DOI: 10.15514/ISPRAS-2019-1(2)-1

Разработка системы тестирования точности и эффективности библиотек гомоморфного шифрования для рациональных чисел с фиксированной мантиссой

[1] *Е.М. Ширяев,* ORCID: 0000-0002-2359-1291 <*eshiriaev@ncfu.ru*>
[1] *М.Г. Бабенко,* ORCID: 0000-0001-7066-0061 <*mgbabenko@ncfu.ru*>
[1] *Е.С. Безуглова,* ORCID: 0000-0002-7608-0452 <*eksbezuglova@ncfu.ru*>
[1] *М.А. Лапина,* ORCID: 0000-0001-8117-9142 <*mlapina@ncfu.ru*>
[1] *Северо-Кавказский Федеральный Университет, 355017, Россия, г. Ставрополь, ул. Пушкина, д. 1.*

Аннотация. По мере увеличения объема данных, обрабатываемых с использованием облачных технологий, конфиденциальность данных как в общедоступных, так и в частных облачных средах становится все более уязвимой для различных угроз безопасности. Для решения этой проблемы были предложены схемы полностью гомоморфного шифрования в качестве потенциального решения для повышения конфиденциальности данных в облачных вычислениях. Схемы полностью гомоморфного шифрования позволяют пользователям выполнять вычисления с зашифрованными данными, тем самым защищая конфиденциальность обрабатываемой информации. Однако, несмотря на потенциальные преимущества полностью гомоморфного шифрования, его широкое внедрение остается сложной задачей из-за его вычислительной сложности. Исследователи активно изучают применение этой технологии в различных областях, а группы ученых разрабатывают библиотеки алгоритмов полностью гомоморфного шифрования, которые обладают различными качествами и характеристиками. В этой статье мы предлагаем концепцию системы тестирования, основанной на точности и эффективности обработки данных за счет использования быстрого преобразования Фурье, выполняемого в зашифрованном формате, с использованием двух наиболее широко известных библиотек полностью гомоморфного шифрования для рациональных чисел с фиксированной мантиссой. Предлагаемая система позволяет анализировать библиотеки на основе их пригодности для использования в различных приложениях с учетом требований как к точности, так и к эффективности обработки. В будущем планируется дальнейшее развитие процесса тестирования путем включения показателей использования памяти, менее популярных библиотек и целочисленных схем.

Ключевые слова: Гомоморфное Шифрование; CKKS; Быстрое Преобразование Фурье; Облачные Вычисления; Точность Вычислений; Скорость Вычислений.

Для цитирования: Ширяев Е.М., Бабенко М.Г., Безуглова Е.С., Лапина М.А. Разработка системы тестирования точности и эффективности библиотек гомоморфного шифрования для рациональных чисел с фиксированной мантиссой. Труды ИСП РАН, том 1, вып. 2, 2019 г., стр. 15–19. DOI: 10.15514/ISPRAS–2019–1(2)–1.

Благодарности: Работа выполнена при поддержке Российского научного фонда 19-71-10033, https://rscf.ru/project/19-71-10033/.

Development of a system for testing the accuracy and efficiency of homomorphic encryption libraries for rational numbers with a fixed mantissa

[1] *E.M. Shiriaev* ORCID: 0000-0002-2359-1291 <*eshiriaev@ncfu.ru*>
[1] *M.G. Babenko* ORCID: 0000-0001-7066-0061 <*mgbabenko@ncfu.ru*>
[1] *E.C. Bezuglova* ORCID: 0000-0002-7608-0452 <*eksbezuglova@ncfu.ru*>
[1] *M.A. Lapina* ORCID: 0000-0001-8117-9142 <*mlapina@ncfu.ru*>
[1] *North Caucasus Federal University,1, Pushkin st., Stavropol, 355017, Russia.*

Abstract. As the volume of data processed using cloud technologies increases, data privacy in both public and private cloud environments is becoming increasingly vulnerable to various security threats. To solve this problem, fully homomorphic encryption schemes have been proposed as a potential solution to enhance data privacy in cloud computing. Fully homomorphic encryption schemes allow users to perform calculations with encrypted data, thereby protecting the confidentiality of the information being processed. However, despite the potential benefits of fully homomorphic encryption, its widespread adoption remains a challenge due to its computational complexity. Researchers are actively exploring the application of this technology in various fields, and groups of scientists are developing libraries of fully homomorphic encryption algorithms that have various qualities and characteristics. In this article, we propose the concept of a testing system based on the accuracy and efficiency of data processing through the use of fast Fourier transform performed in an encrypted format using two of the most widely known libraries of fully homomorphic encryption for rational numbers with a fixed mantissa. The proposed system allows analyzing libraries based on their suitability for use in various applications, taking into account the requirements for both accuracy and processing efficiency. In the future, it is planned to further develop the testing process by including memory usage indicators, less popular libraries and integer schemes.

Keywords: Homomorphic Encryption; CKKS; Fast Fourier Transform; Cloud Computing; Computational Accuracy; Computational Speed.

For citation: Shiriaev E.M., Babenko M.G., Bezuglova E.S., Lapina M.A. Development of a system for testing the accuracy and efficiency of homomorphic encryption libraries for rational numbers with a fixed mantissa. *Trudy ISP RAN/Proc. ISP RAS*, vol. 1, issue 2, 2019. pp. 15-19 (in Russian). DOI: 10.15514/ISPRAS-2019-1(2)-1.

Acknowledgements. This work was supported by the Russian Science Foundation 19-71-10033, https://rscf.ru/project/19-71-10033/.

1. Introduction

The performance of both central and GPU processors is steadily increasing. This fact leads to the fact that computing devices face problems such as a wall of memory and a wall of power, when productivity growth begins to decrease and eventually stops with a decrease in the process and an increase in the number of transistors on a chip [1]. This is due to a mismatch between the processor's processing speed and the memory response time. This fact is one of the reasons for the spread of distributed computing beyond research and government enterprises into the public sphere of activity. Thus, since the beginning of the XXI century, the concept of providing computing resources in the form of a service was born, which eventually led to cloud technologies.

Cloud technologies (CT) is a generalized term that covers concepts such as cloud storage, cloud computing, etc., which have one common feature – the technology is based on a distributed system where nodes are geographically remote from each other and connected to each other over the Internet [2]. n addition, in addition to productivity growth, the volume of processed data is also growing, which has led to the so-called Big Data [3]. CT allows computationally complex technologies to be applied to more consumers, such as artificial intelligence technologies. CT is used for training artificial neural networks (NN) [4]. When training NN, it is necessary to process datasets a large

number of times, which, with a large number of neurons and large datasets, requires a lot of computing resources. The popularity of NN has also led to the popularity of renting CT for training neural networks and the interest of scientific groups in this process [5], [6].

NN, is one example of the application of CT, they are also used in other various fields. As in medicine, for example. In [7] researchers propose an open cloud for drug management. In another paper [8] , the authors show the use of CT for medical research, namely three-dimensional ultrasound examination. CT is also used in various government sectors. For example, in [9], the authors show a CT-based traffic assessment system. CT is also used in the field of finance, for example, in [10] the authors explore a cloud platform for the needs of banking structures. Another example is already real solutions in the field of banking, such as the Russian bank "Sber" provides cloud services in its "Sber Cloud" (until May 2022, after "Cloud.ru" [11]), in addition, Sber has developed its ecosystem based on CT [12]. In addition to Sber, there are other CT ecosystems on the Russian market. For example, there is an ecosystem of Yandex [13], VK [14] and MTC [15]. Amazon can be cited as global examples [16].

