Neural network method for base extension in residue number system

M Babenko[1], E Shiriaev[1], A Tchernykh[2,3,4], and E Golimblevskaia[1]

[1]North-Caucasus Federal University, Stavropol, Russia
[2]CICESE Research Center, Ensenada, BC, México
[3]South Ural State University, Chelyabinsk, Russia
[4]Ivannikov Institute for System Programming, Moscow, Russia

E-mail: chernykh@cicese.mx

Abstract. Confidential data security is associated with the cryptographic primitives, asymmetric encryption, elliptic curve cryptography, homomorphic encryption, cryptographic pseudorandom sequence generators based on an elliptic curve, etc. For their efficient implementation is often used Residue Number System that allows executing additions and multiplications on parallel computing channels without bit carrying between channels. A critical operation in Residue Number System implementations of asymmetric cryptosystems is base extension. It refers to the computing a residue in the extended moduli without the application of the traditional Chinese Remainder Theorem algorithm. In this work, we propose a new way to perform base extensions using a Neural Network of a final ring. We show that it reduces 11.7% of the computational cost, compared with state-of-the-art approaches.

1. Introduction

Currently, many cryptosystems use Montgomery multiplication [1] and exponentiation by numbers with high resolution. Often, Redundant Residue Number Systems (RRNS) are used to implement this operation due to the possibility of parallelizing its arithmetic [2]. For scaling RNS operations, a base extension is required to obtain the new extended moduli system.

This operation is the most computationally expensive since traditional methods of converting a number from RRNS to Weighted Number System (WNS) and calculating the Redundant Residue Number System (RRNS) with a new modulo base are used to perform it.

Thus, an important task is to find an efficient algorithm for expanding the RNS base. In this paper, we study traditional algorithms for converting a number from RNS to WNS, two algorithms for the RNS base extension considering integer arithmetic and floating-point arithmetic. We also design and analyze the efficiency of the neural network method for this operation.

The outline of the paper is as follows. Section 2 describes the concept of RNS. Section 3 discusses base extension in Residue Number System. Section 4 describes base extension based on translating a number from RNS into a WNS. Section 5 includes base extension based on scaling a number using the number range function in RNS. Section 6 contains the use of neural networks to expand the base. The study of the effectiveness of algorithms is performed in Section 7. The conclusion is presented in Section 8.

2. The concept of RNS

In this section, we introduce basic concepts of RNS, its operations, and the translation of numbers from WNSC and back.

RNS is a non-positional number system based on modular arithmetic. The representation of a number in RNS is based on moduli comparison of two integers and the Chinese Remainder Theorem (CRT) [3]. Let us consider the following notations (Table 1).

Table 1. Notations used in the work

Notations	Description
i	i-th number
j	j-th number
X	Integer number in WNS
p_i	i-th RNS moduli
x_i	$\|X\|_{p_i}$ remainder of the division X by modulo p_i
P	dynamic range RRNS
P_i	Base modulo
y_i	WNS coefficient
w_i	Core Function weight
B_i	Orthogonal basis
k_i	Approximate coefficient
g_i	Range coefficient
m_i	Orthogonal basis weight
r_A	Number range

RNS can be defined as a set of mutually prime moduli $(p_1, p_2, \dots, p_n)$, whose vector is called the basis of RNS, and its dynamic range $P = \sum_{i=1}^{n} p_i$. Every integer X, which belongs to the range $[0, P-1]$, can be represented in RNS. A set of residues is presented as $(x_1, x_2, \dots, x_n)$, where

$$\begin{gathered} x_1 \equiv X(mod\, p_1) \\ x_2 \equiv X(mod\, p_2) \\ \dots\dots\dots\dots\dots\dots\dots \\ x_n \equiv X(mod\, p_n) \end{gathered}$$

Based on the corollary of CRT, the uniqueness of the representation of non-negative integers from the interval $[0, P-1]$ is guaranteed. The main advantage of representing a number in RNS is determined by the fact that operations such as addition, subtraction, and multiplication can be performed by the formula:

$$\begin{gathered} A * B = (x_1, x_2, \dots, x_n) * (\beta_1, \beta_2, \dots, \beta_n) = \\ = ((x_1 * \beta_1)\, mod\, p_1), ((x_2 * \beta_2)\, mod\, \rho_2), \dots, ((x_n * \beta_n)\, mod\, p_n), \end{gathered} \tag{1}$$

where * denotes operations such as addition, multiplication, or subtraction.

Such operations are called modular [4]. For their execution in RNS, one cycle of processing numerical values is sufficient. This processing takes place in parallel. The value of the number in each category is independent of other categories.

Let RNS be given by a base $(p_1, p_2, \dots, p_n)$ and a number with a residue system $X = (x_1, x_2, \dots, x_n)$. Let $(p_1, p_2, \dots, p_n)$ be RNS bases, then the number X can be represented as

$$X = (y_n \cdot p_1 \cdot p_2 \cdot \dots \cdot p_{n-1} + y_{n-1} \cdot p_1 \cdot p_2 \cdot \dots \cdot p_{n-2} + \dots + y_3 \cdot p_1 \cdot p_2 + y_2 \cdot p_1 + y_1), \tag{2}$$

where $0 \le y_k < p_1 \cdot p_2 \cdot \ldots \cdot p_{k-1} (k = 1, \ldots, n)$ are RNS coefficients.

The ranges of numbers represented in RNS and WNS are the same. We can talk about a one-to-one correspondence between the set of representations of numbers in RNS and WNS.

3. Base Extension in Residue Number System

The base extension is one of the main non-modular operations in RNS [5]. The base extension of the number in RNS is possible for the following reasons. Firstly, it lacks the significance of the order of numbers in a number record. Secondly, both codes and verified numbers are represented as residues, which allow such codes to be considered completely arithmetic.

Based on these properties, we can conclude that modular arithmetic can effectively solve the problem of building both fault-tolerant and high-performance systems. Due to its properties, RNS [6] has a basis for modifying encryption methods, increasing their cryptographic strength, and performance of information security. Many publications demonstrate the practical application of RNS in digital communication systems, global communication systems, wireless networks, fault-tolerant hybrid memory structures, and others. In this paper, we study the calculation of RNS using various algorithms and methods.

This operation may be necessary when performing the division, calculating positional characteristics, or when overflow is detected after adding or multiplying numbers, for example, Montgomery multiplication.

