zk-STARKs?," [Online]. Available: https://academy.bit2me.com/en/what-are-zk-stark/. [Accessed: 30-Apr-2023].
3. Chainlink, "Understanding the Difference Between zk-SNARKs and zk-STARKS," [Online]. Available: https://chain.link/education-hub/zk-snarks-vs-zk-starks. [Accessed: 25-Apr-2024].
4. Ramses F., "STARKs vs SNARKs," Medium, [Online]. Available: https://medium.com/@ramsesfv/starks-vs-snarks-d2e09c4e6069. [Accessed: 25-Apr-2024].
5. "Throne of ZK: SNARK vs STARK," Medium, [Online]. Available: https://medium.com/nonce-classic/throne-of-zk-snark-vs-stark-e449984d5c36. [Accessed: 25-Apr-2024].
6. Techopedia, "Zero-Knowledge STARK (zkSTARK)," [Online]. Available: https://www.techopedia.com/definition/zero-knowledge-stark-zkstark. [Accessed: 25-Apr-2024].
7. Changelly Blog, "New Technology Explained: Zero-Knowledge Proof – Security Enhancing Protocol," [Online]. Available: https://changelly.com/blog/zero-knowledge-proof-explained/. [Accessed: 25-Apr-2024].
8. Hacken, "Comparing ZK-SNARKs & ZK-STARKs: Key Distinctions In Blockchain Privacy Protocols," [Online]. Available: https://hacken.io/discover/zk-snark-vs-zk-stark/. [Accessed: 25-Apr-2024].
9. Binance Academy, "zk-SNARKs and zk-STARKs Explained," [Online]. Available: https://academy.binance.com/en/articles/zk-snarks-and-zk-starks-explained. [Accessed: 30-Apr-2023].
10. Binance Academy, "zk-STARKs," [Online]. Available: https://academy.binance.com/en/glossary/zk-starks. [Accessed: 25-Apr-2024].
11. StarkWare, "Arithmetization I," Medium, [Online]. Available: https://medium.com/starkware/arithmetization-i-15c046390862. [Accessed: 25-Apr-2024].
12. Polygon Labs, "Introducing Plonky2," Polygon, [Online]. Available: https://polygon.technology/blog/introducing-plonky2. [Accessed: 25-Apr-2024].
13. StarkWare Team, "ethSTARK Documentation Version 1.1," IACR, [Online]. Available: https://eprint.iacr.org/2021/582.pdf. [Accessed: 25-Apr-2024].
14. H. P. William, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes in C++. The Art of Scientific Computing, 2nd ed: Cambridge University Press, Cambridge, 2003, 1004 с.

15. Державний університет "Житомирська політехніка", "Освітній портал," [Online]. Available: https://learn.ztu.edu.ua. [Accessed: 25-Apr-2024].
16. Р. М. Літнарович, *Алгебра матриць: Курс лекцій*, Рівне, Україна: МЕГУ, 2007, 112 с. [Online]. Available: https://core.ac.uk/download/pdf/14034615.pdf. [Accessed: 25-Apr-2024].
17. О. О. Кузнецов, "Конспект лекцій по математичним основам криптографічних доказів з нульовим розголошенням. Лекція №22: Дискретне перетворення Фур'є," PROXIMA LABS, 1501 Larkin Street, Suite 300, San Francisco, USA, 13 с.
18. "The Fast Fourier Transform (FFT): Most Ingenious Algorithm Ever?," Youtube, [Online]. Available: https://www.youtube.com/watch?v=h7apO7q16V0. [Accessed: 25-Apr-2024].
19. О. О. Кузнецов, "Конспект лекцій по математичним основам криптографічних доказів з нульовим розголошенням. Лекція №23: Дискретне перетворення Фур'є в кінцевих полях," PROXIMA LABS, 1501 Larkin Street, Suite 300, San Francisco, USA, 13 с.

О. Ю. Аверков *Студент магістерської програми, кафедри безпеки інформаційних систем і технологій, Факультет комп'ютерних наук ХНУ імені В.Н. Каразіна, майдан Свободи 4, Харків, Україна, 61022; e-mail:* olegaverkov072@gmail.com;

О. О. Кузнецов *Доктор технічних наук; професор кафедри безпеки інформаційних систем і технологій, Факультет комп'ютерних наук ХНУ імені В.Н. Каразіна, майдан Свободи 4, Харків, Україна, 61022; e-mail:* kuznetsov@karazin.ua;

І. В. Лисицька *Доктор технічних наук; професор кафедри безпеки інформаційних систем і технологій, Факультет комп'ютерних наук ХНУ імені В.Н. Каразіна, майдан Свободи 4, Харків, Україна, 61022;* e-mail: ivlisitska@karazin.ua;

Теоретична реалізація та тестування першого етапу роботи протоколу ZK-STARK «Арифметизація»

Актуальність: З появою Інтернету світ почав стрімко змінюватися, до того ж темп змін постійно зростає, тому проблема збереження та обробки даних стає все більш актуальною. Протокол ZK-STARK - це новий криптографічний протокол, який ще не має масового застосування на практиці та дозволяє перевірити повідомлення чи транзакцію в мережі Блокчейн на достовірність, не відтворюючи її повністю. На даний момент, виявлені прогалини та проблеми, пов'язані з його застосуванням: обчислювальна складність, можлива погана сумісність з іншими протоколами, стійкість до атак з боку квантових компп'ютерів. Тому у роботі вирішено доповнити висвітлення проблеми, пов'язаної з обчислювальною складністю та запропонувати варіанти вирішення цієї проблеми.

Мета: на основі теоретичної реалізації першого етапу Arethmetization роботи протоколу ZK-STARK провести тестування його програмної реалізації задля надання рекомендацій щодо його найбільш обчислювально-ефективної версії.

Методи дослідження: математичні відомості з теорії інтерполювання, теорії груп, теорії чисел; відомості про числа Фібоначчі; відомості про функцію Ейлера; породжуючий елемент групи; циклічні групи; інтерполяційний поліном Лагранжа та послідовність обчислень, також середа програмування Visual Studio 2022, мова програмування C++, бібліотека NTL, Microsoft Excel.

Результати: результатом роботи є теоретична реалізація роботи першого етапу протоколу ZK-STARK та тестування на ефективність першого етапу, та надання рекомендацій щодо його найбільш ефективного версії.

Висновок: Тестування показало, що практичне впровадження Арифметизації на основі ШЗПФ має часову складність n*log(n), яка у (n^3)/(n*log(n)) разів менша, ніж часова складність Арифметизації на основі ЗМ та Г, що прискорює роботу етапу Арифметизація протоколу ZK-STARK.

Keywords: protocol, ZK-STARK, ARETHMETIZATION, modeling, program, implementation, C++ programming language, testing, blockchain, efficiency.

Надійшла у першій редакції 07.08.2024, в останній - 11.09.2024.
The first version has been received on 07.08.2024, the final version - on 11.09.2024.