In this case, an ecosystem is understood as a set of interconnected services of one company, where cloud solutions are used not only for storage and computing, but also for building the infrastructure of all customer data, as well as between them [17]. Such ecosystems are a convenient tool from the client's point of view, as they simplify their access to various services. However, such ecosystems contain a large amount of confidential customer data, which poses a threat to their security. For example, in 2023, 420 leaks of databases of Russian companies were recorded [18]. his situation has developed due to the fact that data in large cloud structures is subject not only to external threats, but also internal ones. Building a complex security system against external threats can lead to internal problems, such as failures, application synchronization errors, as well as the prior collusion of compromised employees [19]. End-to-end encryption has not justified most of the hopes placed on it, due to the increased danger from the inside in the absence of proper control during the development of the system to errors and possible backdoors [20].

To increase data privacy in cloud systems, there are several ways to solve this problem. In any case, the system developer will have to choose a balance between speed, quality of service and safety. In the scientific environment, methods based on Secret Sharing Schemes (SSS) are quite popular [21], if we consider Google Scholar, then over the past 10 years, when searching for the keywords "Secret Sharing System", "Cloud Technology", "Cloud Services", "Cloud Computing" and "Cloud Storage", the system outputs more than 10,000 works. Both perfect SSS and threshold SSS are used to ensure safety [21]. In addition, SSS based on the Residue Number System (RNS) are popular, such as the Mignotte Scheme [22], as well as Asmut-Bloom [23]. Based on SSS-RNS, there are, for example, the following solutions [24], [25].

However, using SSS, it is possible to increase the security of access to the cloud structure based on authenticated users, i.e. complete confidentiality is not ensured. To ensure complete confidentiality, it is possible to use the so-called Fully Homomorphic Encryption (FHE) [26]. FHE allows you to process information in encrypted form, based on the operations of homomorphic addition and homomorphic multiplication of ciphertexts. However, even since 2009, the efficiency of FHE schemes has increased significantly due to various mathematical techniques [27], [28], [29], FHE still remains a computationally complex encryption method. Many researchers have developed a variety of FHE libraries, such as SEAL [30], OpenFHE [31], etc. However, although all schemes meet safety standards [32] their technical implementation is quite diverse and has different characteristics, speed, accuracy, etc. For example, in [33] a comparison of the implementations of the SEAL and Palisade libraries (currently OpenFHE) was carried out, however, it was carried out in the field of one operation – matrix multiplication, which is difficult to use as standardized testing.

If we consider high-performance computing, then there is a set of tests aimed at determining the performance of supercomputers. They include tests such as LINPACK [34] and HPC Challenge Benchmark [35]. Two of these test suites can be distinguished. This is the solution of a dense system of linear algebraic equations (SLAE) by the LUP decomposition method [36] and a test for

calculating the fast Fourier transform (FFT) [37]. If the SLAE solution in the case of FHE is not suitable due to the requirement to perform a division operation, then FFT can be calculated using addition and multiplication operations. The calculation of FFT by a specific library can show a lot of different information. To the extent that the organization of library functions and variables is capable of processing recursions, overwriting and changing ciphertexts, performing bootstrapping operations, etc., In fact, these criteria relate to the speed of data processing, in addition, FFT will allow you to analyze the accuracy of the operations performed. Considering the fact that FFT involves the processing of rational complex numbers, the CKKS scheme is a suitable environment for research [38].

Thus, the purpose of this work is to conduct research related to the performance of FHE libraries, namely, to develop a test that will evaluate the performance and accuracy of a particular library. The positive results of this work will allow us to capture more libraries and characteristics of FHE libraries in the future and develop a methodology for evaluating FHE performance, which will help FHE developers and researchers evaluate the performance of new FHE circuits and libraries.

The work is organized as follows: section 2 provides more detailed information about FHE and the CKKS scheme, section 3 discusses FHE testing in more detail, section 4 provides the encrypted FFT algorithm, section 5 presents the main results of the work, section 6 provides conclusions on the work done, as well as a description of further research.

2. Fully Homomorphic Encryption

FHE is an encryption method that allows you to perform arithmetic operations on encrypted numbers. FHE was first presented in 2009 by Gentry in his dissertation [26], however, homomorphic encryption itself has been known for a long time. Cipher schemes that are capable of performing homomorphic addition or homomorphic multiplication over encrypted numbers appeared in the last century, examples of such schemes are RSA [39] and El-Gamal [40]. Since 2009, many schemes based on the Genty scheme have been developed. Which were originally called Somewhat Homomorphic Encryption, due to the strict restrictions imposed on the number of multiplication operations. However, with the development of operations such as bootstrapping [28] and rescaling[41], the restriction has become less stringent, the secret key is modified and the number of allowed multiplications increases, however, this is a rather computationally complex operation and requires more and more resources each time.

Initially, FHE supported either Boolean values [42], or integer values [43]. Currently, there is a scheme FHE Homomorphic Encryption of an approximate numbers (HEAAN) or, according to the names of the authors, CKKS [38] that supports rational numbers with a fixed mantissa. This scheme expands the scope of application of FHE.

CKKS is a homomorphic encryption system designed to efficiently perform approximate arithmetic operations on encrypted data. It is especially suitable for calculations involving real or complex numbers over the field $C^{N/2}$. The plaintext space and the ciphertext space have the same area.

$$Z_Q[X]/(X^N + 1)$$

where N – degree of two.

Batch encoding CKKS $C^{\frac{N}{2}} \leftrightarrow Z_Q[X]/(X^N + 1)$ maps an array of complex numbers to a polynomial with the property: $decode\big(encode(m_1) \otimes encode(m_2)\big) \approx m_1 \odot m_2$, where $\otimes$ is a multiplication of components, and $\odot$ is a non-cyclic convolution.

2.1 CKKS scheme

The CKKS scheme supports the standard recommended parameters [32], selected to provide 128-bit security for a single ternary private key $s \in_u \{-1,0,1\}^N$, according to the group of homomorphic

encryption standards. Encoding in CKKS is performed in the field of complex numbers using Lagrange polynomials.

The scheme uses approximate arithmetic to construct ciphertexts. Let's consider this arithmetic. To do this, the base $p > 0$ and the module q_0, are fixed, as well as $q = p \cdot q_0$ at $0 < l \leq L$. An integer p will be used as a basis for scaling in approximate calculations. As a security setting λ a parameter is selected such that $M = M(\lambda, q_L)$ for a polynomial ring.At the boundaries $0 < l \leq L$ of the ciphertext level l a vector is defined in $\mathcal{R}_{q\ell}^{k}$ for a fixed integer k.

1) **Key generation:** The encryption process begins with the generation of keys, which are public pk and private sk keys. The private key is used to decrypt the data, while the public key is used to encrypt it.
2) **Encryption**: To encrypt the plaintext vector x, the following steps are performed:

- **Padding**: The vector $m(x)$ is filled with zeros, the length of the vector is equal to the specified power of two N.
- **Encoding**: The plaintext vector x is encoded into the plaintext polynomial $m(x)$, which is a polynomial representation of the message.
- **Homomorphic Encryption**: The polynomial $m(x)$ is encrypted using pk to obtain the polynomial $c(x)$ of the ciphertext, while it is necessary to control the error value e, which satisfies $|e|_{\infty}^{can} \leq e_{Max}$, at which the expression $\langle c, sk \rangle = m + e_{Max} (mod\ q_L)$.

3) **Decryption**: To decrypt the polynomial $c(x)$ of the ciphertext, the following steps are performed:

- **Homomorphic Decryption**: The ciphertext polynomial $c(x)$ is decrypted using the secret key to obtain the plaintext polynomial $m(x) \leftarrow \langle c, sk \rangle (mod\ q_l)$.
- **Decoding**: To obtain the original text vector x the text polynomial $m(x)$ is converted back from a polynomial to a message.

