КІБЕРБЕЗПЕКА ТА ЗАХИСТ ІНФОРМАЦІЇ

UDC 004.056.55
DOI https://doi.org/10.32782/2521-6643-2025-2-70.28

Kovalchuk D. M., PhD in Computer Science,
Senior Lecturer
V. N. Karazin Kharkiv National University
ORCID: 0000-0002-8229-836X

METHOD FOR IMPLEMENTING THE SQUARING OPERATION IN THE RABIN CRYPTOSYSTEM BASED ON THE USE OF THE RESIDUE NUMBER SYSTEM

The article proposes a method for implementing the operation of squaring a number in the Rabin cryptosystem, which increases the speed of arithmetic operations by using the Residue Number System (RNS). The proposed approach is based on the transition from the positional numeral system to RNS, which allows computations to be performed independently for each modulus and enables parallel data processing. Unlike traditional methods that rely on sequential execution of arithmetic operations with large numbers, the use of RNS avoids carry propagation, thereby reducing time complexity.

A mathematical model of the process of multiplying two numbers represented in RNS has been developed based on the use of table multiplication coding. This approach takes into account the symmetry properties of table multiplication, allowing the volume of required computations to be reduced to 25% of the full table. Based on this mathematical model, a method for squaring numbers in RNS has been proposed.

A comparative analysis showed that the use of the proposed approach provides a significant performance gain: for 32-bit operands, the speedup is 2048 times, and for 64-bit operands, up to 8192 times compared to the positional numeral system.

The research results confirm the feasibility of applying the Residue Number System in the Rabin cryptosystem for implementing the squaring operation. The proposed method can be effectively used in high-performance cryptographic systems designed for processing large numerical fields. Further research should focus on developing a universal structure for modular computations in RNS for other asymmetric cryptosystems, as well as creating a hardware implementation of the method to practically evaluate its performance.

Key words: *Rabin cryptosystem, Residue Number System, squaring, performance enhancement, parallel computations.*

Ковальчук Д. М. Метод реалізації операції піднесення до квадрату в криптосистемі Рабіна на основі використання системи залишкових класів

У статті запропоновано новий метод реалізації операції піднесення числа до квадрату в криптосистемі Рабіна, який спрямований на суттєве підвищення швидкодії виконання арифметичних операцій завдяки застосуванню системи залишкових класів (СЗК). Основна ідея підходу полягає у переході від традиційної позиційної системи числення до СЗК, що дозволяє виконувати обчислення незалежно для кожного модуля, відкриваючи можливості для паралельної обробки даних. Такий підхід суттєво відрізняється від класичних методів, які оперують великими числами послідовно та обмежені необхідністю обробки переносу розрядів, що підвищує часову складність операцій. Використання СЗК дозволяє усунути цей недолік, забезпечуючи значне скорочення обчислювального часу.

У роботі розроблено математичну модель множення двох чисел у СЗК, що базується на використанні коду табличного множення. Модель враховує властивості симетрії таблиці множення, що дозволяє зменшити обсяг необхідних обчислень до 25 % від повної таблиці На основі цієї моделі запропоновано метод піднесення чисел до квадрату в СЗК, який демонструє високу ефективність для чисел великої розрядності. Метод дозволяє виконувати обчислення незалежно по кожному модулю, що робить його придатним для апаратної реалізації.

Проведений порівняльний аналіз показав, що запропонований підхід забезпечує істотне прискорення обчислень: для 32-розрядних операндів швидкодія зростає у 2048 разів, а для 64-розрядних операндів – до 8192 разів у порівнянні з традиційними методами позиційної системи числення. Це свідчить про високу ефективність використання СЗК у криптосистемі Рабіна та демонструє практичну користь для високопродуктивних криптографічних систем, орієнтованих на обробку великих числових полів.

Отримані результати підтверджують доцільність застосування СЗК для реалізації операцій піднесення до квадрату та відкривають перспективи для подальших досліджень. Подальші напрями включають розробку універсальної структури модульних обчислень у СЗК для інших асиметричних криптосистем, оптимізацію алгоритмів для апаратної

реалізації та практичну оцінку продуктивності запропонованого методу у різних обчислювальних середовищах. Застосування запропонованого підходу може стати ключовим елементом у створенні швидких і надійних криптографічних рішень, що відповідають сучасним вимогам інформаційної безпеки та обробки великих обсягів даних.

Ключові слова: *криптосистема Рабіна, система залишкових класів, піднесення до квадрату, підвищення швидкодії, паралельні обчислення.*

Problem statement. At the current stage of cryptography development, one of the key areas of research is improving the efficiency of basic mathematical operations that form the foundation of cryptographic systems [1]. One such system is the Rabin cryptosystem. The Rabin cryptosystem belongs to the class of asymmetric encryption systems whose security is based on the computational difficulty of residueing large integers [2]. Unlike the RSA system, where encryption involves exponentiation with an arbitrary exponent, the Rabin cryptosystem uses only the squaring operation modulo the product of two prime numbers. This property makes it mathematically elegant and simple to implement, while maintaining a high level of cryptographic strength [3, 4].

For key generation, two large prime numbers p and q are randomly chosen so that they are relatively prime. Their product $n = p \cdot q$ serves as the public key, while the numbers p and q themselves form the private key, which must be kept secret. The choice of such parameters ensures the one-way nature of the encryption function – the inverse transformation is possible only if the residues of n are known [4, 5].

Message encryption consists in transforming the input data block M, which satisfies the condition $0 < M < n$, according to the formula

$$C = M^2 \bmod n, \tag{1}$$

where C is the ciphertext. The encryption process is extremely simple, as it involves only one arithmetic operation – squaring. At the same time, retrieving the original message M from the known C without knowledge of the prime residues of the modulus n is practically impossible due to the computational complexity of the factorization problem [6].

The decryption process involves the use of the secret parameters p and q. At the first stage, the remainders are computed as follows:

$$f = C \bmod p, \quad s = C \bmod q. \tag{2}$$

Next, it is necessary to find the quadratic roots of the obtained values under the corresponding moduli:

$$x^2 \equiv f (\bmod p), \quad y^2 \equiv s (\bmod q). \tag{3}$$

Since each of these equations has two solutions, the total number of possible combinations equals four. Accordingly, a system of congruences is formed:

$$\begin{cases} M \equiv x (\bmod p), \\ M \equiv y (\bmod q), \end{cases} \quad \begin{cases} M \equiv x (\bmod p), \\ M \equiv -y (\bmod q), \end{cases}$$
$$\begin{cases} M \equiv -x (\bmod p), \\ M \equiv y (\bmod q), \end{cases} \quad \begin{cases} M \equiv -x (\bmod p), \\ M \equiv -y (\bmod q). \end{cases} \tag{4}$$

For each of these systems, the corresponding value of M is computed using the Chinese Remainder Theorem [7]. As a result, four potential solutions are obtained, among which only one corresponds to the original message.

