Discrete & Continuous Models
& Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2026, 34 (1) 12–23
http://journals.rudn.ru/miph

Research article

UDC 519.7
PACS 07.05.Tp
DOI: 10.22363/2658-4670-2026-34-1-12-23 EDN: URRPMT

Usage of polynomial representation of numbers for approximate homomorphic encryption

Andrey E. Krouk

Saint-Petersburg State University of Aerospace Instrumentation, 67 B. Morskaia St, 190000, Saint-Petersburg, Russian Federation

(received: December 16, 2025; revised: January 25, 2026; accepted: January 30, 2026)

Abstract. *Introduction* In the modern world of computers and networks the idea of expanding of personal computer resources with the help of cloud storages and computation looks more and more lucrative. However, usage of these resources may endanger data being processed. In last twenty years several algorithms of homomorphic encryption were developed allowing solving of this problem among other applications. However such algorithms are usually constructed as public key systems for long term storage and processing of data. In this article two algorithms of homomorphic encryption optimized for single data processing are proposed. *Purpose* The target of research is development of data coding system which allows safe data processing in public clouds. *Results* Two homomorphic coding systems had been developed, first is based on representation of numbers in the form of polynomials, second based on further representation of polynomials in the form of sets of values. Developed systems allow approximate calculations of coded data without decryption allowing processing of real numbers. System has high level of protection and provides high precision of calculations, comparable with standard personal computer calculation precision. Structure of coded data allows parallel computing. Proposed system allows safe data processing in public networks. Question of finding of optimal parameters for the system stands open both for high precision calculation of limited sets of operations and repeatedly good precision for big sets of operations.

Key words and phrases: homomorphic encryption, cloud calculations

For citation: Krouk, A. E. Usage of polynomial representation of numbers for approximate homomorphic encryption. *Discrete and Continuous Models and Applied Computational Science* **34** (1), 12–23. doi: 10.22363/2658-4670-2026-34-1-12-23. edn: URRPMT (2026).

1. Introduction

People have long been interested in the idea of hidden computing. The need to decrypt data for subsequent processing made it vulnerable. Now, with the introduction of public networks and the ability to process data on external, often public, resources, the task of performing hidden computations on encrypted data has become extremely relevant. To address this challenge, in the mid-20th century, the idea of homomorphic encryption was proposed, a method of transforming (encrypting) numerical data that allows operations to be performed on encrypted data without decryption [1, 2].

First, so-called partially homomorphic encryption systems were developed. These systems typically allowed for one type of arithmetic operation without decryption. For example, the RSA and ElGamal systems [3] enable multiplication without decryption, while the Paillier and Benaloh systems enable addition.

For a long time, attempts to create a fully homomorphic encryption system or combine several partially homomorphic encryption systems into one have been unsuccessful. The turning point was the introduction of a system developed by Craig Gentry in 2009 [4]. The system is based on ideal lattices. Numerical noise is added to the data, making it impossible to decrypt without knowing the key. The system allows for any arithmetic operations without decrypting the data. As the number of operations increases, the noise level increases as well, but the system includes a procedure to reduce the noise level. This allows for an arbitrary number of operations without decrypting the data, but at the cost of increased computational complexity.

Development of homomorphic encryption systems has significantly expanded the possibilities for conducting calculations in cloud and collaborative systems [1, 5].

In 2010, a revised version of Gentry's scheme based on integers was introduced [6]. Despite using a different mathematical foundation, this scheme employs Gentry's method of encryption through the addition of noise and a noise reduction procedure. However, many fully homomorphic encryption systems are based on lattices [7].