4) **Homomorphic Operations**: CKKS supports several approximate arithmetic operations with encrypted data, including addition and multiplication. Homomorphic addition and multiplication can be performed in the ciphertext space without the need to decrypt the ciphertext.

- **Homomorphic Addition**: Taking into account the two ciphertexts $c_1(x)$ and $c_2(x)$, representing the encrypted values $m_1(x)$ and $m_2(x)$ respectively, the homomorphic addition is performed by adding the corresponding coefficients modulo the source text of the module: $c(x) = c_1(x) + c_2(x)$, in addition, errors e_1 and e_2 are also summed.
- **Homomorphic Multiplication**: Taking into account the two ciphertexts $c_1(x)$ and $c_2(x)$, representing the encrypted values $m_1(x)$ and $m_2(x)$, respectively, homomorphic multiplication transforms the ciphertext of polynomials modulo the source text of the module: $c(x) = c_1(x) \times c_2(x)$, for multiplication, the proper error bounds $e_{mult} \in \mathcal{R}$ с $|e_{mult}|^{can}{}_{\infty}\ e_{multMax}(l)$ are allocated, where $e_{multMax}(l)$ is a given constant.

Both addition and multiplication lead to an increase in the approximation error e, the CKKS scheme is able to decrypt data when the error is in certain aisles. When using the CKKS scheme, it is important to control the error growth, which depends on the number of operations and their order. Given the peculiarities of arithmetic, multiplication makes a big mistake. Different software implementations of the CKKS scheme offer different ways to control them.

2.2 Fully Homomorphic Encryption Libraries

There are several FHE libraries. All of them are implemented in different programming languages and support a different set of schemes, a detailed table with the characteristics of the schemes, for example, is presented in [33]. Many of them are implemented in the C++ programming language, this is due to the fact that this language supports most platforms, has memory protection and allows

you to use inserts from low-level code, as well as analyze code performance in detail. If we consider libraries that implement the CKKS scheme, then in addition to SEAL, OpenFHE and Helib, there is also HEaaN [44], a library developed by the authors of the CKKS scheme. In this paper, we will focus on SEAL and OpenFHE. This choice was made because these libraries, in addition to being developed in C++, also have Python implementations. Which expands their application. Python has developed a large toolkit for working with NN, machine learning and data analysis. Given that confidential data is also processed using these technologies, FHE can find applications in them. Moreover, at the moment there is at least one library [45] for working with artificial intelligence methods with FHE, this library uses SEAL for FHE methods, TenzorFlow for organizing data in the form of tensors and PyTorch as a tool for building NN. However, there are several disadvantages in this library: the library does not support encrypted training of neural networks, matrix multiplication does not occur in encrypted form. Porting the library from C++ to Python could lead to a drop in performance and accuracy, in addition, although the libraries themselves are based on the same set of FHE schemas, they are organized completely differently.

The library development paradigm itself has an impact here. OpenFHE is a ready-made toolkit for working with FHE. In most cases, the user only needs to set the cipher parameters, and often boundary ones, and call the necessary functions (for example, addition, multiplication, subtraction, etc.). If the user needs to apply Bootstrapping, he needs to add the appropriate parameters and call the function. SEAL involves more subtle work. If in the case of OpenFHE, the user sets the value of the multiplicative depth [26], then in the case of SEAL, the user needs to select the parameters so that the required multiplicative depth is obtained based on them. In addition, OpenFHE itself determines the set of primes and their number based on a given boundary. For SEAL, you need to set the number and size of modules yourself. Also in SEAL, the user is required to control the CKKS approximation error, the key and other cipher parameters himself after each operation. So we have two fundamentally different libraries. On the one hand, OpenFHE is easy to use and allows you to implement a secure homomorphic system, on the other hand, SEAL allows you to fine-tune each parameter and organize calculations in the required way, which, with sufficient user skills, can increase the speed of data processing while complying with the security standard.

Based on all the above, it follows that conducting a study of the performance and accuracy of even two libraries will allow you to obtain scientifically significant data, and studying their implementations in Python will allow you to study the influence of the programming language scheme on the characteristics of the library, which will allow you to draw conclusions about which library and in which programming language should be used in one or another the application.

3. Performance Testing

Performance testing is a process that is carried out in order to determine how fast a computing system or part of it is running under a certain load. It can also serve to validate and validate other attributes of system quality, such as scalability, reliability, and resource consumption. The most widely known is high-performance computing (HPC) testing. HPC is the practice of combining computing power to achieve higher performance than conventional computers or servers. HPC, or super-calculations, are no different from conventional calculations, except for power. This is a way to process huge amounts of data at very high speeds. To do this, several computers and storage devices are combined into a single system. HPC, in fact, has a lot in common with CT, since CT implies combining geographically remote computing centers into a common system. The use of FHE, as mentioned above, allows you to process information in encrypted form, so when using FHE, the arithmetic of either the entire CT system or part of it changes. The modified arithmetic already has other qualitative indicators of computing power, which requires testing to determine the capabilities of the system being developed. As mentioned earlier, LINPACK and HPC Challenge Benchmark performance tests are the most popular for HPC.

The performance in the LINPACK test provides information to clarify the peak performance claimed by the computer manufacturer (peak performance is the maximum theoretical performance that a computer can achieve, calculated as the product of the processor clock frequency by the number of operations performed per clock cycle). The actual performance will always be lower than the maximum. The performance measured by the LINPACK benchmark shows the number of operations on 64-bit floating point numbers (additions and multiplications) that the computer performed per second. This ratio is referred to as "FLOPS". However, computer performance when working with real applications is likely to be significantly lower than the maximum performance achieved when running the LINPACK test. LINPACK was designed to help users estimate the time it takes their computer systems to solve problems using the LINPACK package. To do this, the performance results obtained on 23 different computers solving a problem with a matrix size of 100 by 100 elements are extrapolated. This task size was chosen considering the characteristic memory sizes and processor performance in that era: 10,000 floating-point numbers with a value from -1 to 1 are randomly generated to fill in a common dense matrix, then the duration of the LU decomposition is measured with partial rotation. Over the following years, additional versions were released with increased different task sizes, for example, with matrices of 300 by 300 and 1000 by 1000 numbers. There are also implementations using hardware capabilities to accelerate matrix-vector and matrix-matrix operations.

However, from the point of view of FHE, testing using the LINPACK package is currently ineffective in terms of the limitations imposed on calculations in FHE. The SLAE solution method used in LINPACK uses a large number of matrix multiplications. As the results of the matrix multiplication study in [33], have shown, one matrix with a size of 20×20 already requires more than 20 GB of RAM. However, with the improvement of the matrix multiplication algorithm, it will also be possible to test using LINPACK with the appropriate modification of the LINPACK package itself to work with FHE.

The HPC Challenge Benchmark also contains a text for solving dense SLAE using the LUP decomposition method, however, in addition to it, it also contains a test based on FFT. (other tests aimed at memory bandwidth are not considered in this paper due to their value for the HPC hardware component). From the point of view of FHE – FTT is the most suitable option due to the fact that it manipulates the terms of the vector, in addition, FTT involves processing complex numbers, which is an advantage in the case of the CKKS scheme. Thus, in this paper, a performance test of FHE libraries will be developed based on the calculation of FFT in encrypted form. The developed test will also allow you to measure the accuracy of the result.