However, in the practical implementation of the Rabin cryptosystem, a number of issues arise due to the high time complexity of arithmetic operations in multi-bit numerical fields. Operations such as multiplication, division, or squaring in the traditional positional number system require significant CPU resources, especially when processing numbers longer than 1024 bits – the minimum required security level under modern conditions. This creates the need to search for new approaches to optimizing computations in cryptographic algorithms [8].

One promising direction is the use of the RNS. The application of RNS enables the parallelization of computations, which is not possible when using a positional number system. As a result, there is no need for carry propagation, and the basic operations – addition, subtraction, and squaring – can be performed independently and in parallel, significantly increasing the computational speed of the system [5, 8].

Analysis of recent research and publications. Research on optimizing cryptographic computations using the RNS has been actively developing in recent years. In particular, work [9] proposed an efficient implementation of the Montgomery algorithm for modular multiplication in a minimally redundant RNS, which significantly accelerates reduction operations and reduces overhead for large modular computations. This approach demonstrates potential applicability to squaring operations characteristic of the Rabin cryptosystem; however, the article does not directly address the problem of computing square roots.

Another study [10] developed RNS reduction algorithms that take into account properties of quadratic residuosity, which is especially relevant for the Rabin cryptosystem because decryption requires finding quadratic roots.

The authors propose methods that reduce the number of base multiplications in RNS and optimize computations, yet the concrete integration of these algorithms into the Rabin scheme has not been implemented to date.

Research presented in [11] introduced the concept of a multi-layer recursive RNS, where large moduli are decomposed into layers, enabling arithmetic operations on very large numbers to be performed in parallel within smaller subsystems. This approach can substantially shorten computation time for large numeric fields; however, a specific adaptation of these algorithms to the Rabin cryptosystem is not described.

In [12], the application of RNS for accelerating modular operations in isogeny-based cryptography at the hardware level was investigated, demonstrating significant performance gains when working with large numbers. Nevertheless, that work does not consider squaring operations or the particular requirements of the Rabin cryptosystem.

Additionally, work [13] describes the use of the Residue Number System to implement modular operations in RSA, which substantially reduces the execution time of large-number computations. Although that research focused on RSA, its methodology can be adapted to optimize the squaring operation in Rabin, since both systems employ large moduli and modular arithmetic.

The analysis of existing studies shows that applying RNS to optimize modular computations is a promising direction: it can accelerate basic operations, reduce complexity, and enable parallel computation. However, direct application of these methods specifically to the squaring operation in the Rabin cryptosystem has not yet received sufficient scientific treatment. This gap underlines the relevance of further research and the development of novel methods for implementing the squaring operation in Rabin using RNS to improve the speed and efficiency of cryptographic processes [14].

The purpose of the article is to develop and analyze the efficiency of a method for implementing the squaring operation in the Rabin cryptosystem using the Residue Number System, which will help reduce computational complexity and increase the speed of cryptographic processes without compromising the system's security.

Presentation of the main research material. The squaring operation of a number M is based on multiplying the number $M^2 = M \cdot M$. In the RNS, the multiplication of a number $M_1 = (\mu_{11}, ..., \mu_{1i}, ..., \mu_{1k})$ by a number $M_2 = (\mu_{21}, ..., \mu_{2i}, ..., \mu_{2k})$ is performed by multiplying the corresponding residues μ_{1i} and μ_{2i} modulo the corresponding modulus η_i $(i = \overline{1,k})$ for independently and in parallel across each of the k RNS bases [13, 15]. The range of representable numbers (or the system modulus) is equal to the product of all moduli:

$$N = \eta_1 \cdot \eta_2 \cdot ... \cdot \eta_k. \tag{5}$$

The multiplication of residues μ_{1i} and μ_{2i} modulo η_i can be efficiently implemented using a table-based method that leverages the symmetry properties of the arithmetic table along its diagonals, rows, and columns. Thanks to this approach, only 25% of the full table is required for implementing modular multiplication. A table multiplication code is used for this purpose [15]. Table 1 is constructed with the numerical values of the first residue $\mu_{1□}$ placed horizontally and the second residue μ_{2i} vertically. The cells at the intersections indicate the corresponding results of the modular multiplication $\mu_{1i} \cdot \mu_{2i} \left(\mathrm{mod}\, \eta_i\right)$.

This table exhibits symmetry with respect to the diagonals, vertical, and horizontal lines passing between the numbers $(\eta_i - 1)/2$ and $(\eta_i + 1)/2$ for an odd η_i The symmetry along the left diagonal is determined by the commutativity of the multiplication operation, while the symmetry along the right diagonal is defined by the relation $\left(\eta_i - \mu_{1i}\right)\left(m_i - \mu_{2i}\right) \equiv \mu_{1i} \cdot \mu_{2i} \left(\mathrm{mod}\, \eta_i\right)$. The symmetry with respect to the vertical and horizontal axes is due to the fact that the sum of multiples of numbers is divisible by η_i, i.e.,

$$\begin{aligned} &\mu_{1i} \cdot \mu_{2i} + \mu_{1i}\left(\eta_i - \mu_{2i}\right) \equiv 0 \left(\mathrm{mod}\, \eta_i\right); \\ &\mu_{1i} \cdot \mu_{2i} + \mu_{2i}\left(\eta_i - \mu_{1i}\right) \equiv 0 \left(\mathrm{mod}\, \eta_i\right). \end{aligned} \tag{6}$$

Thus, to reconstruct the entire table, it is sufficient to know only one-eighth of it, which significantly reduces the size of the table required to implement the multiplication operation.

To achieve this, a special encoding of the residues μ_{1i} and μ_{2i} is applied, known as the "table multiplication code" [15-17]. The values of μ_{1i} and μ_{2i}, within the range $\left[0, (\eta_i - 1)/2\right)$ can be encoded arbitrarily. The values within the range $\left[(\eta_i + 1)/2,\ \eta_i - 1\right)$, are encoded according to the rule $\mu_{1i} \cdot \mu_{2i} + \mu_{2i}\left(\eta_i - \mu_{1i}\right) \equiv 0 \left(\mathrm{mod}\, \eta_i\right)$.