The problem of the base extension can be formulated as follows: To find the residual representation of a number on a new base (new bases). If the representation of a number on other bases is known, to find the residue of division by other numbers. One of the ways to extend the base is to translate the number into a positional number system and calculate the residue of the division by a new module. This path is not efficient in terms of the number of operations. Another method of the base extension is to determine the digit of a number on a new base, using positional characteristics of the number as the range of P', where P' is the range of X in the extended base.

4. Base Extension Based on Translating a Number from RNS into WNS

We study the following algorithms for returning a number in WNS: CRT based method; Translation Method of the Mixed Radix Conversion (MRC); Modified CRT Method; Core function method; Diagonal Function Method; Approximate method.

For these methods, the general operation is the operation of finding the residue of the modulo division. This operation is not displayed in the description of the algorithms since it is obvious.

4.1. CRT based method

CRT based method calculates the formula [7]:

$$X = \sum_{i=1}^{n} \left| x_i \cdot \left(\frac{1}{P_i}\right)_{p_i} \cdot P_i \right|_P, \tag{3}$$

where $\left|\frac{1}{P_i}\right|_{p_i}$ is a multiplicative inversion P_i modulo p_i.

Also, the calculation of this method can be represented as

$$X = |A_1 + A_2 + \cdots + A_n|_P, \tag{4}$$

where $A_i = x_i \cdot \left|\frac{1}{P_i}\right|_{p_i} \cdot P_i$

Algorithm 1. Base extension with CRT

Input: $(x_1, x_2, \ldots, x_n), (p_1, p_2, \ldots, p_n)$ $P = \prod_{i=1}^{n} p_i, B_i = P_i \cdot \left|\frac{1}{P_i}\right|_{p_i}$, for $i = \overline{1,n}$.

Output: X

1. $sum = 0$
2. for $i = 1$ to n do:

2.1 $sum \mathrel{+}= x_1 \cdot B_i$
3. $X = |sum|_P$
4. return X

To illustrate base extension with CRT, let us consider the following Example 1.

Example 1 (Base extension with CRT). The initial data includes a set of residues $x_1 = 1, x_2 = 1, x_3 = 0$ and a set of moduli $p_1 = 3, p_2 = 5, p_3 = 7$. We calculate $P = p_1 \cdot p_2 \cdot p_3 = 105$, $P_1 = \frac{\mathrm{P}}{\mathrm{p_1}} = 35, P_2 = \frac{\mathrm{P}}{\mathrm{p_2}} = 21,$
$P_3 = \frac{\mathrm{P}}{\mathrm{p_3}} = 15$. Then the restoration of the number from (2) is the following.

$$A_1 = x_1 \cdot \left|\frac{1}{P_1}\right|_{p_1} \cdot P_1 = 1 \cdot 2 \cdot 35 = 70, A_2 = x_2 \cdot \left|\frac{1}{P_2}\right|_{p_1} \cdot P_2 = 1 \cdot 1 \cdot 21 = 21,$$
$$A_1 = x_1 \cdot \left|\frac{1}{P_1}\right|_{p_1} \cdot P_3 = 0 \cdot 1 \cdot 15 = 0, X = |A_1 + A_2 + A_3|_{\mathrm{P}} = |70 + 21 + 0|_P$$

4.2. MRC method

MRC performs converting a number from RNS to WNS. Also, in this algorithm, is traced in a generalized form $y_i = |U_i \cdot V_i|_{\rho_i}$, which is defined as the translation coefficient, where U_i defined as follows:

$$U_1 = x_1, U_2 = |x_2 - x_1|_{\rho_1}, U_3 = |x_3 - x_2 - x_1 U_2|_{\rho_2}. \quad (5)$$

Yassin and Moore [8] found that

$$V_1 = 1, V_2 = \left|\frac{1}{p_1}\right|_{p_2} = 1, V_2 = \left|\frac{1}{p_1 p_2}\right|_{p_3} = 1, \quad (6)$$

the number is restored to WNS by (2).

Algorithm 2 Base extension with MRC
Input: $(x_1, x_2, \ldots, x_n), (p_1, p_2, \ldots, p_n)$
Output: X
1. $U_1 = x_1, k_1 = 0, h = 0$
2. for $i = 2$ to n do
2.1 $h \mathrel{*}= n_{i-1}$
2.2 $U_i = (x_i - x_1 - k_{i-1}) \, mod \, p_i$
2.3 $k_i = U_i * h - k_{i-1}$
3. $p = 1, sum = \mathrm{U_0}$
4. for $i = 2$ to n do
5.1 $p \mathrel{*}= p_{i-1}$
5.2 $sum \mathrel{+}= p \cdot |\mathrm{U_i}|_{\mathrm{p_i}} \, mod \, p_i$
5.3 $X = sum$
6. return X

To illustrate base extension with MRC, let us consider the following Example 2.

Example 2 (Base extension with MRC). The initial data contains a set of moduli $p_1 = 127, p_2 = 63, p_3 = 50, \; p_4 = 13$ and a set of residues $x_1 = 78, x_2 = 41, x_3 = 47, x_4 = 7$. $V_1 = 1, V_2 = 1 \; V_3 = 1 \; V_4 = 1$ are the same.

First, we need to find the coefficients:

$y_1 = U_1 \cdot V_1 = 78,$
$y_2 = U_2 \cdot V_2 = (41 - 78)_{63} \cdot 1 = 26,$
$y_3 = U_3 \cdot V_3 = (47 - 78 - 127 \cdot 26)_{50} \cdot 1 = 17,$

$y_3 = U_4 \cdot V_4 = (7 - 78 - 127 \cdot 26 - 127 \cdot 63 \cdot 17)_{13} \cdot 1 = 9.$

Based on the obtained values, the number in WRS is calculated according to (2)

$$78 + 26 \cdot 127 + 17 \cdot 127 \cdot 63 + 9 \cdot 127 \cdot 633 \cdot 50 = 3\,739\,847$$

4.3. Modified CRT Method

In general, this method is the standard CRT method [9] modified by translating coefficients y_i defined in WNS [10], which are calculated as follows:

$$y_1 = \frac{P_1\left|\frac{1}{P_1}\right|_{P_1}^{-1}}{P_1} \text{ and } y_i = \frac{P}{p_1 p_i}\left|\frac{1}{P_i}\right|_{p_i}. \quad (8)$$

A number in WRS is restored by (9) [10]:

$$X = x_1 + p_1|y_1x_1 + y_2x_2|_{p_2} + p_1p_2\left|\left|\frac{y_1x_1+y_2x_2+y_3x_3}{p_2}\right|\right|_{p_3}$$
$$+ \cdots + p_1p_2 \ldots p_{n-1}\left|\left|\frac{y_1x_1 + y_2x_2 + y_3x_3 + \cdots y_nx_n}{p_2p_3 \ldots p_{n-1}}\right|\right|_{p_n}$$

Algorithm 3. Base extension with modified CRT

Input: $(x_1, x_2, \ldots, x_n)$, $(p_1, p_2, \ldots, p_n)$, $(P_1, P_2, \ldots, P_n)$, y_i for $i = \overline{1,n}$.