Along with systems that perform precise arithmetic operations, usually on integers, systems that perform approximate arithmetic operations have begun to appear, which are applicable to performing operations on real numbers. In 2016, the CKKS system [8] was released, the first system that performs approximate calculations and is designed to work with real and complex numbers. In this system, data is first represented as circular polynomials, and then the problem of learning with errors in the ring is solved to create a homomorphic encryption system that has a public key for encrypting data. However, this system is designed to perform a finite number of operations, as increasing the number of operations increases the complexity of the calculations and negatively affects the system's security. The re-encryption procedure proposed a little later [9] only partially solves the problem, as its use leads to an increase in the complexity of calculations, which does not allow it to be used too often [10]. A detailed comparison of the CKKS system [8, 11] with systems that perform exact calculations, such as the BFV system [12–14], is provided in [15]. It should also be noted that although the presence of a public key opens up additional possibilities for using the system, it is not necessary for solving certain problems and negatively affects the system's security [16, 17].

This article presents several variants of approximate fully homomorphic encryption systems that use a fundamentally new approach to creating systems: representing numbers as polynomials and, additionally, replacing the represented polynomials with a set of their values at points. Although this approach does not allow for the use of a public key, it provides near-absolute security by using session-specific keys that are not transmitted outside of the trusted PC.

This method can be effectively used to process large amounts of data on external (unsecure) computing resources (so-called public cloud computing).

2. Usage of polynomials for construction of partially homomorphic encryption system

Simple homomorphic encryption system can be built using residue number system based on Chinese Reminder Theorem (CRT) [18, 19].

This system can be further developed by replacement of residual number system with a polynomial representation. Let's discuss this approach in more detail.

Let's suppose that a sequence of arithmetic operations has to be performed on a set of numbers. We will encode the data for calculations in two stages:

1. We assign to each number a polynomial such that the value of this polynomial at some point x_s is equal to the number. The value of x_s is the same for all pairs of number-polynomial and is the secret key of the system.
2. We will choose a set of points. Values of polynomials in these points we will use to define these polynomials. (It is preferable to choose points where the polynomial values can be calculated relatively easily.) We will assign a set of values in the selected points to each polynomial. This set of values will be the result of the encoding process.

Let's combine into sets the values of all selected polynomials at each of the points (each set consists of the values of all polynomials in one of the points). Now, to perform any sequence of arithmetic operations on the original numbers, it is enough to perform the same sequence of operations within each set. Indeed, since the result of adding (multiplying) two polynomials will be a polynomial whose values at any point will be the addition (multiplication) of the values of the polynomial terms (multipliers) at these points, the resulting values match the values of the polynomial that would result from performing the given sequence of arithmetic operations on the polynomials corresponding to the original data. Let us call this polynomial the result polynomial.

The degree of this result polynomial (m) can be easily estimated by the degrees of the original polynomials. Indeed, when adding polynomials, the degree of the result can be estimated by the highest degree of the terms, and when multiplying, it can be estimated by the sum of the degrees of multipliers.

Thus, when we receive $m + 1$ calculated values back, we can perform polynomial interpolation and obtain the result polynomial.

By calculating the value of the result polynomial in the secret point x_s, we can find the final result of the calculation.

Let's evaluate the advantages and disadvantages of the proposed method. This method retains both the many advantages of the original method (simplicity of calculations, ease of parallelization of calculations) and the main disadvantage (the absence of a division operation). At the same time, the presence of a two-step encoding operation and a secret key allows for high data security without the use of false requests or other additional techniques for data protection.

3. Approximate calculations

Integer homomorphic encryption systems do not allow calculations with non-integer numbers. When working with real numbers, in particular when performing division, it is necessary to perform approximate calculations.

Approximate calculations are widely used, especially in technical and physical problems. This is because many data can only be measured with finite precision, and many elements can also be produced with finite precision. Increasing the precision of calculations leads to an excessive increase in the complexity [8, 20, 21]. In the same time there is no reason to increase the precision of calculations beyond the precision of measurements or production, as the precision of the final result is determined by the precision of the most inaccurate value.

When a computer works with real variables, it also limits the precision of the calculations, and these calculations are strictly speaking approximate.

4. Representation of numbers as polynomials

Let's improve the encoding procedure so that we can implement an approximate division procedure. This time, we will encode the data in stages and evaluate the resulting system at each stage.