To distinguish between the ranges, an index (Table Multiplication Code (TMC)) $\zeta_{\mu_{1i}} \left(\zeta_{\mu_{2i}}\right)$ is introduced, which is defined as follows:

$$\zeta_{\mu_{1i}}, \zeta_{\mu_{2i}} = \begin{cases} 0 \; if \; 0 \le \mu_{1i} \le (\eta_i - 1)/2; \\ 1 \; if \; (\eta_i + 1)/2 \le \mu_{1i} \le \eta_i. \end{cases} \tag{7}$$

Table 1

Table for the implementation of the modular multiplication operation

μ_{2i} \ μ_{1i}	$\mu_{1i}^{(0)}$	…	$\mu_{1i}^{(\frac{\eta-1}{2})}$	$\mu_{1i}^{(\frac{\eta+1}{2})}$	…	$\mu_{1i}^{(\eta-1)}$
$\mu_{2i}^{(0)}$	$(\mu_{1i}^{(0)} \cdot \mu_{2i}^{(0)}) \bmod \eta_i$	…	$\left(\mu_{1i}^{(\frac{\eta-1}{2})} \cdot \mu_{2i}^{(0)}\right) \bmod \eta_i$	$\left(\mu_{1i}^{(\frac{\eta+1}{2})} \cdot \mu_{2i}^{(0)}\right) \bmod \eta_i$	…	$\left(\mu_{1i}^{(\eta-1)} \cdot \mu_{2i}^{(0)}\right) \bmod \eta_i$
…	…	…	…	…	…	…
$\mu_{2i}^{(\frac{\eta-1}{2})}$	$\left(\mu_{1i}^{(0)} \cdot \mu_{2i}^{(\frac{\eta-1}{2})}\right) \bmod \eta_i$	…	$\left(\mu_{1i}^{(\frac{\eta-1}{2})} \cdot \mu_{2i}^{(\frac{\eta-1}{2})}\right) \bmod \eta_i$	$\left(\mu_{1i}^{(\frac{\eta+1}{2})} \cdot \mu_{2i}^{(\frac{\eta-1}{2})}\right) \bmod \eta_i$	…	$\left(\mu_{1i}^{(\eta-1)} \cdot \mu_{2i}^{(\frac{\eta-1}{2})}\right) \bmod \eta_i$
$\mu_{2i}^{(\frac{\eta+1}{2})}$	$\left(\mu_{1i}^{(0)} \cdot \mu_{2i}^{(\frac{\eta+1}{2})}\right) \bmod \eta_i$	…	$\left(\mu_{1i}^{(\frac{\eta-1}{2})} \cdot \mu_{2i}^{(\frac{\eta+1}{2})}\right) \bmod \eta_i$	$\left(\mu_{1i}^{(\frac{\eta+1}{2})} \cdot \mu_{2i}^{(\frac{\eta+1}{2})}\right) \bmod \eta_i$	…	$\left(\mu_{1i}^{(\eta-1)} \cdot \mu_{2i}^{(\frac{\eta+1}{2})}\right) \bmod \eta_i$
…	…	…	…	…	…	…
$\mu_{2i}^{(\eta-1)}$	$\left(\mu_{1i}^{(0)} \cdot \mu_{2i}^{(\eta-1)}\right) \bmod \eta_i$	…	$\left(\mu_{1i}^{(\frac{\eta-1}{2})} \cdot \mu_{2i}^{(\eta-1)}\right) \bmod \eta_i$	$\left(\mu_{1i}^{(\frac{\eta+1}{2})} \cdot \mu_{2i}^{(\eta-1)}\right) \bmod \eta_i$	…	$\left(\mu_{1i}^{(\eta-1)} \cdot \mu_{2i}^{(\eta-1)}\right) \bmod \eta_i$

The method for determining the result of modular multiplication using the TMC is as follows [15].

If two numbers μ_{1i} and μ_{2i} are represented in TMC form as $\mu_{1i} = \left(\zeta_{\mu_{1i}}, \widetilde{\mu_{1i}}\right)$, $\mu_{2i} = \left(\zeta_{\mu_{2i}}, \widetilde{\mu_{2i}}\right)$ then, to obtain their product modulo η_i, it is sufficient to compute $\left(\widetilde{\mu_{1i}} \cdot \widetilde{\mu_{2i}}\right) \bmod \eta_i$ in the TMC. The index $\widetilde{\zeta_{\mu_i}}$ is inverted if $\widetilde{\mu_{1i}}$ and $\widetilde{\mu_{2i}}$ belong to different ranges, that is, $(\mu_{1i} \cdot \mu_{2i}) \bmod \eta_i = \left(\zeta_{\mu_i}, \left(\widetilde{\mu_{1i}} \cdot \widetilde{\mu_{2i}}\right) \bmod \eta_i\right)$ where

$$\zeta_{\mu_i} = \begin{cases} \widetilde{\zeta_{\mu_i}} \ \textit{if} \ \zeta_{\mu_{1i}} \neq \zeta_{\mu_{1i}}; \\ \zeta_{\mu_i} \ \textit{if} \ \zeta_{\mu_{1i}} = \zeta_{\mu_{1i}}; \end{cases} \tag{8}$$

Based on the general mathematical model (7) – (8) for implementing the multiplication of two residues μ_{1i} and μ_{2i} modulo η_i using the table method, we synthesize a mathematical model for multiplying two numbers $M_1 = (\mu_{11}, ..., \mu_{1i}, ..., \mu_{1k})$ and $M_2 = (\mu_{21}, ..., \mu_{2i}, ..., \mu_{2k})$ in the RNS.

The process of table-based implementation of the multiplication operation of two numbers in the residue number system can be represented as follows [17]:

$$\begin{gathered} M = M_1 \cdot M_2 (\bmod N) = M_1 = \\ = \{(\mu_{11}, ..., \mu_{1i}, ..., \mu_{1k}) \cdot (\mu_{21}, ..., \mu_{2i}, ..., \mu_{2k})\} \bmod N = \\ = \{(\mu_{11} \cdot \mu_{21}) \bmod \eta_1, ..., (\mu_{1i} \cdot \mu_{2i}) \bmod \eta_i, ..., (\mu_{1k} \cdot \mu_{2k}) \bmod \eta_k\} = \\ \{\left(\zeta_{\mu_{11}}, \widetilde{\mu_{11}}\right) \cdot \left(\zeta_{\mu_{11}}, \widetilde{\mu_{11}}\right) \bmod \eta_1, ..., \left(\zeta_{\mu_{1i}}, \widetilde{\mu_{1i}}\right) \cdot \left(\zeta_{\mu_{1i}}, \widetilde{\mu_{1i}}\right) \bmod \eta_i, ... \\ ... \left(\zeta_{\mu_{1k}}, \widetilde{\mu_{1k}}\right) \cdot \left(\zeta_{\mu_{1k}}, \widetilde{\mu_{1k}}\right) \bmod \eta_{ik}\} = \{[\zeta_{\mu_1}, \left(\widetilde{\mu_{1i}} \cdot \widetilde{\mu_{2i}}\right) \bmod \eta_i], ... \\ ...[\zeta_{\mu_i}, \left(\widetilde{\mu_{1i}} \cdot \widetilde{\mu_{2i}}\right) \bmod \eta_i], ..., [\zeta_{\mu_k}, \left(\widetilde{\mu_{1k}} \cdot \widetilde{\mu_{2k}}\right) \bmod \eta_k]\} = \\ = (\mu_1, ..., \mu_i, ..., \mu_k). \end{gathered} \tag{9}$$