Output: X

1. $X = a_1$
2. $p = p_1$
3. $temp = 1$
4. for $i = 1$ to n do:

4.1 $X \mathrel{+}= p \cdot ((y_i \cdot x_i)//temp)\ mod\ p_i$

4.2 $p \mathrel{*}= p_i$

4.3 $temp \mathrel{*}= p_i$

5. return X

To illustrate base extension with modified CRT, let us consider the following Example 3.

Example 3 (Base extension with modified CRT). The input in this example is a set of residues $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 4)$ calculated by $p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11$, $P_1 = 385, P_2 = 231, P_3 = 165, P_4 = 105$.

In the beginning, we calculate the coefficients y_i.

$$y_1 = \frac{385\left|\frac{1}{385}\right|_3^{-1}}{3} = 128, y_2 = \frac{1155}{3\cdot5}\left|\frac{1}{231}\right|_5 = 77, y_3 = \frac{1155}{3\cdot7}\left|\frac{1}{165}\right|_7 = 110, y_4 = \frac{1155}{3\cdot11}\left|\frac{1}{105}\right|_{11} = 70$$

Then, based on the calculated coefficients, the number is restored

$$X = 1 + 3 \cdot (128 + 154)_5 + 15 \cdot \left|\left|\frac{128 + 154 + 330}{5}\right|\right|_7 + 105 \cdot \left|\left|\frac{128 + 154 + 330 + 280}{35}\right|\right|_{11}$$
$$= 1 + 3 \cdot 2 + 15 \cdot 3 + 105 \cdot 3 = 367$$

4.4. Core function method

In addition to using sequential MRC to translate a number from RNS to WNS, it uses core functions [11-13]. This method is about calculating the weights of the function (9):

$$w_i \equiv \left|\left|\frac{1}{P_i}\right|_{p_i} \cdot C(P)\right|_{p_i}, \quad (9)$$

where $C(P)$ is a range from moduli range,

$$B_i = P_i \cdot \left|\frac{1}{P_i}\right|_{p_i}, \quad (10)$$

which allow the calculation of these functions [11]:

$$C(X) = |\sum_{i=1}^{n} x_i \cdot C(B_i)|_P, \quad (11)$$

and use them to translate the number

$$X = \frac{P}{C(P)}\left(C(p) + \sum_{i=1}^{n} \frac{w_i}{p_i} x_i\right), \quad (12)$$

Algorithm 4 Base extension with Core Function

Input: $(x_1, x_2, \dots, x_n), (p_1, p_2, \dots, p_n), P, (P_1, P_2, \dots, P_n), C(P), C(B_i), w_i$ for $i = \overline{1,n}$.

Output: X

1. $sum = 0$
2. for $i = 1$ to n do:
2.1 $sum \mathrel{+}= x_i \cdot C(B_i)$
3. $C = |sum|_{C(P)}$
4. $temp = 0$
5. for $i = 1$ to n do:
5.1 $temp \mathrel{+}= (w_i/p_i) \cdot x_i$
6. $X = \frac{P}{C(P)} \cdot (C + temp)$
7. return X

To illustrate base extension with Core Function, let us consider the following Example 4.

Example 4 (Base extension with Core Function). The input in this example is a set of residues $x_1 = 1, x_2 = 2, x_3 = 3$, and $x_4 = 4$ calculated by $p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11, P = \prod_{i=1}^{4} p_i = 1155$, $P_1 = \frac{P}{p_1} = 385$,

$P_2 = \frac{P}{p_2} = 231, P_3 = \frac{P}{p_3} = 165,\ P_4 = \frac{P}{p_4} = 105.$

Let $w_1 = 0, w_2 = 0, w_3 = 0, w_4 = 1$.

Further, the coefficients are calculated

$$B_1 = P_1 \cdot \left|\frac{1}{P_1}\right|_{p_1} = 385 \cdot \left|\frac{1}{385}\right|_3 = 385, B_2 = P_2 \cdot \left|\frac{1}{P_2}\right|_{p_2} = 231 \cdot \left|\frac{1}{231}\right|_5 = 231,$$

$$B_3 = P_3 \cdot \left|\frac{1}{P_3}\right|_{p_3} = 165 \cdot \left|\frac{1}{165}\right|_7 = 330, B_4 = P_4 \cdot \left|\frac{1}{P_4}\right|_{p_4} = 105 \cdot \left|\frac{1}{105}\right|_{11} = 210$$

Calculating the functions, the following values were obtained

$$C(B_1) = \sum_{i=1}^{4} w_i \left\lfloor \frac{B_1}{p_i} \right\rfloor = \left\lfloor \frac{B_1}{p_4} \right\rfloor = 35,\ C(B_2) = \sum_{i=1}^{4} w_i \left\lfloor \frac{B_2}{p_i} \right\rfloor = \left\lfloor \frac{B_2}{p_4} \right\rfloor = 21,$$

$$C(B_3) = \sum_{i=1}^{4} w_i \left\lfloor \frac{B_3}{p_i} \right\rfloor = \left\lfloor \frac{B_3}{p_4} \right\rfloor = 30, C(B_4) = \sum_{i=1}^{4} w_i \left\lfloor \frac{B_4}{p_i} \right\rfloor = \left\lfloor \frac{B_4}{p_4} \right\rfloor = 19 \text{ and}$$

$$C(P) = \sum_{i=1}^{4} w_i \left\lfloor \frac{P}{p_i} \right\rfloor = \left\lfloor \frac{P}{p_4} \right\rfloor = P_4 = 105$$

$$C(X) = \left|\sum_{i=1}^{4} x_i \cdot C(B_i)\right|_{C(P)} = |1 \cdot 35 + 2 \cdot 21 + 3 \cdot 30 + 4 \cdot 19|_{105} = 33$$