4. Fast Fourier Transform

In addition to the testing benefits described in the previous section, FTT itself has many applications in various applications. Let's look at FTT in more detail. FFT is an accelerated version of the Discrete Fourier Transform (DFT) [46]. DFT is one of the Fourier transforms that is used in digital signal processing algorithms, in addition, they have found wide application in other areas, such as audio compression in MP3, JPEG image compression. In addition, DFT is used in solving partial differential equations [47], as well as for performing the convolution operation [48]. DFT uses trigonometric functions as well as exponential in its calculations, which complicates the hardware processing of DFT. In order to speed up data processing, FFT was developed, the algorithm allowed reducing the computational complexity from $O(N^2)$ to $O(N\log(N))$. In general, the algorithm that converts $(a_1, a_2, \dots a_{n-1})$ to $(b_1, b_2, \dots, b_n)$ can be described by the following formula:

$$b_i = \sum_{j=0}^{q-1} \varepsilon_N^{ij} \left(\sum_{k=0}^{p-1} a_{kq+j} \varepsilon_p^{ki} \right),$$

where $\varepsilon = e^{\frac{2\pi i}{n}}$, $N = pq, p > 1, q > 1, 0 < k < 1$ and $i = \overline{0, p-1}$. The algorithm splits the total number of points $a \in \overline{1, N}$ into p and q points and processes them in parallel, p and q are also split into two parts until the number of processed values becomes $n = 2$.

FFT also finds its application in the field of artificial intelligence, such as for performing convolution in convolutional neural networks [48], and solving partial differential equations is used in NN learning algorithms [49]. In addition, there are other FFT applications in NN [50], [51], [52]. Thus, the development of the confidential FFT algorithm will also expand the use of NN in the field of confidential data processing.

The FFT algorithm is not encrypted and consists of the following:

Algorithm 1. Fast Fourier Transform

Input: $a = \{a_1, a_2, \dots, a_N), \varepsilon, N$

Output: $b = \{b_1, b_2, \dots, b_n\}$

1. *if* $N \leq 1$ *do:*
1.1. ***return*** b
2. $h\_n = N/n$
3. $even = \{h\_n\}$
4. $odd = \{h\_n\}$
5. *for i from* 0 *to h_n do:*
5.1. $even_i = a_{2 \cdot i}$
5.2. $odd_i = a_{2 \cdot i+1}$
6. $even = Fast\ Fourier\ Transform(even, \varepsilon, h\_n)$
7. $odd = Fast\ Fourier\ Transform(odd, \varepsilon, h\_n)$
8. *for i from* 0 *to h_n do:*
8.1. $b_i = even_i + odd_i$
8.2 $b_{n-i-1} = (even_i - odd_i) \cdot \varepsilon$
9. ***return*** b

The algorithm uses recursions that are efficiently calculated in parallel, where the threads are connected sequentially. The libraries in question have functionality for parallel computing.

In the case of homomorphic encryption, in addition to the modified FFT algorithm, it is also necessary to implement an algorithm for initializing parameters. Despite the library differences. SEAL and OpenFHE using pseudocode, we can describe the general initialization of parameters for the CKKS scheme.

Algorithm 2. Encryption Parameters

Input: $degree$

Output: $Evaluator, Encryptor, Decryptor, Crypto\ context$

1. $parameters \leftarrow polynom\ modulus\ degree = degree$
2. $parameters \leftarrow set\ coefficient\ modulus =$
$= polynom\ modulus\ degree(60, 40, 40, 40, 60)$
3. $scale = 2^{40}$
4. $Crypto\ context \leftarrow parameters$
5. $Key\ generator \leftarrow Crypto\ context$
6. $Key\ generator \rightarrow Secret\ key$
7. $Key\ generator \rightarrow Public\ key$
8. $Key\ generator \rightarrow Relin\ key$

9. $Encryptor \leftarrow (Crypto\ context, public\ key)$
10. $Evaluator \leftarrow (Crypto\ context)$
11. $Decryptor \leftarrow (Crypto\ context, secret\ key)$
12. $Encoder \leftarrow (Crypto\ context)$
13. $\boldsymbol{return}\ Evaluator, Encryptor, Decryptor, Crypto\ context$

The above algorithm generates the parameters necessary to work with CKKS. In this algorithm, we indicate that the output is a complete tuple of CKKS parameters, but in the future, we will only work with $Evaluator$ and $Crypto\ context$. Since the $Evaluator$ contains all the necessary parameters to perform homomorphic operations, and $Crypto\ Context$ allows you to create "empty" vectors represented in CKKS. The remaining parameters are used for encryption and decryption, we omit these operations in our study, since they are trivial. Next, let's look at the algorithm for encrypted FFT calculation.

Algorithm 3. Encrypted Fast Fourier Transform

Input: $a = \{a_1, a_2, \dots, a_N), \varepsilon, N, Evaluator, Crypto\ context$

Output: $b = \{b_1, b_2, \dots, b_n\}$

1.*if* $N \leq 1$ *do:*
1.1. $\boldsymbol{return}\ b$
2. $h\_n = N/n$
3.$even = Crypto\ context \rightarrow \{h\_n\}$
4.$odd = Crypto\ context \rightarrow \{h\_n\}$
5. *for* i *from* 0 *to* h_n *do:*
5.1. $even_i = a_{2 \cdot i}$
5.2. $odd_i = a_{2 \cdot i+1}$
6. $even = Encrypted\ Fast\ Fourier\ Transform(even, \varepsilon, h\_n, Evaluator, Crypto\ context)$
7. $odd = Encrypted\ Fast\ Fourier\ Transform(odd, \varepsilon, h\_n, Evaluator, Crypto\ context)$
8. *for* i *from* 0 *to* h_n *do*:
8.1. $b_i = Evaluator \rightarrow Encrypted\_Addition(even_i, odd_i)$
8.2. $temp = Evaluator \rightarrow Encrypted\_Subtraction(even_i, odd_i)$
8.3 $b_{n-i-1} = Evaluator \rightarrow Encrypted\_Multiplication(temp \cdot \varepsilon)$
9. $\boldsymbol{return}\ b$

This algorithm differs from the original one in that vectors of encrypted numbers are processed. To do this, the $temp$ temporary variable is used, which stores the intermediate result of subtraction, the connected parameters $Crypto\ Context$ and $Evaluator$, are also used, they are necessary to create an empty length vector that meets the requirements of the created encryption scheme and perform homomorphic operations. Here it is worth noting the disadvantage of this algorithm. Given that the algorithm is recursive, we need to multiply the scheme parameters with each iteration of the recursion. However, this does not harm the security of the scheme, since the encryption keys are not in them, additional RAM is still spent, which must be controlled.

Thus, based on the above algorithms, a system for testing the performance and accuracy of the SEAL and OpenFHE libraries was developed. Next, let's look at the results of the conducted testing.

5. Experimental analysis

The testing system is organized as follows:

1) A vector of complex *input* numbers is created based on a sequence of pseudorandom numbers in the range from 0 to 1;
2) Using Algorithm 1, the FFT is calculated, the results are stored in vector x, to obtain a

reliable result, Algorithm 1 is run 10,000 times;

3) Using Algorithm 2, the encryption parameters for the CKKS scheme are set;
4) Based on the specified parameters, the input vector is encrypted into the c_input vector;
5) Using Algorithm 3, an encrypted FTP is calculated, the time spent on the calculation is measured, the results are stored in the vector c_x, to obtain a reliable result, Algorithm 3 is run 10,000 times;
6) The error is calculated as $\Delta x = |x - c\_x|$;
7) The output is supplied with Δx and the time spent on executing the algorithm in ms.

The developed testing system was launched on a computer with the following characteristics:

- Processor frequency: 1600–3400 MHz;
- Number of cores/threads: 4/8;
- Process technology: 14 nm;
- Architecture: x86;
- L1 cache: 256 KB;
- L2 Cache: 1 MB;
- L3 cache: 6 MB;
- Video memory capacity: 4 GB;
- RAM capacity: 16 GB;
- RAM Type: DDR4;
- ROM capacity: 256 GB;
- ROM Type: SSD.