In this case, for an even number η_i :

$$\widetilde{\mu_{1i}}, \widetilde{\mu_{2i}} = \begin{cases} \mu_{1i}, \mu_{2i}, \ \textit{if} \ 0 \leq \mu_{1i}, \mu_{2i} \leq \eta_i / 2 \ , \\ \eta - \mu_{1i}, \eta - \mu_{2i}, \ \textit{if} \ \eta_i / 2 < \mu_{1i}, \mu_{2i} \leq \eta_i - 1, \end{cases} \tag{10}$$

$$\zeta_{\mu_{1i}},\zeta_{\mu_{2i}}=\begin{cases}0\ if\ 0\le\mu_{1i},\mu_{2i}\le\eta_i/2,\\ 1\ if\ \eta_i/2<\mu_{1i},\mu_{2i}\le\eta_i-1,\end{cases}\tag{11}$$

in this case $0\le\widetilde{\mu_{1i}},\widetilde{\mu_{2i}}\le\eta_i/2$.

If η_i is an odd number, then:

$$\widetilde{\mu_{1i}},\widetilde{\mu_{2i}}=\begin{cases}\mu_{1i},\mu_{2i},\ if\ 0\le\mu_{1i},\mu_{2i}\le(\eta_i-1)/2,\\ \eta-\mu_{1i},\eta-\mu_{2i},\ if\ (\eta_i+1)/2\le\mu_{1i},\mu_{2i}\le\eta_i-1,\end{cases}\tag{12}$$

$$\zeta_{\mu_{1i}},\zeta_{\mu_{2i}}=\begin{cases}0\ if\ 0\le\mu_{1i},\mu_{2i}\le(\eta_i-1)/2,\\ 1\ if\ (\eta_i+1)/2\le\mu_{1i},\mu_{2i}\le\eta_i-1,\end{cases}\tag{13}$$

in this case $0\le\widetilde{\mu_{1i}},\widetilde{\mu_{2i}}\le(\eta_i-1)/2$.

Thus, the relationships (9) – (13) represent the mathematical model of the process of multiplying two numbers $M_1=(\mu_{11},...,\mu_{1i},...,\mu_{1k})$ and $M_2=(\mu_{21},...,\mu_{2i},...,\mu_{2k})$. Let us now consider the case where the number M needs to be squared:

$$\begin{aligned}M^2(\bmod N)&=M\cdot M\ (\bmod N)=\\ &=\{(\mu_1,...,\mu_i,...,\mu_k)\cdot(\mu_1,...,\mu_i,...,\mu_k)\}\bmod N=\\ &=\{(\mu_1\cdot\mu_1)\bmod\eta_1,...,(\mu_i\cdot\mu_i)\bmod\eta_i,...,(\mu_k\cdot\mu_k)\bmod\eta_k\}.\end{aligned}\tag{14}$$

First, we show that for any residue in the RNS, the following mathematical relationship holds [17]:

$$\mu_i^2\bmod\eta_i=(\eta_i-\mu_i)^2\bmod\eta_i\tag{15}$$

Indeed, the square of the residue μ_i^2 can be expressed as $\mu_i^2=\Omega_i\eta_i+\Psi_i\ (0\le\Psi_i\le\eta_i-1)$, which means that $\mu_i^2\bmod\eta_i=\Psi_i$. Then, $\eta_i^2-2\eta_i\mu_i+\mu_i^2=\eta_i(\eta_i-2\mu_i+\Omega_i)+\Psi_i$.. In this case, $(\eta_i^2-2\eta_i\mu_i+\mu_i^2)\bmod\eta_i=\Psi_i$. Thus, this relationship holds true for both even and odd values of η_i. The analytical expression (15) represents a mathematical model of the process used to implement $\mu_i^2\bmod\eta_i$ in practical computations [18]. The structural diagram of this operation $M^2(\bmod N)$ is shown in fig. 1. The operation of the device proceeds as follows: a number μ_i in binary code is fed to the input and stored in the input register. Then, from the output of the decoder, the number μ_i in unary code passes through the corresponding OR logic elements to the input of the encoder, which corresponds to the value $\mu_i^2\bmod\eta_i$. From the encoder's output, the obtained value $\mu_i^2\bmod\eta_i$ in binary form is sent to the output register. It is evident that the key element in the technical implementation of the operation $\mu_i^2\bmod\eta_i$ is the correct coding of the connections between the decoder and the encoder.

Let us consider an example of the practical application of the developed method for exponentiation in the RNS for the Rabin cryptosystem, with the bases $\eta_1=3,\ \eta_2=5,\ \eta_3=7,$ and $N=\eta_1\cdot\eta_2\cdot\eta_3=3\cdot5\cdot7$. The range of code words in the RNS is presented in table 2. The algorithm for implementing the operation of squaring a number modulo $M^2(\bmod N)$ for $\eta_1=3,\ \eta_2=5,\ \eta_3=7,$ is presented in table 3.

Let us consider examples of squaring numbers using the proposed method. Suppose we need to calculate 7^2. In the RNS, this number 7 is represented as $M=(1,2,0)$. In binary code, this value is $M=(01_2,010_2,000_2)$ and is fed to the inputs of the respective decoders. At the output of the encoders, we obtain $M^2=(01_2,100_2,000_2)$ in binary codes, which corresponds to $M^2=(1_{10},4_{10},0_{10})$ in decimal. From table 4, it is seen that (1,4,0) in RNS corresponds to 49 in the positional number system, i.e., 7^2=49, confirming that the result is correct.

Let us determine the time gain when performing the squaring operation in the Rabin cryptographic system as the ratio between the execution time of the operation in the positional number system and that achieved using the proposed method in the RNS. It is known that in a positional number system, the execution time of a multiplication operation is given by $t_{PNS}=2\sigma^2\tau$ where σ is the number of binary digits in the operand representation, and τ is the propagation delay of the "OR" logic elements. In the RNS, due to the parallel nature of computations, the time required for squaring a number is $t_{RNS}=\tau$ where τ is again the propagation delay of the "OR" logic elements. Thus, when using 32-bit operands, the performance gain can be calculated as $t_{PNS}/t_{RNS}=2\sigma^2\tau/\tau=2\cdot32^2\tau/\tau=2048$ times. Similarly, for 64-bit operands, the performance gain is $t_{PNS}/t_{RNS}=2\cdot64^2\tau/\tau=8192$ times. Therefore, the proposed squaring method demonstrates a significant improvement in computational performance, making it highly suitable for practical implementation in Rabin cryptosystems.