Thus, the number is restored from the calculated values

$$X = \frac{P}{C(P)}\left(C(p) + \sum_{i=1}^{4} \frac{w_i}{p_i} x_i\right) = \frac{1155}{105}\left(33 + \frac{4}{11}\right) = 11 \cdot \frac{367}{11} = 367$$

4.5. Diagonal Function Method

The Diagonal function is calculated using the following formula:

$$D(X) = \sum_{i=1}^{n} \left\lfloor \frac{X}{p_i} \right\rfloor$$

$$D(X) = \left| \sum_{i=1}^{n} \hat{k}_i \cdot x_i \right|_{SQ}$$

where $\hat{k}_i = \left| -\frac{1}{p_i} \right|_{SQ}$ and $SQ = \sum_{i=1}^{n} P_i$ is Sum of Quotients.

This method is similar to the Core function method, with some exceptions. The weight $w_i = 1$ for all i of this function is 1 [15-17], so the formulas in Section 4.4 look as follows:

$$X = \frac{P}{SQ}\left(D(X) + \sum_{i=1}^{n} \frac{x_i}{p_i}\right) \quad (13)$$

Algorithm 5. Base extension with Diagonal function

Input: $(x_1, x_2, \dots, x_n), (p_1, p_2, \dots, p_n), P, (P_1, P_2, \dots, P_n), SQ, \hat{k}_i = \left| -\frac{1}{p_i} \right|_{SQ}$ for $i = \overline{1,n}$.

Output: X

1. $sum = 0$
2. for $i = 1$ to n do:
2.1 $sum \mathrel{+}= x_i \cdot \hat{k}_i$
3. $D = |sum|_{SQ}$
4. $temp = 0$
5. for $i = 1$ to n do:
5.1 $temp \mathrel{+}= (w_i/p_i) \cdot x_i$
6. $X = \frac{P}{SQ}(D(X) + temp)$
7. return X

To illustrate base extension with Diagonal function, let us consider the following Example 5.

Example 5 (Base extension with Diagonal function). The input in this example is a set of residues $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 4$ calculated by $p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11$, $P = \prod_{i=1}^{4} p_i = 1155$, $P_1 = \frac{P}{p_1} = 385, P_2 = \frac{P}{p_2} = 231$, $P_3 = \frac{P}{p_3} = 165$, $P_4 = \frac{P}{p_4} = 105$, and $SQ = P_1 + P_2 + P_3 + P_4 = 886$. The coefficients Diagonal function are calculated as follows:

$$\hat{k}_1 = \left| -\frac{1}{p_1} \right|_{SQ} = \left| -\frac{1}{3} \right|_{886} = 295, \hat{k}_2 = \left| -\frac{1}{p_2} \right|_{QS} = \left| -\frac{1}{5} \right|_{886} = 177, \hat{k}_3 = \left| -\frac{1}{p_3} \right|_{SQ} = \left| -\frac{1}{7} \right|_{886} = 253,$$

$$\hat{k}_4 = \left| -\frac{1}{p_4} \right|_{SQ} = \left| -\frac{1}{11} \right|_{886} = 161$$

Calculating D(X), we obtained:

$$D(X) = \left| \sum_{i=1}^{4} \hat{k}_i \cdot x_i \right|_{SQ} = |295 \cdot 1 + 177 \cdot 2 + 253 \cdot 3 + 161 \cdot 4|_{886} = |2052|_{886} = 280$$

Thus, the number is restored from the calculated values:

$$X = \frac{P}{SQ}\left(D(X) + \sum_{i=1}^{n} \frac{x_i}{p_i}\right) = \frac{1155}{886}\left(280 + \frac{1}{3} + \frac{2}{5} + \frac{3}{7} + \frac{4}{11}\right) = \frac{1155}{886} \cdot \frac{325162}{1155} = 367$$

4.6. Approximate method

An approach of the approximate method is to modify CRT using fractional values, which first defined in [16], to determine the sign of a number in RNS, so we can set the approximate coefficient (14).

$$k_i = \left\lceil \frac{|1/P_i|_{p_i}}{p_i} \cdot 2^N \right\rceil, \quad (14)$$

where $N = \lceil \log_2(P \cdot \mu) \rceil$, and $\mu = \sum_{i=1}^{n} p_i - n$. Next, the number is calculated as (15)

$$X = \left\lfloor \frac{|\sum_{i=1}^{n} k_i \cdot x_i|_{2^N} \cdot P}{2^N} \right\rfloor, \tag{15}$$

Algorithm 6. Base extension with CRT approximate method

Input: $(x_1, x_2, \ldots, x_n), (p_1, p_2, \ldots, p_n), P, (P_1, P_2, \ldots, P_n), N, k_i = \left\lceil \frac{|1/P_i|_{p_i}}{p_i} \cdot 2^N \right\rceil$ for $i = \overline{1, n}$.

Output: X

1. $sum = 0$
2. for $i = 1$ to n do:
2.1 $sum += k_i \cdot x_i$
3. $sum = sum$ AND $2^N - 1$
4. $X = (sum \cdot P) \gg N$
5. return X

To illustrate base extension with CRT approximate method, let us consider the following Example 6.

Example 6 (Base extension with CRT approximate method). The input in this example is a set of residues $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 4$ calculated by $p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11, P = \prod_{i=1}^{4} p_i = 1155,\ P_1 = \frac{P}{p_1} = 385, P_2 = \frac{P}{p_2} = 231,\ P_3 = \frac{P}{p_3} = 165,\ P_4 = \frac{P}{p_4} = 105,\quad \mu = \sum_{i=1}^{n} p_i - n = 3 + 5 + 7 + 11 - 4 = 22$ and $N = \lceil \log_2(P \cdot \mu) \rceil = \lceil \log_2(1155 \cdot 22) \rceil = 15$.