To exclude the influence of various approaches to bootstrapping and key relinearization, the number of numbers in the vector is limited to 250. Let's consider the results obtained in the form of graphs.

5.1 Investigating the accuracy of fully homomorphic encryption libraries

First, let's look at the general graph (Fig. 1). For clarity, the graph is presented on a logarithmic scale. In general, SEAL and OpenFHE have shown good results in both programming languages. The overall accuracy of the result is obtained up to 7 decimal places. This is a good result, which is within the bounds for both NN [53], [54], and FFT [55]. If we consider the result exclusively within the framework of these libraries, then the implementation of SEAL in Python showed the worst accuracy result, while the implementation of SEAL in C++ shows the best result. At the same time, OpenFHE shows a stable error on both implementations. The result obtained can explain the peculiarities of the organization of variables in programming languages. In C++, long arithmetic is not initially implemented, the NTL [56] library is used to support it, , and in addition to NTL, GMP [57] is also used to speed up the processing of large numbers by SEAL and OpenFHE. In Python, long arithmetic is implemented initially. For C++, the developer can use NTL only when necessary, thereby increasing the processing speed of the values that the processor architecture supports. Python also has this feature, but the developer could have ignored it.

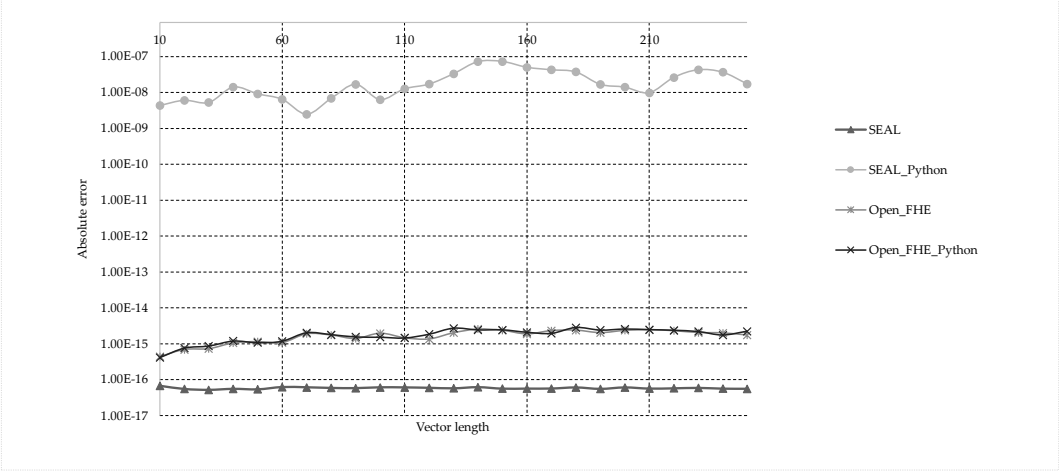

Fig. 1. The results of the study of the error of the encrypted FFT algorithm

For more clarity, let's look at an illustration of how many times the SEAL implementation is more accurate than the OpenFHE implementation in C++ (Fig. 2)

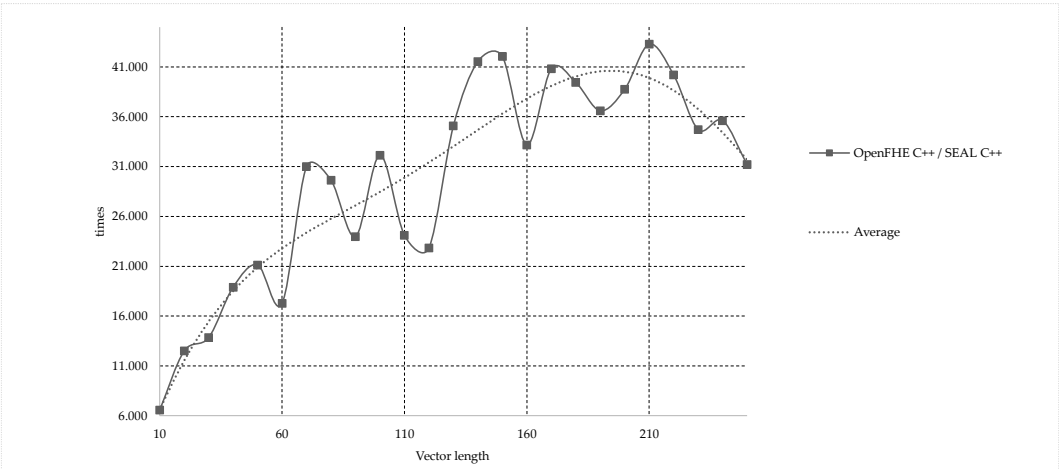

Fig. 2. The result of a study of the error ratio of the algorithm encrypted by the FFT libraries SEAL and OpenFHE in C++ localization.

As we can observe with small values of SEAL, more precisely 6 times, the greatest advantage is in the range from 130 to 210 values in the vector, then the advantage begins to decrease. In general, we can say that SEAL is more accurate than OpenFHE by about ± 30 times (The average values were obtained based on the polynomial $y_1 = 2 \cdot 10^{-12}x^6 - 1 \cdot 10^{-9}x^5 + 2 \cdot 10^{-7}x^4 + 2 \cdot 10^{-5}x^3 - 0.0069x^2 + 0.6861x + 0.3432$ with a likelihood coefficient of $R^2 = 0.8564$).

Thus, we can conclude the following. The considered implementations of the FHE CKKS scheme libraries are quite accurate, but the most accurate is SEAL in the C++ implementation. Based on the result obtained, it is possible to evaluate which implementation is most suitable for which application in terms of accuracy. For example, any implementation can be used to build a confidential NN, but for transmitting an encrypted signal in an area with increased interference, SEAL in the C++ implementation is the most suitable. Next, let's look at the results of performance testing.

5.2 Investigating the performance of fully homomorphic encryption libraries

Similarly to Section 5.1, we will first analyze the general graph (Fig. 3). For clarity, the graph is presented on a logarithmic scale.