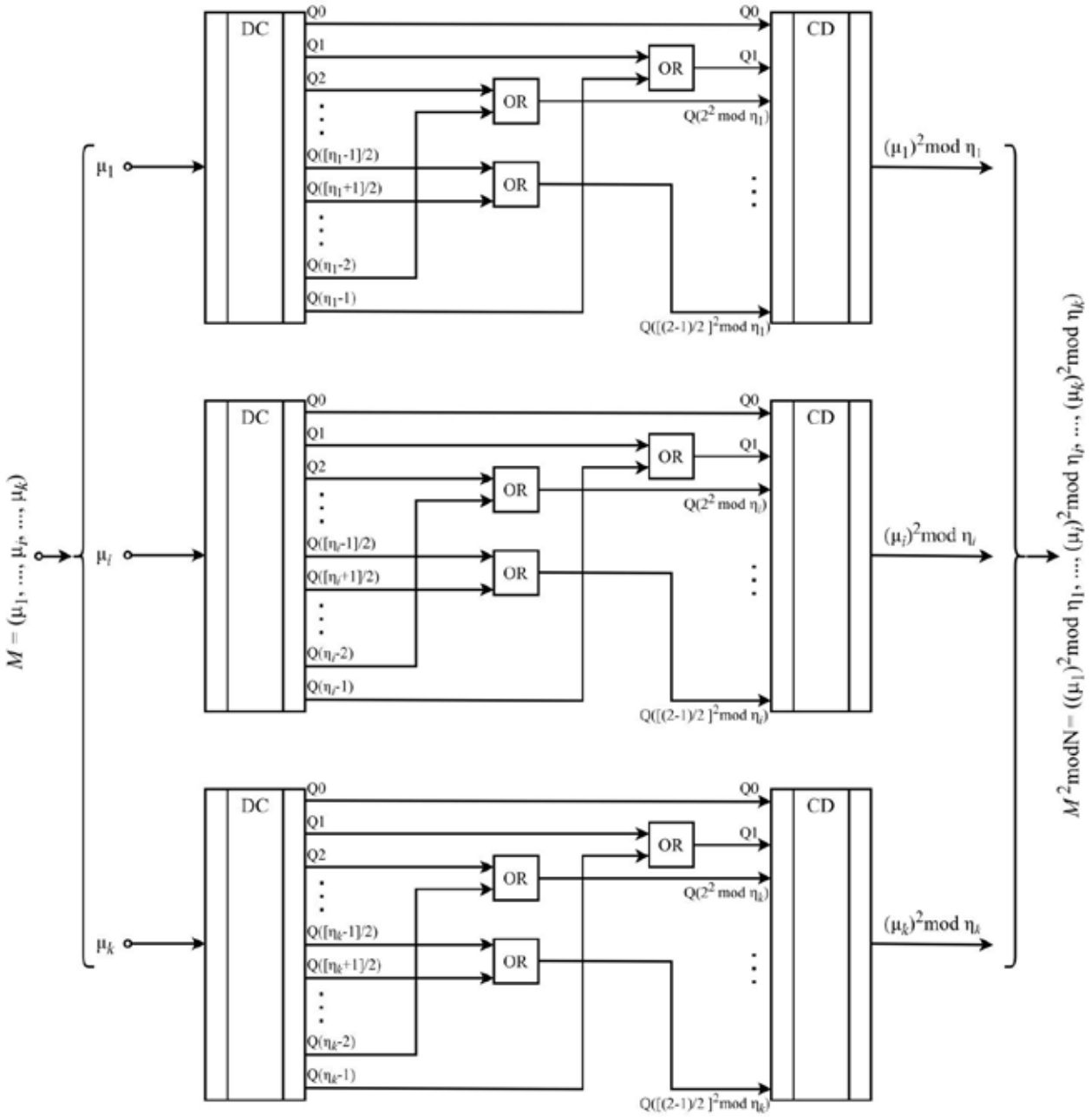

Fig. 1. Scheme of the technical implementation of the operation $M^2 (\mathrm{mod}\, N)$

Table 2

Table of code words in the RNS with the moduli $\eta_1 = 3, \eta_2 = 5, \eta_3 = 7$

