ISSN 0361-7688, Programming and Computer Software, 2019, Vol. 45, No. 8, pp. 532–543. © Pleiades Publishing, Ltd., 2019.

Positional Characteristics for Efficient Number Comparison over the Homomorphic Encryption

M. Babenko[a,*], A. Tchernykh[b,c,d,], N. Chervyakov[a,***], V. Kuchukov[a,****], V. Miranda-López [b,*****], R. Rivera-Rodriguez[b,******], Z. Du[e,*******], and E.-G. Talbi[f,********]**

[a] *North-Caucasus Federal University, Stavropol, Russia*
[b] *CICESE Research Center, Ensenada, Mexico*
[c] *Institute for System Programming of the Russian Academy of Sciences, Moscow, Russia*
[d] *South Ural State University, Chelyabinsk, Russia*
[e] *Tsinghua University, Beijing, P. R. China*
[f] *Université de Lille, Villeneuve d'Ascq, France*
**e-mail: mgbabenko@ncfu.ru*
***e-mail: chernykh@cicese.mx*
****e-mail: ncherviakov@ncfu.ru*
*****e-mail: vkuchukov@ncfu.ru*
******e-mail: vmiranda@cicese.edu.mx*
*******e-mail: rrivera@cicese.mx*
********e-mail: duzh@tsinghua.edu.cn*
*********e-mail: el-ghazali.talbi@univ-lille.fr*

Received September 26, 2019; revised October 19, 2019; accepted October 30, 2019

Abstract—Modern algorithms for symmetric and asymmetric encryptions are not suitable to provide security of data that needs data processing. They cannot perform calculations over encrypted data without first decrypting it when risks are high. Residue Number System (RNS) as a homomorphic encryption allows ensuring the confidentiality of the stored information and performing calculations over encrypted data without preliminary decoding but with unacceptable time and resource consumption. An important operation for encrypted data processing is a number comparison. In RNS, it consists of two steps: the computation of the positional characteristic of the number in RNS representation and comparison of its positional characteristics in the positional number system. In this paper, we propose a new efficient method to compute the positional characteristic based on the approximate method. The approximate method as a tool to compare numbers does not require resource-consuming non-modular operations that are replaced by fast bit right shift operations and taking the least significant bits. We prove that in case when the dynamic range of RNS is an odd number, the size of the operands is reduced by the size of the module. If one of the RNS moduli is a power of two, then the size of the operands is less than the dynamic range. We simulate proposed method in the ISE Design Suite environment on the FPGA Xilinx Spartan-6 SP605 and show that it gains 31% in time and 37% in the area on average with respect to the known approximate method. It makes our method efficient for hardware implementation of cryptographic primitives constructed over a prime finite field.

DOI: 10.1134/S0361768819080115

1. INTRODUCTION

Distributed cloud systems for storing and processing big data are widely used in the concepts of smart city, smart health [1]. However, in the case of cloud computing, there is also the issue of their reliability and security. One of the methods of processing encrypted data is homomorphic encryption [2]. A Residue Number System (RNS) can be considered as a homomorphic cipher.

Non-positional number systems such as RNS do not require carries between digits during operations. However, for many applied problems, it is required to compare numbers or determine the sign of a number. Number comparison is a fundamental operation for implementation of cryptographic algorithms (Chang et al., 2015 [3], Chervyakov et al., 2017 [4], Sousa et al., 2016 [5]), digital signal processing (Chervyakov et al., 2014) [6], wireless communication systems (Ye et al., 2018) [7], cloud computing (Tchernykh et al., 2016 [8], Miranda-López et al., 2017 [9], Tchernykh et al., 2017 [10], Babenko et al., 2017 [11], Tchernykh et al., 2018 [27]), etc. Due to non-positional

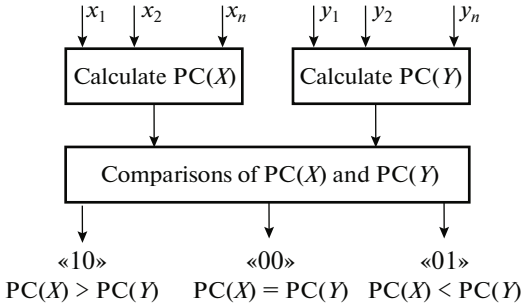

Fig. 1. Number comparison scheme in RNS.

nature of RNS, these operations are computationally complex.