Let's replace operations on numbers with operations on polynomials. To do this, we assign to each number a polynomial so that the value of this polynomial at some point x_s is equal to the number. The value of x_s must be the same for all pairs of number–polynomial and is the secret key of the system (we will call it the "secret point").

In this representation, it is easy to implement addition and multiplication operations, but there are problems with implementing division. Indeed, in most cases, it is impossible to divide polynomials completely, meaning that there will be a remainder when dividing.

$$\frac{f(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)}.$$

However, this problem can be resolved by considering division as an approximate operation and choosing x to be sufficiently big. Indeed, since the degree of the remainder $r(x)$ is less than the degree of the divisor $g(x)$, the following formula holds

$$\frac{r(x)}{g(x)} \underset{x\to\infty}{\longrightarrow} 0.$$

However, this imposes restrictions on the encoding procedure. In order to use this formula, it is necessary that x be sufficiently big, meaning that the value (weight) of the leading term of the polynomial at the secret point would be significantly greater than the sum of the values of the other terms.

In addition, it is necessary to complicate the system somewhat in order to be able to divide polynomials in the case when the degree of the dividend is less than the degree of the divisor. We will use not polynomials to represent numbers, but constructions $f(x)/x^k$, i.e. polynomials divided by the degree of x. In this representation, when performing the division operation, we get the opportunity to multiply and divide the dividend by x^k (division will be carried out after calculating the result of the main operation, by simply adding the corresponding degree of x to the denominator) so that its degree exceeds the degree of the divisor. Choosing bigger values of k allows to improve the operation precision. In addition, this method can be used to improve the precision of division even when the degree of the dividend is greater than the degree of the divisor.

5. Homomorphic encryption system based on representation of numbers as polynomials

Let's build a homomorphic encryption system that satisfies the principles described in the previous paragraph. We will calculate multiplication, addition, and subtraction in the traditional exact way, and division in an approximate way. To do this, we will assign to each number in our system a polynomial divided by the power of x ($f(x)/x^k$) such that the values of the polynomials at a secret point (the secret point is the same for all polynomials) are equal to the encoded numbers, and replace operations on numbers with operations on the coefficients of the polynomials. It is impossible to restore the values of the numbers in the system without knowing the secret point. The absence of the need to transmit this secret value and the ability to choose a new secret value for each session ensure high security of the system.

Operations on polynomial coefficients are implemented as follows.

– Division. An approximate operation. First, the degree of the dividend is increased by multiplying it by x^k, and then division with remainder is performed. The incomplete quotient is used as the result of the division, and the remainder is neglected. Increasing the degree of x^k allows for greater precision in division, at the cost of increase in the complexity of the calculations and potentially the complexity of decryption. Finally, x^k is included in the weight factor (denominator) of the element.

  For example, when dividing $x + 1$ by x^3, you can multiply and divide $x + 1$ by x^2

$$\frac{x^3 + x^2}{x^3}\frac{1}{x^2} = \left(1 + \frac{1}{x}\right)\left(\frac{1}{x^2}\right) = \frac{1}{x^2}.$$

  If x^5 is used instead of x^2, the result will be

$$\frac{x^6 + x^5}{x^3}\frac{1}{x^5} = \left(x^3 + x^2\right)\frac{1}{x^5},$$

  this provides greater accuracy, as the polynomials are completely divided.
– Multiplication. Polynomials are multiplied classically, and the weight factors (denominators) x^k are also multiplied. For example:

$$(x + 1)x^{-1}(x + 2)x^{-2} = (x^2 + 3x + 2)x^{-3}.$$

– Addition and subtraction. To add and subtract elements, they should first be brought to a common weight factor (denominator) (will be x with the highest absolute value of the degree), and then the polynomials are added or subtracted using the classical method. For example:

$$(x + 1)x^{-1} + (x + 2)x^{-2} = (x^2 + x)x^{-2} + (x + 2)x^{-2} = (x^2 + 2x + 2)x^{-2}.$$

It is necessary to choose an encoding algorithm to implement the system within the restrictions described in the previous paragraph. That is, it is necessary to choose the value of the secret point to be sufficiently big, and the coefficients of the polynomials to be chosen so that the leading term of the polynomials is sufficiently heavy (that is, so that the leading term contains most of the value of the number; with 70% of the value, the error is approximately 10 times greater than with 90%).