$M^{(PNS)}$	$M^{(RNS)}$						$M^{(PNS)}$	$M^{(RNS)}$					
	$\mu_1 = 3$		$\mu_1 = 5$		$\mu_1 = 7$			$\mu_1 = 3$		$\mu_1 = 5$		$\mu_1 = 7$	
0	0_{10}	00_2	0_{10}	000_2	0_{10}	000_2	53	2_{10}	10_2	3_{10}	011_2	4_{10}	100_2
1	1_{10}	01_2	1_{10}	001_2	1_{10}	001_2	54	0_{10}	00_2	4_{10}	100_2	5_{10}	101_2
2	2_{10}	10_2	2_{10}	010_2	2_{10}	010_2	55	1_{10}	01_2	0_{10}	000_2	6_{10}	110_2
3	0_{10}	00_2	3_{10}	011_2	3_{10}	011_2	56	2_{10}	10_2	1_{10}	001_2	0_{10}	000_2
4	1_{10}	01_2	4_{10}	100_2	4_{10}	100_2	57	0_{10}	00_2	2_{10}	010_2	1_{10}	001_2
5	2_{10}	10_2	0_{10}	000_2	5_{10}	101_2	58	1_{10}	01_2	3_{10}	011_2	2_{10}	010_2
6	0_{10}	00_2	1_{10}	001_2	6_{10}	110_2	59	2_{10}	10_2	4_{10}	100_2	3_{10}	011_2
7	1_{10}	01_2	2_{10}	010_2	0_{10}	000_2	60	0_{10}	00_2	0_{10}	000_2	4_{10}	100_2
8	2_{10}	10_2	3_{10}	011_2	1_{10}	001_2	61	1_{10}	01_2	1_{10}	001_2	5_{10}	101_2
9	0_{10}	00_2	4_{10}	100_2	2_{10}	010_2	62	2_{10}	10_2	2_{10}	010_2	6_{10}	110_2
10	1_{10}	01_2	0_{10}	000_2	3_{10}	011_2	63	0_{10}	00_2	3_{10}	011_2	0_{10}	000_2
11	2_{10}	10_2	1_{10}	001_2	4_{10}	100_2	64	1_{10}	01_2	4_{10}	100_2	1_{10}	001_2
12	0_{10}	00_2	2_{10}	010_2	5_{10}	101_2	65	2_{10}	10_2	0_{10}	000_2	2_{10}	010_2
13	1_{10}	01_2	3_{10}	011_2	6_{10}	110_2	66	0_{10}	00_2	1_{10}	001_2	3_{10}	011_2
14	2_{10}	10_2	4_{10}	100_2	0_{10}	000_2	67	1_{10}	01_2	2_{10}	010_2	4_{10}	100_2
15	0_{10}	00_2	0_{10}	000_2	1_{10}	001_2	68	2_{10}	10_2	3_{10}	011_2	5_{10}	101_2
16	1_{10}	01_2	1_{10}	001_2	2_{10}	010_2	69	0_{10}	00_2	4_{10}	100_2	6_{10}	110_2
17	2_{10}	10_2	2_{10}	010_2	3_{10}	011_2	70	1_{10}	01_2	0_{10}	000_2	0_{10}	000_2
18	0_{10}	00_2	3_{10}	011_2	4_{10}	100_2	71	2_{10}	10_2	1_{10}	001_2	1_{10}	001_2
19	1_{10}	01_2	4_{10}	100_2	5_{10}	101_2	72	0_{10}	00_2	2_{10}	010_2	2_{10}	010_2
20	2_{10}	10_2	0_{10}	000_2	6_{10}	110_2	73	1_{10}	01_2	3_{10}	011_2	3_{10}	011_2
21	0_{10}	00_2	1_{10}	001_2	0_{10}	000_2	74	2_{10}	10_2	4_{10}	100_2	4_{10}	100_2
22	1_{10}	01_2	2_{10}	010_2	1_{10}	001_2	75	0_{10}	00_2	0_{10}	000_2	5_{10}	101_2
23	2_{10}	10_2	3_{10}	011_2	2_{10}	010_2	76	1_{10}	01_2	1_{10}	001_2	6_{10}	110_2
24	0_{10}	00_2	4_{10}	100_2	3_{10}	011_2	77	2_{10}	10_2	2_{10}	010_2	0_{10}	000_2
25	1_{10}	01_2	0_{10}	000_2	4_{10}	100_2	78	0_{10}	00_2	3_{10}	011_2	1_{10}	001_2
26	2_{10}	10_2	1_{10}	001_2	5_{10}	101_2	79	1_{10}	01_2	4_{10}	100_2	2_{10}	010_2
27	0_{10}	00_2	2_{10}	010_2	6_{10}	110_2	80	2_{10}	10_2	0_{10}	000_2	3_{10}	011_2
28	1_{10}	01_2	3_{10}	011_2	0_{10}	000_2	81	0_{10}	00_2	1_{10}	001_2	4_{10}	100_2
29	2_{10}	10_2	4_{10}	100_2	1_{10}	001_2	82	1_{10}	01_2	2_{10}	010_2	5_{10}	101_2
30	0_{10}	00_2	0_{10}	000_2	2_{10}	010_2	83	2_{10}	10_2	3_{10}	011_2	6_{10}	110_2
31	1_{10}	01_2	1_{10}	001_2	3_{10}	011_2	84	0_{10}	00_2	4_{10}	100_2	0_{10}	000_2
32	2_{10}	10_2	2_{10}	010_2	4_{10}	100_2	85	1_{10}	01_2	0_{10}	000_2	1_{10}	001_2
33	0_{10}	00_2	3_{10}	011_2	5_{10}	101_2	86	2_{10}	10_2	1_{10}	001_2	2_{10}	010_2
34	1_{10}	01_2	4_{10}	100_2	6_{10}	110_2	87	0_{10}	00_2	2_{10}	010_2	3_{10}	011_2
35	2_{10}	10_2	0_{10}	000_2	0_{10}	000_2	88	1_{10}	01_2	3_{10}	011_2	4_{10}	100_2
36	0_{10}	00_2	1_{10}	001_2	1_{10}	001_2	89	2_{10}	10_2	4_{10}	100_2	5_{10}	101_2
37	1_{10}	01_2	2_{10}	010_2	2_{10}	010_2	90	0_{10}	00_2	0_{10}	000_2	6_{10}	110_2
38	2_{10}	10_2	3_{10}	011_2	3_{10}	011_2	91	1_{10}	01_2	1_{10}	001_2	0_{10}	000_2
39	0_{10}	00_2	4_{10}	100_2	4_{10}	100_2	92	2_{10}	10_2	2_{10}	010_2	1_{10}	001_2
40	1_{10}	01_2	0_{10}	000_2	5_{10}	101_2	93	0_{10}	00_2	3_{10}	011_2	2_{10}	010_2
41	2_{10}	10_2	1_{10}	001_2	6_{10}	110_2	94	1_{10}	01_2	4_{10}	100_2	3_{10}	011_2
42	0_{10}	00_2	2_{10}	010_2	0_{10}	000_2	95	2_{10}	10_2	0_{10}	000_2	4_{10}	100_2
43	1_{10}	01_2	3_{10}	011_2	1_{10}	001_2	96	0_{10}	00_2	1_{10}	001_2	5_{10}	101_2
44	2_{10}	10_2	4_{10}	100_2	2_{10}	010_2	97	1_{10}	01_2	2_{10}	010_2	6_{10}	110_2
45	0_{10}	00_2	0_{10}	000_2	3_{10}	011_2	98	2_{10}	10_2	3_{10}	011_2	0_{10}	000_2
46	1_{10}	01_2	1_{10}	001_2	4_{10}	100_2	99	0_{10}	00_2	4_{10}	100_2	1_{10}	001_2
47	2_{10}	10_2	2_{10}	010_2	5_{10}	101_2	100	1_{10}	01_2	0_{10}	000_2	2_{10}	010_2
48	0_{10}	00_2	3_{10}	011_2	6_{10}	110_2	101	2_{10}	10_2	1_{10}	001_2	3_{10}	011_2
49	1_{10}	01_2	4_{10}	100_2	0_{10}	000_2	102	0_{10}	00_2	2_{10}	010_2	4_{10}	100_2
50	2_{10}	10_2	0_{10}	000_2	1_{10}	001_2	103	1_{10}	01_2	3_{10}	011_2	5_{10}	101_2
51	0_{10}	00_2	1_{10}	001_2	2_{10}	010_2	104	2_{10}	10_2	4_{10}	100_2	6_{10}	110_2
52	1_{10}	01_2	2_{10}	010_2	3_{10}	011_2							