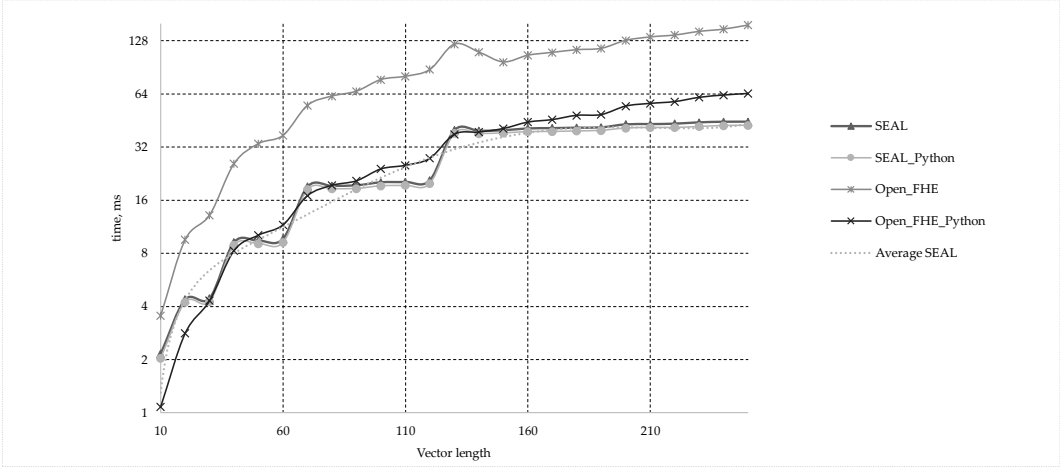

Fig. 3. The results of the performance study of the encrypted FFT algorithm

Analyzing the graph, you can establish the following. OpenFHE has the worst performance in the C++ implementation, followed by OpenFHE in the Python implementation. SEAL has the best performance in both implementations. However, it is worth noting that in some cases, OpenFHE in the Python implementation works faster, such as for a vector of length 10, 20,40,70 and 140 values. Analyzing the curves, it can be established that the OpenFHE curve in the Python implementation is more smoothed than SEAL. The SEAL curve has a stepped appearance up to a vector length of 130, after which it is smoothed. If we denote the stepwise shape of the curve as the effect of noise and approximate the curve using the polynomial $y_2 = 1 \cdot 10^{-12}x^6 + 2 \cdot 10^{-9}x^5 - 9 \cdot 10^{-7}x^4 + 0.0002x^3 - 0.0128x^2 + 0.5851x - 3.4336$ with a likelihood of $R^2 = 0.9575$, it can be noted that the advantage of OpenFHE in the Python implementation remains only on small vectors. Thus, we can say that SEAL has the best performance in both implementations. Now let's conduct a more detailed analysis of the SEAL implementations (Fig. 4).

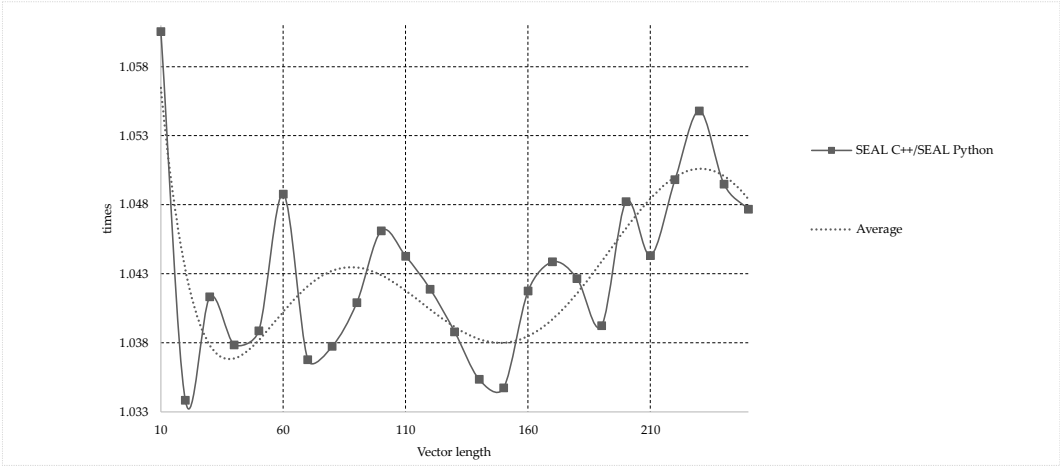

Fig. 4. The result of a study of the performance ratio of the FFT encrypted algorithm by the SEAL library in C++ and Python implementations.

Here we can observe that the Python implementation has a slight performance advantage (The average values were obtained based on the polynomial $y_3 = 2 \cdot 10^{-14}x^6 - 2 \cdot 10^{-11}x^5 + 6 \cdot 10^{-9}x^4 - 1 \cdot 10^{-6}x^3 + 9 \cdot 10^{-5}x^2 - 0.0033x + 1.0814$ with a likelihood coefficient $R^2 = 0.6121$) in the range from 1.033 to 1.060. The average value of the advantage is within ± 1.40. This result can be explained by the fact that in section 5.1 it was found that the Python implementation is less accurate, thus the C++ implementation provides greater accuracy while losing performance. However, as follows from Fig. 4, the loss is quite small.

Thus, we can conclude the following. The SEAL library in the C++ implementation has the best characteristics related to the accuracy of the calculation result. This can be explained by the fact that it has lower performance compared to the Python implementation and more fine-tuning of parameters compared to the OpenFHE library. OpenFHE sets modules based on a given multiplicative depth automatically when, as a SEAL, explicit module assignment is required. Due to this fact, SEAL can support the required number of multiplications at a lower multiplicative depth than OpenFHE, which affects computing performance.

The general results of the testing can be considered as follows. SEAL has the best performance due to more subtle and complex parameter settings. The Python implementation shows the best performance, and the C++ implementation shows the highest accuracy. Thus, SEAL can be used in applications that require the best performance and accuracy with an uncompressed time frame of the application development process. OpenFHE allows you to reduce the development time by reducing the quality and speed of data processing. In addition, both SEAL and OpenFHE in Python implementations are generally suitable for working with artificial intelligence methods, however, the SEAL implementation is not recommended for use in applications requiring high accuracy.

6. Conclusion

In this paper, we developed a system for testing the accuracy and effectiveness of homomorphic encryption libraries for rational numbers with a fixed mantissa. During the study, it was found that the FHE CKKS library falls under the testing criteria. The testing system itself can be built in the likeness of HPC tests, since the scope of FHE can be described as adjacent to HPC, since they are used in CT-based systems. Of the HPC tests, the most suitable for FHE, at the moment, is the calculation of FFT, in addition, it was found that FFT itself has many applications and the development of encrypted FFT is relevant in addition to the development of a testing system.

Thus, a testing system based on encrypted FFT was developed. The most popular FFT libraries that contain the CKKS scheme, namely SEAL and OpenFHE, were selected as test objects. The libraries were considered in two implementations, in C++ and Python. Testing has shown that the SEAL library has the best characteristics. This advantage is achieved due to a more complex and precise selection of parameters for the CKKS scheme, as well as the library structure itself. At the same time, OpenFHE has the advantage of simpler configuration, the user needs to set the minimum parameters, then the library will calculate the remaining parameters automatically.

The paper presents a minimal testing system that does not cover the entire list of characteristics, and also allows testing only two libraries. In the future, it is planned to implement testing of memory casts, a larger set of libraries (for example, HEaaN and HElib), as well as integer FHE schemes such as BFV and BGV. The resulting testing system will allow for detailed research of FHE libraries and circuits, which will enable researchers and developers to select the FHE circuit and library that is most suitable for their research or application.

References

[1] M. Horowitz, ‘1.1 computing’s energy problem (and what we can do about it)’, in *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, IEEE, 2014, pp. 10–14. Accessed: Mar. 28, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6757323/?casa_token=fMNbJ8Nw94gAAAA

A:vmTZxzOOldPqzK0lmLdXWbdnZvqPy6OySZLPaVvb83AZfBNYY0n05qzHfWvjyB5FzWMavxPwDtodgQ

[2] M. A Vouk, 'Cloud computing–issues, research and implementations', *Journal of computing and information technology*, vol. 16, no. 4, pp. 235–246, 2008.

[3] S. Sagiroglu and D. Sinanc, 'Big data: A review', in *2013 international conference on collaboration technologies and systems (CTS)*, IEEE, 2013, pp. 42–47.

[4] R. M. Golden, *Mathematical methods for neural network analysis and design*. MIT Press, 1996. Accessed: Oct. 10, 2023. [Online]. Available: https://books.google.com/books?hl=ru&lr=&id=ru9yRNMfpbsC&oi=fnd&pg=PA1&dq=artificial+neural+network+mathematics&ots=dasu5AVuyu&sig=ZjVg_cHUG8mDRkaV9jidAvXdW7M

[5] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, 'Fast deep neural network training on distributed systems and cloud TPUs', *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 11, pp. 2449–2462, 2019.