Calculating the coefficients, the following values were obtained

$$k_1 = \left\lceil \frac{|1/P_1|_{p_1}}{p_1} \cdot 2^N \right\rceil = \left\lceil \frac{|1/385|_3}{3} \cdot 2^{15} \right\rceil = 10923, k_2 = \left\lceil \frac{|1/P_2|_{p_2}}{p_2} \cdot 2^N \right\rceil = \left\lceil \frac{|1/231|_5}{5} \cdot 2^{15} \right\rceil = 6554,$$

$$k_3 = \left\lceil \frac{|1/P_3|_{p_2}}{p_3} \cdot 2^N \right\rceil = \left\lceil \frac{|1/165|_7}{7} \cdot 2^{15} \right\rceil = 9363, k_4 = \left\lceil \frac{|1/P_4|_{p_4}}{p_4} \cdot 2^N \right\rceil = \left\lceil \frac{|1/105|_{11}}{11} \cdot 2^{15} \right\rceil = 5958$$

Calculate $|\sum_{i=1}^{n} k_i \cdot x_i|_{2^N}$, we have:

$$\left| \sum_{i=1}^{n} k_i \cdot x_i \right|_{2^N} = |10923 \cdot 1 + 6554 \cdot 2 + 9363 \cdot 3 + 5958 \cdot 4|_{2^{15}} = |75952|_{2^{15}} = 10416$$

Thus, based on the coefficients, we can translate the number

$$X = \left\lfloor \frac{|\sum_{i=1}^{n} k_i \cdot x_i|_{2^N} \cdot P}{2^N} \right\rfloor = \left\lfloor \frac{10416 \cdot 1155}{2^{15}} \right\rfloor = 367$$

5. Base Extension Based on Scaling a Number with the Number Range Function in RNS

We concentrate on two algorithms for extension of the range of the base:

- Extension method using integer arithmetic.
- Extension method using floating point arithmetic.

For both methods, the general operation is the operation of finding the residue of the division modulo. Therefore, this operation is not displayed in the description of the algorithms, since it is obvious.

5.1. Extension method using integer arithmetic

The first method uses the equation as shown in (16) [19].

$$x_{n+j} = \left| \sum_{i=1}^{n} \left| x_i \cdot \left| \frac{1}{P_i} \right|_{p_i} \right|_{p_i} \cdot |P_i|_{p_j'} - v \cdot |P|_{p_j'} \right|_{p_j'}, \tag{16}$$

This method can be considered as calculating CRT without decreasing the dynamic modulo range P. This means that the converted value can have multiples of P.

Algorithm 7. Base extension with new CRT

Input: $(x_1, x_2, \ldots, x_n), (p_1, p_2, \ldots, p_n, p_j), P, v, |P|_{p_j'}, (P_1, P_2, \ldots, P_n), p_j', \left|\frac{1}{P_i}\right|_{p_i}$ and $|P_i|_{p_j'}$ for $i = \overline{1, n}$.

Output: x_j

1. $\text{sum} = -\text{v} \cdot |\text{P}|_{\text{p}_\text{j}'}$

1. for $\text{i} = 1$ to n do:

1.1 $\text{sum} \mathrel{+}= \left| \text{x}_\text{i} \cdot \left| \frac{1}{\text{P}_\text{i}} \right|_{\text{p}_\text{i}} \right|_{\text{p}_\text{i}} \cdot |\text{P}_\text{i}|_{\text{p}_\text{j}'}$

2. $\text{x}_\text{j} = \text{sum} \bmod \text{p}_\text{j}'$

3. return x_j

To illustrate base extension with new CRT, let us consider the following Example 7.

Example 7 (Base extension with new CRT). The input in this example is a set of residues $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 4$ calculated by $p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11$, base extends by $p_1' = 17$, $P = \prod_{i=1}^{4} p_i = 1155$, $P_1 = \frac{P}{p_1} = 385, P_2 = \frac{P}{p_2} = 231$, $P_3 = \frac{P}{p_3} = 165$, $P_4 = \frac{\text{P}}{\text{p}_4} = 105$, and $|P_1|_{p_1'} = |385|_{17} = 11$, $|P_2|_{p_1'} = |231|_{17} = 10$, $|P_3|_{p_1'} = |165|_{17} = 12$, $|P_4|_{p_1'} = |105|_{17} = 3$, $\left|\frac{1}{P_1}\right|_{p_1} = \left|\frac{1}{385}\right|_3 = 1, \left|\frac{1}{P_2}\right|_{p_2} = \left|\frac{1}{231}\right|_5 = 1, \left|\frac{1}{P_3}\right|_{p_3} = \left|\frac{1}{165}\right|_7 = 2, \left|\frac{1}{P_4}\right|_{p_4} = \left|\frac{1}{105}\right|_{11} = 2$, and $|P|_{p_j'} = |1155|_{17} = 16$.

We use from Example 6 $\sum_{\text{i}=1}^{\text{n}} \text{k}_\text{i} \cdot \text{x}_\text{i} = 10923 \cdot 1 + 6554 \cdot 2 + 9363 \cdot 3 + 5958 \cdot 4 = 75952$ then $\text{v} = \left\lfloor \frac{\sum_{\text{i}=1}^{\text{n}} \text{k}_\text{i} \cdot \text{x}_\text{i}}{2^\text{N}} \right\rfloor = \left\lfloor \frac{75952}{2^{15}} \right\rfloor = 2$.

$$x_{4+1} = \left| \sum_{i=1}^{n} \left| x_i \cdot \left| \frac{1}{P_i} \right|_{p_i} \right|_{p_i} \cdot |P_\text{i}|_{\text{p}_\text{j}'} - v \cdot |P|_{p_j'} \right|_{p_j'}$$
$$= ||1 \cdot 1|_3 \cdot 11 + |2 \cdot 1|_5 \cdot 10 + |3 \cdot 2|_7 \cdot 12 + |4 \cdot 2|_{11} \cdot 3 - 2 \cdot 16|_{17} = 10$$

5.2. Extension method using floating-point arithmetic

This method, instead of an approximate conversion (a conversion with the excess flow), estimates v and uses the exact value for the conversion [19, 21-25]. Equation (17) can be reformulated to find v as follows:

$$v = \left\lfloor \sum_{i=1}^{n} \left| x_i \cdot \left| \frac{1}{P_i} \right|_{p_i} \right|_{\text{p}_\text{i}} \cdot \frac{1}{\text{p}_\text{i}} \right\rfloor, \tag{17}$$

The key problem of this method is to ensure that the estimate of v is correct. Since v is estimated using floating-point operations, errors due to limited accuracy can occur and lead to a value of v to be equal to one.