Table 3

The algorithm for implementing the operation of squaring a number modulo $M^2 (\mathrm{mod}\ N)$ for $\eta_1 = 3, \eta_2 = 5, \eta_3 = 7$

Value of the modulus η_i	Value of the remainder μ_i that is applied to the input of the decoder	Bus number with a "1" value at the decoder output	Value at the encoder input	Value $\mu_i^2 \bmod \eta_i$ generated at the encoder output
3	$0_{10}=00_2$	0	00	0
	$1_{10}=01_2$	1	01	1
	$2_{10}=10_2$	2	01	1
5	$0_{10}=000_2$	0	000	0
	$1_{10}=001_2$	1	001	1
	$2_{10}=010_2$	2	100	4
	$3_{10}=011_2$	3	100	4
	$4_{10}=100_2$	4	001	1
7	$0_{10}=000_2$	0	000	0
	$1_{10}=001_2$	1	001	1
	$2_{10}=010_2$	2	100	4
	$3_{10}=011_2$	3	010	2
	$4_{10}=100_2$	4	010	2
	$5_{10}=101_2$	5	100	4
	$6_{10}=110_2$	6	001	1

Conclusions. In the article research investigates the process of implementing the squaring operation in the Rabin cryptosystem using the RNS. The proposed method is based on transitioning from the positional number system to RNS, which enables arithmetic operations to be performed independently for each modulus and ensures a high level of computational parallelism.

A mathematical model of the process of multiplying two numbers represented in the RNS has been developed based on the application of the table multiplication code. This approach takes into account the symmetry properties of the table multiplication, which allows reducing the amount of necessary calculations to 25% of the full table. Based on the mathematical model of multiplying two numbers in the RNS, a method of squaring numbers in the RNS has been proposed.

A comparative analysis has shown that the use of RNS provides a significant increase in performance. For 32-bit operands, the acceleration is 2048 times, and for 64-bit operands – up to 8192 times compared to performing the same operation in a positional number system.

The obtained results confirm the feasibility of applying the Residue Number System in the Rabin cryptosystem for implementing basic arithmetic operations, particularly the squaring operation. The proposed approach increases the efficiency of cryptographic computations without reducing the system's security.

Further research should focus on developing a universal modular computation framework in RNS for other cryptosystems, as well as on the hardware implementation of the proposed method to evaluate its performance under real operating conditions.

Bibliography:

1. Usmani Z. A., Rai M., Khan F. Reconfigurable Cryptoprocessor Design for RSA and Rabin-p Cryptosystems. 2024 26th International Multi-Topic Conference (INMIC), Karachi, Pakistan, 2024. P. 1–6. DOI: 10.1109/INMIC64792.2024.11004308.

2. Yakymenko M., Kasianchuk I., Shylinska R., Shevchuk V., Yatskiv V., Karpinski M. Polynomial Rabin Cryptosystem Based on the Operation of Addition. 2022 12th International Conference on Advanced Computer Information Technologies (ACIT), Ruzomberok, Slovakia, 2022. P. 345–350. DOI: 10.1109/ACIT54803.2022.9913089.

3. Schoinianakis D. Residue arithmetic systems in cryptography: a survey on modern security applications. *Journal of Cryptographic Engineering*, 2020, Vol. 10, P. 249–267. DOI: 10.1007/s13389-020-00231-w.

4. Yakymenko M., Kasianchuk O., Martyniuk S., Martyniuk A., Martyniuk Y., Yakymenko Y. A Symmetric Cryptoalgorithm in a Polynomial Hierarchical Residual Number System. 2025 15th International Conference

on Advanced Computer Information Technologies (ACIT), Sibenik, Croatia, 2025. P. 501–504. DOI: 10.1109/ACIT65614.2025.11185808.

5. Yatskiv V., Yatskiv N., Ivasiev S., Kulyna S., Tsavolyk T., Yatskiv I. The McEliece Cryptosystem Based on the Redundant Residue Number System. 2025 15th International Conference on Advanced Computer Information Technologies (ACIT), Sibenik, Croatia, 2025. P. 573–577. DOI: 10.1109/ACIT65614.2025.11185887.

6. Nykolaychuk Y. M., Yakymenko I. Z., Vozna N. Y. et al. Residue Number System Asymmetric Cryptoalgorithms. *Cybernetics and Systems Analysis*, 2022, Vol. 58, No. 4, P. 611–618. DOI: 10.1007/s10559-022-00494-7.

7. Zhan J., Shiue P. J., Huang S. C., Lowe B. J. Towards a Novel Generalized Chinese Remainder Algorithm for Extended Rabin Cryptosystem. *IEEE Access*, 2020, Vol. 8, P. 26433–26444. DOI: 10.1109/ACCESS.2020.2967396.

8. Nykolaychuk Y. M., Yakymenko I. Z., Vozna N. Y., Kasianchuk M. M. Residue number system asymmetric cryptoalgorithms. *Cybernetics and Systems Analysis*, 2022, Vol. 58, No. 4, P. 611–618.

9. Selianinau M., Woźna-Szcześniak B. An Efficient Implementation of Montgomery Modular Multiplication Using a Minimally Redundant Residue Number System. *Applied Sciences*, 2025, Vol. 15, No. 10, 5332. DOI: 10.3390/app15105332.

10. Kawamura S., Komano Y., Shimizu H. et al. RNS Montgomery Reduction Algorithms Using Quadratic Residuosity. *Journal of Cryptographic Engineering*, 2019, Vol. 9, P. 313–331. DOI: 10.1007/s13389-018-0195-8.

11. Hollmann H. D. L., Rietman R., de Hoogh S., Tolhuizen L. M. G. M., Gorissen P. A Multi-layer Recursive Residue Number System. arXiv, 2018. DOI: 10.48550/arxiv.1801.07561.

12. Jacquemin D., Mert A. C., Roy S. S. Exploring RNS for Isogeny-Based Cryptography. Cryptology ePrint Archive, 2022. URL: https://eprint.iacr.org/2022/1289.

13. Krasnobayev V., Yanko A., Koshman S. Conception of Realization of Cryptographic RSA Transformations with Using of the Residue Number System. *Computer Science and Cybersecurity*, 2016, No. 2, P. 5–12. URL: https://periodicals.karazin.ua/cscs/article/view/6207.

14. Yatskiv V., Kulyna S., Bykovyy P., Maksymyuk T., Sachenko A. Method of Reliable Data Storage Based on Redundant Residue Number System. 2020 IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Dortmund, Germany, 2020. P. 1–4. DOI: 10.1109/IDAACS-SWS50031.2020.9297052.