The following algorithm was used during the analysis of the system.

– The degree of the polynomial and the weight of the leading term as a percentage are selected. These are the parameters of the algorithm.
– The coefficient at the highest power is calculated so that the leading term has a value equal to the specified part of the encoded value.
– The next coefficient is chosen so that the value of the second term is equal to the specified part of the difference between the required value and the value of the leading term. And so on.
– The constant term is selected as the difference between the required value and the values of all other terms, ensuring that the required value is accurately matched.

Let's consider the algorithm's operation using the following example. Let's encrypt the number 100 using a polynomial of the second degree at the point 10 with a weight of 80% for the leading term.

– With the selected parameters, the value of the leading term should be $100 \times 0.8 = 80$. Since the degree of the polynomial is 2, the coefficient for the x^2 should be $80/10^2 = 0.8$.
– The value of the second term should be $(100 - 80) \times 0.8 = 16$. Since the degree of the second term is 1, the coefficient for the x should be $16/10 = 1.6$.
– Finally, the value of the constant term is obtained as $100 - 80 - 16 = 4$.

Thus, the number 100 is associated with a polynomial $0.8x^2 + 1.6x + 4$.

When encoding using this algorithm, the weight factors x^{-k} of the polynomials obtained during the encoding phase are always assumed to be equal to 1 ($k = 0$). However, during the division operation, the weight factor may change and participate in further calculations.

Additional measures can be taken to improve security of the system:

- Variable polynomial degrees: the polynomials describing the different points must have different degrees, but at least 2.
- Variable weight of the leading term: the weight of the leading term for each polynomial (or even for each term in each polynomial) is chosen as a random number within a specified range that ensures acceptable precision, such as between 70% and 90%.

To decrypt, it is enough to calculate the value of the polynomial at the secret point (taking into account the weight coefficient x^{-k}).

Let's evaluate the advantages and disadvantages of the proposed system. The main advantages of this system are its completeness (it supports all four arithmetic operations) and its high security due to the use of one-time, non-transmittable secret keys. The main disadvantage of this system is its computational complexity, as it requires operations on sets of numbers instead of individual numbers, and it does not support easy parallelization of calculations, which is very convenient for external networks.

6. Usage of interpolation for operations with polynomials

To increase the security of the system, as well as to facilitate parallel calculations, each polynomial can be represented as a set of values at points. After that, operations are performed not on the coefficients of the polynomial, but on these values. As the operations on the values at different points are independent of each other, they can be performed in parallel or, for example, as separate tasks for cloud computing (including in the public cloud, due to high security of the system).

There is no point in describing these arithmetic operations in detail, as they are literally operations on values (numbers) — addition, subtraction, multiplication, or division of values.

In principle, to speed up calculations, it is possible to use the values of polynomials at small points, but research has shown that this leads to a noticeable loss of division operation precision. However, this allows for the construction of an effective partially homomorphic encription system for three operations (addition, subtraction, and multiplication), with a high computational speed that can compete with modular arithmetic (such system was described above). At the same time, as long as the values used in the calculations (or a part of them) are comparable to the secret value, the precision of the operations is sufficient, allowing for the freedom to choose specific values or patterns for their generation. For example, to simplify calculations, values of the form 2^k can be used as long as at least one of them is greater than the secret value.

Encoding process follows these steps:

- Encoding of values with polynomials (as described in the previous paragraph).
- Selecting of a set of points and calculating the values of the polynomials at those points.

The values of the points in the selected set are also a secret, and are also not transmitted anywhere, which contributes to the high security of the system. The number of points in the set is selected based on the estimation of the degree of the polynomial that should be obtained as a result of the calculations.