Algorithm 8. Base extension with CRT fractions number

Input: $(\text{x}_1, \text{x}_2, \ldots, \text{x}_\text{n}), (\text{p}_1, \text{p}_2, \ldots, \text{p}_\text{n}, \text{p}_\text{j}), \text{P}, (\text{P}_1, \text{P}_2, \ldots, \text{P}_\text{n})$, for $\text{i} = \overline{1, \text{n}}$.

Output: a_j

1. for 1 to n do:

1.1 $\text{v}_\text{i} = \frac{1}{\text{p}_\text{i}} \cdot \text{x}_\text{i} \bmod \text{p}_\text{i} \cdot \left| \frac{1}{\text{P}_\text{i}} \right|_{\text{p}_\text{i}}$

2. for 1 to n do:

2.1 $\text{F} \mathrel{+}= (\text{x}_\text{i} \cdot \text{p}_\text{i})_{\text{p}_\text{i}} \cdot \frac{\text{P}}{\text{p}_\text{i}} - \text{v}_\text{i} \cdot \text{P}_\text{i}$

3. $\text{a}_\text{j} = \text{F} \bmod \text{p}_\text{j}$

To illustrate base extension with CRT fractions number, let us consider the following Example 8.

Example 8 (Base extension with CRT fractions number). The input in this example is the set of residues $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 4$) calculated by moduli $p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11$. The

base is extended by 15, $P_1 = 385, P_2 = 231, P_3 = 165, P_4 = 105$. We use from Example 7 $\left|\frac{1}{P_1}\right|_{p_1} = \left|\frac{1}{385}\right|_3 = 1, \left|\frac{1}{P_2}\right|_{p_2} = \left|\frac{1}{231}\right|_5 = 1, \left|\frac{1}{P_3}\right|_{p_3} = \left|\frac{1}{165}\right|_7 = 2, \left|\frac{1}{P_4}\right|_{p_4} = \left|\frac{1}{105}\right|_{11} = 2$

$$v = \left\lfloor \sum_{i=1}^{n} \left| x_i \cdot \left|\frac{1}{P_i}\right|_{p_i} \right|_{p_i} \cdot \frac{1}{\mathrm{p_i}} \right\rfloor = \left\lfloor \frac{|1 \cdot 1|_3}{3} + \frac{|2 \cdot 1|_5}{5} + \frac{|3 \cdot 2|_7}{7} + \frac{|4 \cdot 2|_{11}}{11} \right\rfloor = \left\lfloor 2\frac{367}{1155} \right\rfloor = 2$$

Using (16) let's calculate x_{4+1}, we have:

$$x_{4+1} = \left| \sum_{i=1}^{n} \left| x_i \cdot \left|\frac{1}{P_i}\right|_{p_i} \right|_{p_i} \cdot |P_i|_{\mathrm{p_j'}} - v \cdot |P|_{p_j'} \right|_{p_j'}$$

$$= ||1 \cdot 1|_3 \cdot 11 + |2 \cdot 1|_5 \cdot 10 + |3 \cdot 2|_7 \cdot 12 + |4 \cdot 2|_{11} \cdot 3 - 2 \cdot 16|_{17} = 10$$

Using rational numbers, allows reducing computational complexity of calculation v.

6. Base Extension with NN

To increase efficiency of the base extension, we use NN [20]. The calculation of the new residue is based on the range of the number, which can be defined as:

$$r_A = |\sum_{i=1}^{n} x_i g_i|_{p_n}, \tag{18}$$

where $g_i = \left|\frac{m_i}{p_i}\right|_{p_n}$, m_i is orthogonal basis weight, $\forall i, i \neq n$; $g_n = p_n - \left|\frac{1}{P}\right|_{p_n}$.

Based on CRT and orthogonal bases, the number A in the base $\mathrm{p_1}, \mathrm{p_2}, \ldots, \mathrm{p_{n-1}}$ can be written as:

$$X = \sum_{i=1}^{n} a_i B_i - x_j P \tag{19}$$

Thus, substituting (19) into (18) we obtain the following:

$$X = \left| \sum_{i=1}^{n-1} x_i |B_i|_{p_j} + x_j (p_j - |P|_{p_j}) \right|_{p_j} \tag{20}$$

For base extension, it is necessary:

- Calculate number range x_j in the base system $p_1, p_2, \ldots, p_{n-1}$ according to the expression (18);
- Find the residue x_j according to the (19).

The proposed method of base extension is characterized by the calculation of small modulo $\mathrm{p_n}$ instead of calculation of the large modulo P in traditional CRT. The residue of the number on an extendable base can be obtained using modular NN. Moreover, the constants of expressions (19), (20) can be calculated in advance and used in the network structure. NN presented in Figure 1.

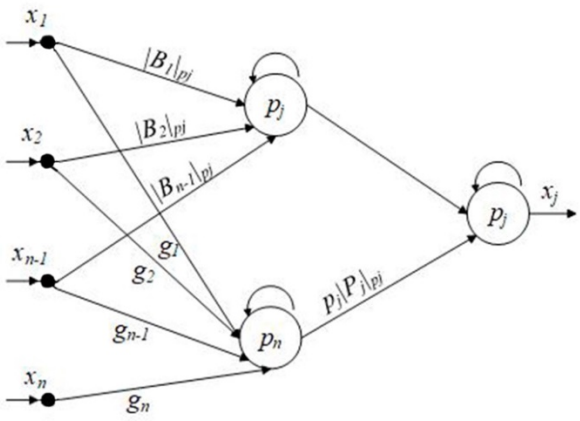

Figure 1. NN architecture of modular code base system extension

The input network receives modular values $x_1 \div x_n$. At the first stage, the modular neural network modulo p_n by weighted summation of the modular values of the number $x_1 \div x_n$ with coefficients $g_1 \div g_n$ calculates the range of a number r_A, and the modular network modulo p_j calculates the value $\sum_{i=1}^{n-1} a_i |B_i|_{p_{n+1}}$. The second step is calculating $x_j = |X|_{p_j}$ using a computational model (20).