[6] S. Teerapittayanon, B. McDanel, and H.-T. Kung, 'Distributed deep neural networks over the cloud, the edge and end devices', in *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*, IEEE, 2017, pp. 328–339. Accessed: Mar. 28, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7979979/?casa_token=BTZNjd-0zbgAAAAA:Ehg9i405WgCFV6X_2NRKgPDTkSP0wuLbn-MJCLTesrLw9yZovV62v9FW1srTSOhr8U-GuhlKMUqK1g

[7] K. Chung and R. C. Park, 'P2P-based open health cloud for medicine management', *Peer-to-Peer Networking and Applications*, vol. 13, no. 2, pp. 610–622, 2020.

[8] A. Meir and B. Rubinsky, 'Distributed network, wireless and cloud computing enabled 3-D ultrasound; a new medical technology paradigm', *PloS one*, vol. 4, no. 11, p. e7974, 2009.

[9] Z. Deng, D. Huang, J. Liu, B. Mi, and Y. Liu, 'An assessment method for traffic state vulnerability based on a cloud model for urban road network traffic systems', *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 7155–7168, 2020.

[10] S. Ghule, R. Chikhale, and K. Parmar, 'Cloud computing in banking services', *International Journal of Scientific and Research Publications*, vol. 4, no. 6, pp. 1–8, 2014.

[11] A. "Cloud Technologies", 'Cloud Cloud.ru — Russian IaaS/PaaS cloud platform and ML services, cloud services from a leading provider', Cloud Cloud.ru — Russian IaaS/PaaS cloud platform and ML services, cloud services from a leading provider. Accessed: Apr. 01, 2024. [Online]. Available: https://cloud.ru/ru

[12] 'Cloud Technology Provider Cloud.ru opened a quantum laboratory', TAdviser.ru. Accessed: Apr. 01, 2024. [Online]. Available: https://www.tadviser.ru/index.php/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F:Cloud.ru_(%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8)_%D1%80%D0%B0%D0%BD%D0%B5%D0%B5_SberCloud

[13] 'Comprehensive promotion in the Yandex ecosystem. Promotion in Yandex.Business, Direct and Zen.' Accessed: Apr. 01, 2024. [Online]. Available: https://www.site-rb.ru/kompleksnoe-prodvizhenie-v-yandekse/

[14] 'VK Ecosystem Services'. Accessed: Apr. 01, 2024. [Online]. Available: https://vk.com/vk_ecosystem_services

[15] 'Products of the MTS ecosystem'. Accessed: Apr. 01, 2024. [Online]. Available: https://personal.mts.ru/

[16] 'Amazon Ecosystem: a brief overview - Research-Methodology'. Accessed: Apr. 01, 2024. [Online]. Available: https://research-methodology.net/amazon-ecosystem-a-brief-overview-2/

[17] B. P. Rimal, E. Choi, and I. Lumb, 'A Taxonomy, Survey, and Issues of Cloud Computing Ecosystems', in *Cloud Computing*, N. Antonopoulos and L. Gillam, Eds., in Computer Communications and Networks. , London: Springer London, 2010, pp. 21–46. doi: 10.1007/978-1-84996-241-4_2.

[18] '420 leaks of databases of Russian companies were recorded during the year', TAdviser.ru. Accessed: Apr. 01, 2024. [Online]. Available: https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A3%D1%82%D0%B5%D1%87%D0%BA%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_%D0%B2_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8

[19] S. Vinoth, H. L. Vemula, B. Haralayya, P. Mamgain, M. F. Hasan, and M. Naved, 'Application of cloud computing in banking and e-commerce and related security threats', *Materials Today: Proceedings*, vol. 51, pp. 2172–2175, 2022.

[20] M. Nabeel, 'The many faces of end-to-end encryption and their security analysis', in *2017 IEEE international conference on edge computing (EDGE)*, IEEE, 2017, pp. 252–259. Accessed: Apr. 01, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8029288/?casa_token=sR8nOdFjE6YAAAAA:sTh2ZzCH5i6m4WqJXBNoRwWdeP_S3_EF_OyBqyAVL1Pbv0pxy0gzz-L66G48XC9V0F6eiNRnBafb

[21] A. Shamir, 'How to share a secret', *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[22] M. Mignotte, 'How to share a secret', in *Workshop on cryptography*, Springer, 1982, pp. 371–375.

[23] C. Asmuth and J. Bloom, 'A modular approach to key safeguarding', *IEEE transactions on information theory*, vol. 29, no. 2, pp. 208–210, 1983.

[24] A. Tchernykh *et al.*, 'AC-RRNS: Anti-collusion secured data sharing scheme for cloud storage', *International Journal of Approximate Reasoning*, vol. 102, pp. 60–73, 2018.

[25] V. Miranda-López *et al.*, 'Experimental analysis of secret sharing schemes for cloud storage based on rns', in *Latin American High Performance Computing Conference*, Springer, 2017, pp. 370–383.

[26] C. Gentry, *A fully homomorphic encryption scheme*. Stanford university, 2009.

[27] A. Al Badawi, Y. Polyakov, K. M. M. Aung, B. Veeravalli, and K. Rohloff, 'Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption Scheme', *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 941–956, Apr. 2021, doi: 10.1109/TETC.2019.2902799.

[28] J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No, 'High-Precision Bootstrapping of RNS-CKKS Homomorphic Encryption Using Optimal Minimax Polynomial Approximation and Inverse Sine Function', in *Advances in Cryptology – EUROCRYPT 2021*, A. Canteaut and F.-X. Standaert, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 618–647. doi: 10.1007/978-3-030-77870-5_22.

[29] C. Gentry, S. Halevi, C. Peikert, and N. P. Smart, 'Ring Switching in BGV-Style Homomorphic Encryption', in *Security and Cryptography for Networks*, vol. 7485, I. Visconti and R. De Prisco, Eds., in Lecture Notes in Computer Science, vol. 7485. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 19–37. doi: 10.1007/978-3-642-32928-9_2.

[30] 'Microsoft SEAL'. Microsoft, Dec. 08, 2022. Accessed: Dec. 10, 2022. [Online]. Available: https://github.com/microsoft/SEAL

[31] 'OpenFHE.org – OpenFHE – Open-Source Fully Homomorphic Encryption Library'. Accessed: Apr. 01, 2024. [Online]. Available: https://www.openfhe.org/

[32] 'Homomorphic Encryption Standardization – An Open Industry / Government / Academic Consortium to Advance Secure Computation'. Accessed: Dec. 10, 2022. [Online]. Available: https://homomorphicencryption.org/

[33] M. Babenko *et al.*, 'A Comparative Study of Secure Outsourced Matrix Multiplication Based on Homomorphic Encryption', *Big Data and Cognitive Computing*, vol. 7, no. 2, p. 84, 2023.

[34] J. J. Dongarra, P. Luszczek, and A. Petitet, 'The LINPACK Benchmark: past, present and future', *Concurrency and Computation*, vol. 15, no. 9, pp. 803–820, Aug. 2003, doi: 10.1002/cpe.728.

[35] P. Luszczek *et al.*, 'Introduction to the HPC challenge benchmark suite', 2005, Accessed: Apr. 02, 2024. [Online]. Available: https://escholarship.org/content/qt6sv079jp/qt6sv079jp.pdf

[36] A. Schwarzenberg-Czerny, 'On matrix factorization and efficient least squares solution.', *Astronomy and Astrophysics Supplement, v. 110, p. 405*, vol. 110, p. 405, 1995.

[37] H. J. Nussbaumer, 'The Fast Fourier Transform', in *Fast Fourier Transform and Convolution Algorithms*, vol. 2, in Springer Series in Information Sciences, vol. 2. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, pp. 80–111. doi: 10.1007/978-3-642-81897-4_4.

[38] J. H. Cheon, A. Kim, M. Kim, and Y. Song, 'Homomorphic encryption for arithmetic of approximate numbers', in *International conference on the theory and application of cryptology and information security*, Springer, 2017, pp. 409–437.