Decryption process follows these steps:

- Estimating of the x^{-k} weight coefficient on your own computing base (this is easy to do because all operations process it uniquely) and the degree of the resulting polynomial (to determine the number of points required for decryption)
- Calculating of the values of the polynomial at all points, taking into consideration the weight coefficient x^{-k} (that is, multiply the obtained values by x^k).
- Restoring of the polynomial using the interpolation method.
- Calculating the value at the secret point (again, taking into consideration the x^{-k} weight coefficient).

To further improve security and reduce waiting times, an excessive set of points can be used (decoding can be performed as soon as sufficient number of values is obtained). This reduces the impact of various delays that occur both on data-processing servers and during packet transition through the network (so-called transport coding) [22].

It should be noted that in order to improve the precision of the system, when evaluating the weight coefficient, it is possible to include a multiplication-division operation by a weight coefficient of a relatively high degree in division operation (as in description of division in the previous section), which allows for a more precise division operation. Let's refer to the degree of x used for the multiplication-division operation as the correction and use it as a parameter for evaluating the precision of the operations. Using higher values of the correction can result in an increase in the degree of the final polynomial, which can lead to increase in complexity of the decryption process and number of points required for decryption.

When using the interpolation representation of polynomials, the coefficients of the polynomials and their degrees are hidden. This makes it unnecessary to use the additional security measures mentioned in the previous paragraph, as they can negatively impact precision. Moreover, using the interpolation representation allows for the selection of a specific structure of the polynomials to facilitate subsequent calculations, such as encoding numbers with monomials of a given degree (i.e., using 100% weight in the leading term) without compromising security.

Let's evaluate the advantages and disadvantages of the proposed system. Usage of interpolation for polynomial operations further enhances the system's security while reducing overall computational complexity by representing data as independent sets that can be processed in parallel.

7. Results of experiments

The system was tested both with and without the interpolation representation of polynomials.

The system's functionality was tested without the use of interpolation representation. Proposed methods for increasing the system's security were also tested. As a result of these tests, the division precision sufficient for engineering calculations was proved. As this method is inferior to the interpolation representation, the purpose of these tests was to verify the feasibility of the idea, and multiple tests were not conducted to verify the precision.

Multiple tests were conducted for the interpolation representation. The interpolation point sets were selected according to the principle $(i + 1) \bullet 25$, where $i = 0, 1, ...$, as they allowed for more precise calculations, and 2^i, as using this set allows for faster calculations. The secret point was not part of either set. The results were obtained for different values of the correction parameter (the correction parameter was introduced in the section describing the use of interpolation for polynomial operations) and different numbers of interpolation points.

Table 1

Single division error with leading monomial weight of 95%

Correction	Number of points	Average error rate	Maximum error
Selecting points based on the principle $(i+1) \bullet 25$			
1	2	$2.6 \cdot 10^{-16}$	$1.5 \cdot 10^{-15}$
1	3	$1.5 \cdot 10^{-15}$	$7.1 \cdot 10^{-15}$
2	5	$1.5 \cdot 10^{-14}$	$7.8 \cdot 10^{-14}$
Selecting points based on the principle 2^i			
4	5	$3.1 \cdot 10^{-16}$	$1.6 \cdot 10^{-15}$

Table 2

Multiple operation (8 divisions, 7 multiplications, 2 additions) error with leading monomial weight of 95%

Correction	Number of points	Average error rate	Maximum error
Selecting points based on the principle $(i+1) \bullet 25$			
1	2	$5.8 \cdot 10^{-16}$	$3.2 \cdot 10^{-15}$
1	3	$2.1 \cdot 10^{-15}$	$1.4 \cdot 10^{-14}$
2	5	$1.6 \cdot 10^{-14}$	$7.6 \cdot 10^{-14}$
Selecting points based on the principle 2^i			
4	5	$4.9 \cdot 10^{-16}$	$2.4 \cdot 10^{-15}$

During the tests, two important results were achieved. First, by using a weight of 95% for the leading term in the encoding algorithm, the division precision was achieved at the level of the PC's precision in performing precise (addition and multiplication) arithmetic operations (due to the PC's inaccuracy in handling real-number variables) for real numbers (with a maximum deviation of 10^{-14} and an average deviation of 10^{-15}).