In the positional number system, they are reduced to the simple bitwise comparison. In RNS, there are no simple number comparison algorithms (Szabo and Tanaka, 1969) [12]. The algorithm to compare numbers in RNS consists of two steps.

The first step is the computation of the Positional Characteristic (PC) of the number in residue form. The second step is the comparison of positional characteristics of numbers in the Positional Number System (PNS).

Figure 1 shows the general scheme of comparison of numbers $X\underline{RNS}(x_1, x_2, \ldots, x_n)$ and $Y\underline{RNS}(y_1, y_2, \ldots, y_n)$ in RNS.

To convert the number from RNS to the positional number system, we can apply the Chinese residue theorem (CRT), mixed-radix conversion (Bi and Gross, 2008) [13], recursive number pairing algorithm (nCRT) (Wang, 2000) [14], and their modifications.

In order to reduce the computational complexity of the number comparison in RNS, the researchers proposed the following functions as positional characteristics of the modular number: the diagonal function (Dimauro et al., 1993) [15], the core function (Burgess, 2003) [16], the factor function (Dimauro et al., 2003) [17], the monotonic function of Pirlo and Impedovo (Pirlo and Impedovo, 2013) [18], etc.

The most efficient function is based on the approximate method (Chervyakov et al., 2017) [19]. It allows to replace the division with the remainder by the operation of taking the most significant bits of the number.

In this paper, we optimize the approximate method for number comparison by reducing the number of divisions with the remainder and improving the accuracy of computations.

The article is organized as follows. In Section 2, we review the main aspects of RNS arithmetic. In Section 3, we consider number comparison algorithms based on RNS to PNS conversion. In Section 4, we provide an overview of the comparison methods based on the positional characteristics. In Section 5, we study the number comparison in RNS based on the algorithm for determining the sign of a number. In Section 6, we present a modification of the number comparison algorithm and its properties. In Section 7, we present a comparison of the proposed method with existing ones and provide simulation results. In the last section, the main conclusions are presented.

2. RNS AND ITS PROPERTIES

Residue from division of a number X by modulus p_i is a number x_i, which satisfies the equation $X = x_i + b \cdot p_i$ where b is an integer and $0 \le x_i < p_i$. Residue from division can be written in terms of congruence theory $x_i \equiv Xmodp_i$, or shorter $|X|_{p_i}$.

Residue number system is defined by a set of pairwise coprime numbers p_i, which is called RNS moduli set, i.e. $\{p_1, p_2, \ldots, p_n\}$, where $gcd(p_i, p_j) = 1$ for $i \ne j$, and n is a number of moduli.

Any number $X \in [0, P-1]$ can be uniquely represented in RNS as $X\underline{RNS}(x_1, x_2, \ldots, x_n)$, where $x_i \equiv Xmodp_i$ and $P = \prod_{i=1}^{n} p_i$ is RNS dynamic range.

An important criterion for efficient computations in RNS is the right selection of a moduli set. There are several approaches. For example, in (Patronik et al., 2014) [20], a moduli set of a special form $(2^n - 1, 2^n, 2^n + 1)$ is presented. In (Phatak & Houston, 2016) [21], all prime numbers from a given interval are used as a moduli set for computations in GPU.

Let $A\underline{RNS}(a_1, a_2, \ldots, a_n)$ and $B\underline{RNS}(b_1, b_2, \ldots, b_n)$ be two numbers. As shown in (Akuschskiy & Juditzkiy, 1968) [22], the following holds

$$C = A * B\underline{RNS}(|a_1 * b_1|_{p_1}, \ldots, |a_n * b_n|_{p_n}),$$

where is the operation from the set $\{+, -, \times\}$.