Each set of bases of modular code is characterized by orthogonal bases, based on which, extending the base system, it is necessary to recalculate the bases $B'_i, i = \overline{1, n+1}$. For recalculation, the input data includes orthogonal bases $B_i, i = \overline{1, n}$, base systems $p_1, p_2, \dots, p_n$ and the values of the extendable base p_j. Thus, we get the following expression:

$$m'_i P'_i \equiv m_i P_i (mod\ p_i) \tag{21}$$

where m'_i is the orthogonal basis weight B'_i. Based on the fact that $P'_i = P'/p_i$ and P_i are prime, the orthogonal bases of the extended base system can be calculated as follows

$$B'_i \equiv \frac{P'}{p_i} \cdot \left|\frac{m_i}{p_j}\right|_{p_i} \tag{21}$$

To solve this problem in a neural network basis, it is necessary to calculate two constants: $\left|\frac{1}{p_j}\right|_{p_i}$ and $P'_i = \frac{P'}{p_i}$. Figure 2 shows the NN structure.

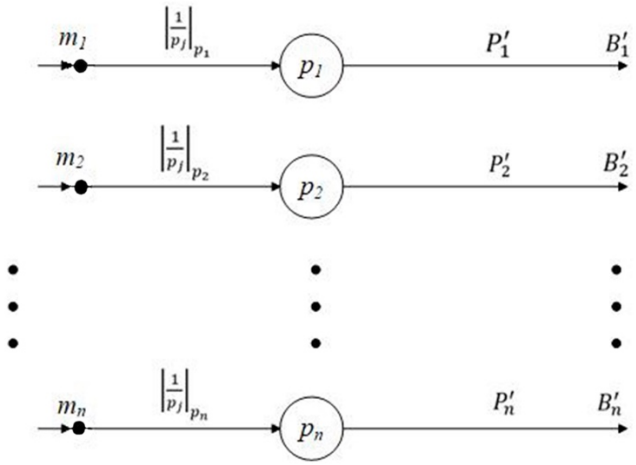

Figure 2. NN structure of recalculation of bases of the extended base

The proposed algorithm has less computational complexity compared to the methods described above. However, the method involves multiplying by pre-calculated constants. Assuming that these constants are usually known in advance, we can conclude that the algorithm is more efficient.

To illustrate base extension with NN, let us consider the following example.

Example 9. The initial data is a set of residues $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 2$ and a set of moduli $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7$. Then the extension of the base modulo $p_j = 11$ is the following.
The representation range of numbers is in a system with bases p_1, p_2, p_3 $P = 2 \cdot 3 \cdot 5 = 30$, orthogonal bases are equal $B_1 = 15, B_2 = 10, B_3 = 6$. We calculate the range, while $g_1 = 4, g_2 = 5, g_3 = 3, g_4 = 3$:$r_A = |4 \cdot 1 + 5 \cdot 2 + 3 \cdot 3 + 3 \cdot 2|_7 = |4 + 10 + 9 + 6|_7 = 1$..

Putting the obtained value in (25) we get:

$$x_j = |1 \cdot |15|_{11} + 2 \cdot |10|_{11} + 3 \cdot |6| + 1 \cdot (11 - |30|_{11})|_{11} = |4 + 20 + 18 + 3|_{11} = 1.$$

7. Performance Analysis

For the implementation of the algorithms, Python was chosen. It uses long arithmetic (work with high-bit numbers) without connecting additional libraries and APIs.

We consider 128-bit numbers, sets of 7 moduli (6 in the “vector length”, 1 extending range) with a 24-64 bit lengths.

All methods are implemented as separate functions. Each function is called separately to work with the already calculated RNS for the "vector length".

The study is carried out in 2 stages. The first one considers the efficiency of algorithms with a static "vector length" and a dynamic modulo length in bits. The second stage is the opposite of the first stage: the length of the moduli is static, and the “vector length” is dynamic.

7.1. The First Stage of the Research

Figure 3 shows execution time obtained at the first stage. Our algorithms show the best results.

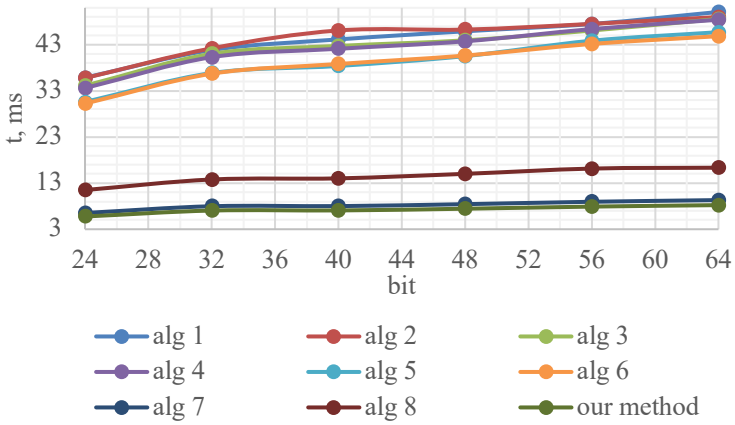

Figure 3. Execution time of base extension algorithms versus bit depth

The growth dynamics of the execution time of the algorithms is uneven. It can be explained by the fact that as the length of the numbers of moduli increases, their basis increases by 10^n times, where n is the difference in the length of numbers.

It also depends on many other parameters that can affect the speed of a function in Python.

The most important result is the fact that algorithms with integer arithmetic have shown their efficiency equal to about 200% and, unlike traditional algorithms, work more stably. Based on the analysis of NN, we can see the advantage of this method over others. In general, we can observe a low drop in productivity with an increase in the capacity of the moduli, which allows us to declare the promise of using this method as the best for systems with a dynamic range of RNS moduli.

7.2. The Second Stage of the Research

The second stage of research shows a similar result to what was obtained in the first stage (Figure 4).

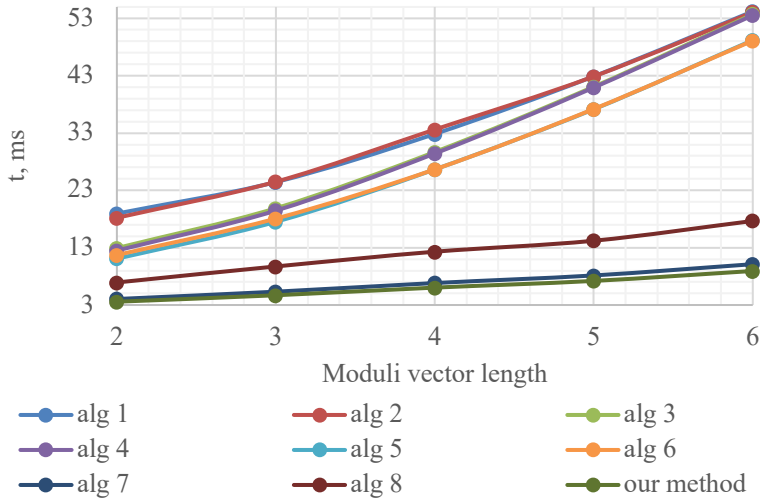

Figure 4. Execution time of base extension algorithms versus the moduli vector length

With a vector length of six, the execution time of traditional algorithms is almost identical, except for the approximate method. Arithmetic-based algorithms show better performance. With a vector length of two, a performance is increased about 100%. With a vector length of six is 400% faster. We show that NN is the most suitable method, since this method is the most productive.