The results of the most interesting test runs are presented in Tables 1 and 2.

Secondly, a research was conducted on the growth of the error as a function of the number of operations performed. In this research, significantly less convenient encoding parameters were used, with the weight of the leading term for each point randomly selected between 80% and 90%. A large number of operations was achieved through consecutive divisions and multiplications (with one more division to ensure that the degree of the result polynomial was 0, before taking correction in consideration) of various randomly selected five-digit numbers. As a result, it was possible to find parameter values (albeit not optimal) under which the increase in error with increase in operations number is almost non-existent.

Examples of errors for such parameter values (with different numbers of operations) are given in Tables 3 and 4.

Table 3

Error for different number of consecutive operations

Number of operations	Average error rate	Maximum error
3	$2.4 \cdot 10^{-10}$	$1.3 \cdot 10^{-9}$
5	$2.4 \cdot 10^{-10}$	$1.4 \cdot 10^{-9}$
17	$2.4 \cdot 10^{-10}$	$1.3 \cdot 10^{-9}$

Encoding is used with a leading term weight of 80%–90%, and the values of the points (where the calculations are performed) are selected using the formula $(i + 1) \bullet 25$.

11 points are used for correction of 8.

Table 4

Error for different number of consecutive operations

Number of operations	Average error rate	Maximum error
3	$9.1 \cdot 10^{-8}$	$4.4 \cdot 10^{-7}$
5	$9.1 \cdot 10^{-8}$	$5.1 \cdot 10^{-7}$
17	$9.1 \cdot 10^{-8}$	$5.33 \cdot 10^{-7}$

Encoding is used with a leading term weight of 80%–90%, and the values of the points (where the calculations are performed) are selected using the formula 2^i.

14 points are used for correction of 12.

8. System security evaluation

The system decryption is performed in two stages. Let's try to assess the security of these stages.

1. At the first stage, a polynomial is determined based on the values. The main problem for unauthorized decryption at this stage is that the attacker does not have information about the degree of the polynomial (this information is calculated on the base computer and is not transmitted anywhere) and about the points where the polynomial values are calculated (these values are also not transmitted anywhere). The attacker may attempt to obtain some information by using points in the set that are close to the secret point, but he does not know which points in the set to use for evaluation, and the accuracy of the evaluation remains questionable.
2. In the second step, the value at the secret point is determined using the polynomials. To determine the value at the secret point, secret key is needed. This key is unique for each calculation and is not shared with anyone, so the attacker does not know its value. If there are several values in the result, the attacker can try to exploit the higher weight of the leading term and find the relationship between them by dividing one polynomial by another. The information obtained in this way depends on the specific polynomials used for the encoding. For example, if this attack is performed on a system used during research, the result will be inaccurate because the weight of the leading term is chosen with considerable random error during the encoding process.

9. Conclusions

- The paper proposes a method of approximate homomorphic encription that provides sufficient precision for engineering calculations. The conducted research allows to hope for the existence of encoding algorithms that ensure the precision of the system's operation at the precision level of PC operations with real numbers.
- The high security of the method and the ease of dividing calculations into parallel processes make it suitable for use in public cloud networks.
- The ability to increase the number of points and the division operation parameters allows to adjust the precision and speed of calculations.
- The conducted research shows that there are system parameters under which the errors of multiple operations do not accumulate significantly.

Author Contributions: Conceptualization, Krouk A. E.; methodology, Krouk A. E.; writing—review and editing Krouk A. E.; supervision, Krouk A. E.; project administration, Krouk A. E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data sharing is not applicable.

Conflicts of Interest: The author declares no conflict of interest.

Declaration on Generative AI: The author has not employed any generative AI tools.