15. Krasnobayev V., Yanko A., Kovalchuk D. Method of Tabular Implementation of the Arithmetic Operation of Multiplying Two Numbers Represented in the System of Residual Classes. 2022 IEEE 9th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), Kharkiv, Ukraine, 2022. P. 63–68. DOI: 10.1109/PICST57299.2022.10238624.

16. Yatskiv V., Kulyna S., Yatskiv N., Kulyna H. Protected Distributed Data Storage Based on Residue Number System and Cloud Services. 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), Deggendorf, Germany, 2020. P. 796–799. DOI: 10.1109/ACIT49673.2020.9208849.

17. Krasnobayev V. A., Yanko A. S., Kovalchuk D. M. Mathematical Model of the Process of Raising Integers to an Arbitrary Power of a Natural Number in the System of Residual Classes. *Theoretical and Applied Cybersecurity*, 2023, Vol. 5, No. 2, P. 5–14. DOI: 10.20535/tacs.2664-29132023.2.278891.

18. Ochoa-Jiménez E., Rivera-Zamarripa L., Cruz-Cortés N., Rodríguez-Henríquez F. Implementation of RSA Signatures on GPU and CPU Architectures. *IEEE Access*, 2020, Vol. 8, P. 9928–9941. DOI: 10.1109/ACCESS.2019.2963826.

References:

1. Usmani, Z. A., Rai, M., & Khan, F. (2024). Reconfigurable cryptoprocessor design for RSA and Rabin-p cryptosystems. 2024 26th International Multi-Topic Conference (INMIC), Karachi, Pakistan, 1–6. https://doi.org/10.1109/INMIC64792.2024.11004308

2. Yakymenko, M., Kasianchuk, I., Shylinska, R., Shevchuk, V., Yatskiv, V., & Karpinski, M. (2022). Polynomial Rabin cryptosystem based on the operation of addition. 2022 12th International Conference on Advanced Computer Information Technologies (ACIT), Ruzomberok, Slovakia, 345–350. https://doi.org/10.1109/ACIT54803.2022.9913089

3. Schoinianakis, D. (2020). Residue arithmetic systems in cryptography: A survey on modern security applications. *Journal of Cryptographic Engineering*, 10(3), 249–267. https://doi.org/10.1007/s13389-020-00231-w

4. Yakymenko, M., Kasianchuk, O., Martyniuk, S., Martyniuk, A., Martyniuk, Y., & Yakymenko, Y. (2025). A symmetric cryptoalgorithm in a polynomial hierarchical residual number system. 2025 15th International Conference on Advanced Computer Information Technologies (ACIT), Sibenik, Croatia, 501–504. https://doi.org/10.1109/ACIT65614.2025.11185808

5. Yatskiv, V., Yatskiv, N., Ivasiev, S., Kulyna, S., Tsavolyk, T., & Yatskiv, I. (2025). The McEliece cryptosystem based on the redundant residue number system. 2025 15th International Conference on Advanced Computer Information Technologies (ACIT), Sibenik, Croatia, 573–577. https://doi.org/10.1109/ACIT65614.2025.11185887

6. Nykolaychuk, Y. M., Yakymenko, I. Z., Vozna, N. Y., et al. (2022). Residue number system asymmetric cryptoalgorithms. *Cybernetics and Systems Analysis*, 58(4), 611–618. https://doi.org/10.1007/s10559-022-00494-7

7. Zhan, J., Shiue, P. J., Huang, S. C., & Lowe, B. J. (2020). Towards a novel generalized Chinese remainder algorithm for extended Rabin cryptosystem. *IEEE Access*, 8, 26433–26444. https://doi.org/10.1109/ACCESS.2020.2967396

8. Nykolaychuk, Y. M., Yakymenko, I. Z., Vozna, N. Y., & Kasianchuk, M. M. (2022). Residue number system asymmetric cryptoalgorithms. *Cybernetics and Systems Analysis*, 58(4), 611–618.

9. Selianinau, M., & Woźna-Szcześniak, B. (2025). An efficient implementation of Montgomery modular multiplication using a minimally redundant residue number system. *Applied Sciences*, 15(10), 5332. https://doi.org/10.3390/app15105332

10. Kawamura, S., Komano, Y., Shimizu, H., et al. (2019). RNS Montgomery reduction algorithms using quadratic residuosity. *Journal of Cryptographic Engineering*, 9(3), 313–331. https://doi.org/10.1007/s13389-018-0195-8

11. Hollmann, H. D. L., Rietman, R., de Hoogh, S., Tolhuizen, L. M. G. M., & Gorissen, P. (2018). A multi-layer recursive residue number system. arXiv. https://doi.org/10.48550/arxiv.1801.07561

12. Jacquemin, D., Mert, A. C., & Roy, S. S. (2022). Exploring RNS for isogeny-based cryptography. Cryptology ePrint Archive. https://eprint.iacr.org/2022/1289

13. Krasnobayev, V., Yanko, A., & Koshman, S. (2016). Conception of realization of cryptographic RSA transformations with using of the residue number system. *Computer Science and Cybersecurity*, 2, 5–12. https://periodicals.karazin.ua/cscs/article/view/6207

14. Yatskiv, V., Kulyna, S., Bykovyy, P., Maksymyuk, T., & Sachenko, A. (2020). Method of reliable data storage based on redundant residue number system. 2020 IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Dortmund, Germany, 1–4. https://doi.org/10.1109/IDAACS-SWS50031.2020.9297052

15. Krasnobayev, V., Yanko, A., & Kovalchuk, D. (2022). Method of tabular implementation of the arithmetic operation of multiplying two numbers represented in the system of residual classes. 2022 IEEE 9th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), Kharkiv, Ukraine, 63–68. https://doi.org/10.1109/PICST57299.2022.10238624

16. Yatskiv, V., Kulyna, S., Yatskiv, N., & Kulyna, H. (2020). Protected distributed data storage based on residue number system and cloud services. 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), Deggendorf, Germany, 796–799. https://doi.org/10.1109/ACIT49673.2020.9208849

17. Krasnobayev, V. A., Yanko, A. S., & Kovalchuk, D. M. (2023). Mathematical model of the process of raising integers to an arbitrary power of a natural number in the system of residual classes. *Theoretical and Applied Cybersecurity*, 5(2), 5–14. https://doi.org/10.20535/tacs.2664-29132023.2.278891

18. Ochoa-Jiménez, E., Rivera-Zamarripa, L., Cruz-Cortés, N., & Rodríguez-Henríquez, F. (2020). Implementation of RSA signatures on GPU and CPU architectures. *IEEE Access*, 8, 9928–9941. https://doi.org/10.1109/ACCESS.2019.2963826

Дата надходження статті: 19.10.202
Дата прийняття статті: 10.11.2025
Опубліковано: 30.12.2025