[39] R. L. Rivest, A. Shamir, and L. Adleman, 'A method for obtaining digital signatures and public-key cryptosystems', *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978, doi: 10.1145/359340.359342.

[40] T. ElGamal, 'A public key cryptosystem and a signature scheme based on discrete logarithms', *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[41] A. Kim *et al.*, 'General bootstrapping approach for RLWE-based homomorphic encryption', *IEEE Transactions on Computers*, 2023, Accessed: Apr. 02, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10261340/

[42] C. Gentry, 'Fully homomorphic encryption using ideal lattices', in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, Bethesda MD USA: ACM, May 2009, pp. 169–178. doi: 10.1145/1536414.1536440.

[43] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, 'Fully homomorphic encryption over the integers', in *Annual international conference on the theory and applications of cryptographic techniques*, Springer, 2010, pp. 24–43.

[44] 'HEaaN'. Accessed: Apr. 02, 2024. [Online]. Available: https://heaan.it/

[45] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, 'TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption'. arXiv, Apr. 28, 2021. Accessed: Sep. 27, 2023. [Online]. Available: http://arxiv.org/abs/2104.03152

[46] W. Jenkins and M. Desai, 'The discrete frequency Fourier transform', *IEEE transactions on circuits and systems*, vol. 33, no. 7, pp. 732–734, 1986.

[47] D. H. Mugler and R. A. Scott, 'Fast fourier transform method for partial differential equations, case study: The 2-D diffusion equation', *Computers & Mathematics with Applications*, vol. 16, no. 3, pp. 221–228, 1988.

[48] S. Li *et al.*, 'Falcon: A fourier transform based approach for fast and secure convolutional neural network predictions', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8705–8714. Accessed: Apr. 04, 2024. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2020/html/Li_FALCON_A_Fourier_Transform_Based_Approach_for_Fast_and_Secure_CVPR_2020_paper.html

[49] T. Highlander and A. Rodriguez, 'Very Efficient Training of Convolutional Neural Networks using Fast Fourier Transform and Overlap-and-Add'. arXiv, Jan. 25, 2016. Accessed: Apr. 04, 2024. [Online]. Available: http://arxiv.org/abs/1601.06815

[50] H. Pratt, B. Williams, F. Coenen, and Y. Zheng, 'FCNN: Fourier Convolutional Neural Networks', in *Machine Learning and Knowledge Discovery in Databases*, vol. 10534, M. Ceci, J. Hollmén, L. Todorovski, C. Vens, and S. Džeroski, Eds., in Lecture Notes in Computer Science, vol. 10534. , Cham: Springer International Publishing, 2017, pp. 786–798. doi: 10.1007/978-3-319-71249-9_47.

[51] H. M. El-Bakry and Q. Zhao, 'Fast object/face detection using neural networks and fast Fourier transform', *International Journal of Computer and Information Engineering*, vol. 1, no. 11, pp. 3748–3753, 2007.

[52] Y. He, H. Chen, D. Liu, and L. Zhang, 'A framework of structural damage detection for civil structures using fast fourier transform and deep convolutional neural networks', *Applied Sciences*, vol. 11, no. 19, p. 9345, 2021.

[53] J. Qi, J. Du, S. M. Siniscalchi, X. Ma, and C.-H. Lee, 'On mean absolute error for deep neural network based vector-to-vector regression', *IEEE Signal Processing Letters*, vol. 27, pp. 1485–1489, 2020.

[54] J. Qi, J. Du, S. M. Siniscalchi, X. Ma, and C.-H. Lee, 'Analyzing upper bounds on mean absolute errors for deep neural network-based vector-to-vector regression', *IEEE Transactions on Signal Processing*, vol. 68, pp. 3411–3422, 2020.

[55] A. Prošek, F. D'Auria, and B. Mavko, 'Review of quantitative accuracy assessments with fast Fourier transform based method (FFTBM)', *Nuclear Engineering and Design*, vol. 217, no. 1–2, pp. 179–206, 2002.

[56] 'NTL: A Library for doing Number Theory'. Accessed: May 23, 2023. [Online]. Available: https://libntl.org/

[57] 'The GNU MP Bignum Library'. Accessed: Apr. 06, 2024. [Online]. Available: https://gmplib.org/

Information about authors

Егор Михайлович ШИРЯЕВ – аспирант и ассистент кафедры вычислительной математики и кибернетики Северо-Кавказского Федерального Университета. Его научные интересы включают искусственные нейронные сети, криптография, гомоморфное шифрования, схемы разделения секрета, модулярная арифметика, облачные вычисления.

Egor Mikhailovich SHIRYAEV – graduate student and assistant at the Department of Computational Mathematics and Cybernetics of the North Caucasus Federal University. His research interests include artificial neural networks, cryptography, homomorphic encryption, secret sharing schemes, modular arithmetic, cloud computing.

Михаил Григорьевич БАБЕНКО – доктор физико-математических наук, доцент, заведующий кафедрой вычислительной математики и кибернетики Северо-Кавказского Федерального Университета с 2020 года. Его научные интересы включают искусственные нейронные сети, криптография, гомоморфное шифрования, схемы разделения секрета, модулярная арифметика, облачные вычисления.

Mikhail Grigorievich BABENKO – Doctor of Physical and Mathematical Sciences, Associate Professor, Head of the Department of Computational Mathematics and Cybernetics of the North Caucasus Federal University since 2020. His research interests include artificial neural networks, cryptography, homomorphic encryption, secret sharing schemes, modular arithmetic, cloud computing.

Екатерина Сергеевна БЕЗУГЛОВА - стажер-исследователь отдела теоретико-числовых систем регионального научно-образовательного математического центра «Северо-Кавказский центр математических исследований». Научные интересы: искусственные нейронные сети, криптография, гомоморфное шифрования, схемы разделения секрета, модулярная арифметика, облачные вычисления

Ekaterina Sergeevna BEZUGLOVA is a trainee researcher in the department of number-theoretic systems of the regional scientific and educational mathematical center "North Caucasus Center for Mathematical Research". Research interests: artificial neural networks, cryptography, homomorphic encryption, secret sharing schemes, modular arithmetic, cloud computing

Мария Анатольевна ЛАПИНА - опытный специалист в области научных исследований и международных отношений, в настоящее время занимающая должность заместителя директора по международным связям и доцента кафедры информационной безопасности автоматизированных систем. Она защитила кандидатскую диссертацию по математике в Ставропольском государственном университете с отличием в 2005 году и получила ученую степень доцента в 2009 году. Мария также имеет степень магистра в области информационной безопасности в Северо-Кавказском федеральном университете с отличием, полученную в 2015 году. Мария участвовала в международных стажировках в Римском университете Сапиенца (Италия), Университет Любляны (Словения), Технический университет Дрездена (Германия) и 2-я инновационная лаборатория высшего образования ASEF (ASEFInnoLab) при Университете Фудань в Шанхае, Китай.

Maria Anatolyevna LAPINA is an experienced specialist in the field of scientific research and international relations, currently holding the position of Deputy Director for International Relations and Associate Professor of the Department of Information Security of Automated Systems. She defended her PhD thesis in mathematics at Stavropol State University with honors in 2005 and received an associate professor degree in 2009. Maria also holds a Master's degree in Information Security from the North Caucasus Federal University with honors, obtained in 2015. Maria has participated in international internships at Sapienza University of Rome (Italy), the University of Ljubljana (Slovenia), the Technical University of Dresden (Germany) and the 2nd ASEF Innovation Laboratory of Higher Education (ASEFInnoLab) at Fudan University in Shanghai, China.