References

1. Rivest, R. L., Adleman, L. & Dertouzos, M. L. *On Data Banks and Privacy Homomorphisms* in *Foundations of Secure Computation* (1978).
2. Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., Fitzek, F. H. P. & Aaraj, N. Survey on Fully Homomorphic Encryption, Theory, and Applications. *Proceedings of the IEEE* **110,** 1572–1609. doi:10.1109/JPROC.2022.3205665 (2022).
3. Van Tilborg, H. C. A. *Fundamentals of Cryptology* [in Russian] (Mir, Moscow, 2006).
4. Gentry, C. *A Fully Homomorphic Encryption Scheme* PhD thesis (Stanford University, 2009).
5. Brakerski, Z., Gentry, C. & Vaikuntanathan, V. *(Leveled) fully homomorphic encryption without bootstrapping* in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)* (Association for Computing Machinery, Cambridge, Massachusetts, 2012), 309–325. doi: 10.1145/2090236.2090262.
6. van Dijk, M., Gentry, C., Halevi, S. & Vaikuntanathan, V. Fully Homomorphic Encryption over the Integers. Cryptology ePrint Archive, Report 2009/616 (2010).
7. Albrecht, M. *et al. Homomorphic Encryption Standard* in *Protecting Privacy through Homomorphic Encryption* (eds Lauter, K., Dai, W. & Laine, K.) (Springer International Publishing, Cham, 2021). doi:10.1007/978-3-030-77287-1_2.
8. Cheon, J. H., Kim, A., Kim, M. & Song, Y. *Homomorphic Encryption for Arithmetic of Approximate Numbers* in *Advances in Cryptology – ASIACRYPT 2017* (eds Takagi, T. & Peyrin, T.) (Springer International Publishing, Cham, 2017), 409–437. doi:10.1007/978-3-319-70694-8_15.
9. Brakerski, Z. & Vaikuntanathan, V. *Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages* in *Advances in Cryptology – CRYPTO 2011* (ed Rogaway, P.) **6841** (Springer Berlin Heidelberg, Berlin, Heidelberg, 2011), 505–524. doi:10.1007/978-3-642-22792-9_29.

10. Cheon, J. H., Han, K., Kim, A., Kim, M. & Song, Y. *Bootstrapping for Approximate Homomorphic Encryption* in *Advances in Cryptology – EUROCRYPT 2018* (eds Nielsen, J. B. & Rijmen, V.) (Springer International Publishing, Cham, 2018), 360–384. doi:10.1007/978-3-319-78381-9_14.
11. Chen, H., Laine, K. & Player, R. *Simple Encrypted Arithmetic Library – SEAL (v2.1)* in *Financial Cryptography and Data Security* (eds Brenner, M., Rohloff, K., Bonneau, J., Miller, A., Ryan, P. Y., Teague, V., Bracciali, A., Sala, M., Pintore, F. & Jakobsson, M.) **10323** (Springer International Publishing, Cham, 2017), 3–18. doi:10.1007/978-3-319-70278-0_1.
12. Lepoint, T. & Naehrig, M. *A Comparison of the Homomorphic Encryption Schemes FV and YASHE* in *Progress in Cryptology – AFRICACRYPT 2014* **8469** (2014), 318–335. doi:10.1007/978-3-319-06734-6_20.
13. Bajard, J.-C., Eynard, J., Hasan, M. A. & Zucca, V. *A Full RNS Variant of FV Like Somewhat Homomorphic Encryption Schemes* in *Cryptographic Hardware and Embedded Systems – CHES 2016* (eds Avanzi, R. & Heys, H.) **10532** (2016), 423–442. doi:10.1007/978-3-319-69453-5_23.
14. Halevi, S., Polyakov, Y. & Shoup, V. *An Improved RNS Variant of the BFV Homomorphic Encryption Scheme* in *Topics in Cryptology – CT-RSA 2019* (ed Matsui, M.) **11405** (2019), 83–105. doi:10.1007/978-3-030-12612-4_5.
15. Babenko, M. G., Golimblevskaia, E. I. & Shiriaev, E. M. Comparative Analysis of Homomorphic Encryption Algorithms Based on Learning with Errors. *Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS)* **32,** 37–51. doi:10.15514/ISPRAS-2020-32(2)-4 (2020).
16. Gentry, C. *Fully Homomorphic Encryption Using Ideal Lattices* in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)* (2009), 169–178. doi:10.1145/1536414.1536440.
17. Pulido-Gaytan, B., Tchernykh, A., Cortés-Mendoza, J. M., Babenko, M., Radchenko, G., Avetisyan, A. & Drozdov, A. Y. Privacy-preserving Neural Networks with Homomorphic Encryption: Challenges and Opportunities. *Peer-to-Peer Networking and Applications* **14,** 1666–1691. doi:10.1007/s12083-021-01076-8 (2021).
18. Akritas, A. *Elements of Computer Algebra with Applications* 425 pp. (John Wiley & Sons, Inc., United States, 1989).
19. Krouk, A. E. & Fedorenko, S. V. *Construction of the solution of the Chinese Remainder Theorem for polynomials using the method of undetermined coefficients* in *2019 XVI International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY)* (Moscow, Russia, 2019), 115–116. doi:10.1109/REDUNDANCY48165.2019.9003344.
20. Halevi, S. & Shoup, V. *Bootstrapping for HElib* in *Advances in Cryptology – EUROCRYPT 2015* (eds Oswald, E. & Fischlin, M.) **9056** (2015), 641–670. doi:10.1007/978-3-662-46800-5_25.
21. Li, B. & Micciancio, D. *On the Security of Homomorphic Encryption on Approximate Numbers* in *Advances in Cryptology – EUROCRYPT 2021* **12696** (2021), 648–677. doi:10.1007/978-3-030-77870-5_23.
22. Kabatiansky, G., Krouk, E. & Semenov, S. *Error Correcting Coding and Security for Data Networks. Analysis of the Super Channel Concept* doi:10.1002/0470867574 (John Wiley & Sons, Ltd., 2005).