8. Conclusion

In this paper, we studied the algorithms for base extension in RNS. We analyze algorithms for both traditional base extension methods and methods implemented using integer and floating-point arithmetic. We show the superiority of the NN-based algorithm. It uses machine learning to calculate a new residue. The number of pre-calculated constants is small and does not require large computing power. Thus, the result of this work is the definition of the base extension algorithm using NN as the most suitable for practical use in various computing systems with RNS.

Acknowledgments

This work was supported by a grant from the President of the Russian Federation MK-341.2019.9.

References

[1] Eldridge S.E. and Walter C. D.. Hardware implementation of Montgomery's modular multiplication algorithm. IEEE Transaction on Computers, 42 (6): 693-699, June 1993.

[2] Parhami B. RNS representations with redundant residues. Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference, 2001, vol. 2, pp. 1651–1655.

[3] Inyutin S. A. Modular arithmetic in Meot archaeological culture - Mongol-Tatar invasion. - M.: Big Russian Encyclopedia, 2012. - (Big Russian Encyclopedia: [in 35 vols.] / Ch. Ed. Yu. S. Osipov; 2004—2017, vol. 20). - ISBN 978-5-85270-354-5.

[4] Sagalovich Yu. L. Introduction to algebraic codes - 2nd ed. - M.: IPPI RAS, 2010 .-- 320 p. - ISBN 978-5-901158-14-2

[5] Lavrinenko A.N., Chervyakov N.I. Research of non-modular operations in the system of residual classes in Scientific statements of Belgorod State University. Series: Economics. Computer science. 2012. No. 1-1 (120).

[6] Erdnieva N. S. The use of special moduli of the Residue Numbers System for redundant representations in Vestnik ASTU. Series: Management, Computing and Informatics. 2013. No2.

[7] Bi G., Jones E.V. Fast conversion between binary and Residue Numbers. Electron Lett. 24, 1195–1197 (1988)
[8] H.M. Yassine, W.R. Moore, Improved Mixed radix conversion for residue number system architectures. Proc. IEE Part G 138, 120–124 (1991)
[9] Wang Y., Residue to binary converters based on New Chinese Residue theorems. IEEE Trans. Circuits Syst. II 47, 197–205 (2000)
[10] Bi S., Gross W.J., The Mixed-Radix Chinese Residue Theorem and its applications to Residue comparison. IEEE Trans. Comput. 57, 1624–1632 (2008)
[11] Miller D.D. et al., Analysis of a Residue Class Core Function of Akushskii, Burcev and Pak, in RNS Arithmetic: Modern Applications in DSP, ed. by G.A. Jullien (IEEE Press, Piscataway, 1986)
[12] Gonnella J., The application of core functions to residue number systems. IEEE Trans. Signal Process. SP-39, 69–75 (1991)
[13] Burgess N., Scaled and unscaled residue number systems to binary conversion techniques using the core function, in Proceedings of 13th IEEE Symposium on Computer Arithmetic, pp 250–257 (1997)
[14] Krishnan R., Ehrenberg J., Ray G., A core function-based residue to binary decoder for RNS filter architectures, in Proceedings of 33rd Midwest Symposium on Circuits and Systems, pp. 837–840 (1990)
[15] Dimauro G., Impedevo S., Pirlo G. A new technique for fast number comparison in the Residue Number system. IEEE Trans. Comput. 42, 608–612 (1993)
[16] Dimauro G., Impedevo S., Pirlo G., Salzo A., RNS architectures for the implementation of the diagonal function. Inf. Process. Lett. 73, 189–198 (2000)
[17] Pirlo G., Impedovo D., A new class of monotone functions of the Residue number system. Int. J. Math. Models Methods Appl. Sci. 7, 802–809 (2013)
[18] Hung C.Y. and Parhami B., An approximate sign detection method for residue numbers and its application to RNS division, Comput. Math. Appl. 27 (4) (1994), pp. 23–35
[19] Harvey D. Faster Arithmetic for Number-Theoretic Transforms. Journal of Symbolic Computation, 60: 113–119, 2014.
[20] Vasetskiy V.V. The implementation of artificial neural networks in a non-positional number system for automated control systems in Journal of SFU. Technics and techology. 2018. No1
[21] Chervyakov N, Babenko M, Tchernykh A, Kucherov N, Miranda-López V and Cortés-Mendoza J M 2019 AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security *Futur. Gener. Comput. Syst.* **92** 1080–92
[22] Tchernykh A, Babenko M, Chervyakov N, Miranda-Lopez V, Avetisyan A, Drozdov A Y, Rivera-Rodriguez R, Radchenko G and Du Z 2020 Scalable Data Storage Design for Non-Stationary IoT Environment with Adaptive Security and Reliability *IEEE Internet Things J.*
[23] Tchernykh A, Babenko M, Chervyakov N, Miranda-López V, Kuchukov V, Cortés-Mendoza J M, Deryabin M, Kucherov N, Radchenko G and Avetisyan A 2018 AC-RRNS: Anti-collusion secured data sharing scheme for cloud storage *Int. J. Approx. Reason.* **102** 60–73
[24] Babenko M, Tchernykh A, Chervyakov N, Kuchukov V, Miranda-López V, Rivera-Rodriguez R, Du Z and Talbi E-G 2019 Positional Characteristics for Efficient Number Comparison over the Homomorphic Encryption *Program. Comput. Softw.* **45** 532–43
[25] Tchernykh A, Miranda-López V, Babenko M, Armenta-Cano F, Radchenko G, Drozdov A Y and Avetisyan A 2019 Performance evaluation of secret sharing schemes with data recovery in secured and reliable heterogeneous multi-cloud storage *Cluster Comput.* **22** 1173–85