Information about the authors

Krouk, Andrey E.—Candidate of Technical Sciences, Associate Professor (e-mail: svinenka@mail.ru, ORCID: 0009-0008-1162-2020)

УДК 519.7
PACS 07.05.Tp
DOI: 10.22363/2658-4670-2026-34-1-12-23 EDN: URRPMT

Использование представления чисел в виде многочленов для реализации скрытых приближённых вычислений

А. Е. Крук

Санкт-Петербургский государственный университет аэрокосмического приборостроения,
ул. Б. Морская, д. 67, Санкт-Петербург, 190000, Российская Федерация

Аннотация. *Введение* В современном мире компьютеров и сетей всё более привлекательной выглядит возможность расширения ресурсов персонального компьютера за счет облачных хранилищ и вычислений, однако, использование таких ресурсов может поставить под угрозу безопасность обрабатываемых данных. В последние двадцать лет появилось множество алгоритмов гомоморфного шифрования, позволяющих в частности решить эту задачу. Однако эти алгоритмы проектируются в основном как системы с открытым ключом, предназначенные для долгосрочного хранения и обработки данных. В данной статье предлагается два алгоритма гомоморфного шифрования, оптимизированных для однократной обработки данных. *Цель* Целью работы является разработка системы кодирования информации, позволяющей проводить безопасную обработку данных в публичных облаках. *Результаты* Разработаны две системы скрытых вычислений: первая, основанная на представлении чисел в виде многочленов и вторая, основанная на дальнейшем представлении многочленов в виде набора значений. Разработанные системы позволяют проводить приближённые вычисления над зашифрованными данными без их расшифровки, что позволяет проводить обработку вещественных чисел. Система отличается высоким уровнем защиты и обеспечивает высокую точность вычислений, сравнимую с точностью обеспечиваемой стандартными вычислениями компьютера. Структура зашифрованных данных позволяет проведение параллельных вычислений. Предложенная система позволяет безопасную обработку данных в публичных облачных сетях. Остаётся открытым вопрос оптимальных параметров системы защиты информации, обеспечивающих более высокую точность для ограниченного набора операций, либо постоянную точность для больших наборов операций.

Ключевые слова: облачные вычисления, гомоморфное шифрование