However, it is possible that the result of arithmetic operations is out of the range $C \notin [0, P-1]$. In this overflow case, the result is different from the desired by the value of the range. We illustrate it by the following example.

Let RNS be defined by moduli set {3, 5, 7} and two numbers be $A = 15\underline{RNS}(0,0,1)$ and $B = 20\underline{RNS}(2,0,6)$.

Now, we compute arithmetic operations:

$$A + B\underline{RNS}(|0+2|_3, |0+0|_5, |1+6|_7) = (2,0,0) = 35,$$

$$A - B\underline{RNS}(|0-2|_3, |0-0|_5, |1-6|_7) = (1,0,2) = 100 \ne -5,$$

$$A \cdot B\underline{RNS}(|0 \cdot 2|_3, |0 \cdot 0|_5, |1 \cdot 6|_7) = (0,0,6) = |300|_{105} = 90 \ne 300.$$

To verify that the result is correct, (Chervyakov et al., 2017) [4] developed an algorithm based on properties of the rank of a number. The advantage of the proposed approach is that it verifies the correctness without RNS to PNS conversion.

3. METHODS OF NUMBERS COMPARISON BASED ON RNS TO PNS CONVERSION

In most methods, the problem of comparing numbers is solved by converting numbers to PNS and comparing them.

3.1. Chinese Remainder Theorem

According to (Omondi and Premkumar, 2007) [23], RNS to PNS conversion is provided by Chinese remainder theorem, which can be written as:

$$X = \left| \sum_{i=1}^{n} P_i \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_P \quad (1)$$

where $P_i = \frac{P}{p_i}$, and $\left| P_i^{-1} \right|_{p_i}$ is the multiplicative inverse of $\left| P_i \right|_{p_i}$. We illustrate RNS to PNS conversion and number comparison by the following example.

Let RNS be defined by moduli set {3, 5, 7} and $X\underline{RNS}(2,2,3)$, $Y\underline{RNS}(1,3,4)$ numbers represented in RNS. Dynamic range of this RNS is $P = 3 \cdot 5 \cdot 7 = 105$.

We compute P_i:

$$P_1 = P/p_1 = 105/3 = 35,$$

$$P_2 = P/p_2 = 105/5 = 21,$$

$$P_3 = P/p_3 = 105/7 = 15.$$

In order to compute multiplicative inverse of P_i, we need to find x, which satisfies the congruence $x \cdot P_i \equiv 1 mod p_i$. Thus, $\left| P_1^{-1} \right|_3 = 2$, $\left| P_2^{-1} \right|_5 = 1$, $\left| P_3^{-1} \right|_7 = 1$. We verify that $\left| P_1^{-1} \right|_3 = 2$ by the expression $2 \cdot 35 \equiv 1 mod 3$. Thus, all necessary variables to use (1) are computed. We convert first number to RNS:

$$X = \left| 35 \cdot 2 \cdot 2 + 21 \cdot 2 \cdot 1 + 15 \cdot 3 \cdot 1 \right|_{105} =$$

$$\left| 227 \right|_{105} = 17.$$

and the second:

$$Y = \left| 35 \cdot 1 \cdot 2 + 21 \cdot 3 \cdot 1 + 15 \cdot 4 \cdot 1 \right|_{105} =$$

$$\left| 193 \right|_{105} = 88.$$

Since $17 < 88$, then $X < Y$.

The following is the algorithm of the number comparison based on (1).

Algorithm 1. Number comparison in RNS with Chinese remainder theorem from [23]

Input: $X\underline{RNS}(x_1,\ldots,x_n)$, $Y\underline{RNS}(y_1,\ldots,y_n)$, $P = \prod_{i=1}^{n} p_i$, $P_i = \frac{P}{p_i}$, $w_i = \left| P_i^{-1} \right|_{p_i}$ for each $i = \overline{1,n}$.

Output: $X > Y$-'10', $X < Y$-'01', $X = Y$ – '00'.

1. $S_X = 0$, $S_Y = 0$,
2. **For** $i = 1$ **to** n **do**:
2.1. $S_X = S_X + w_i \cdot P_i \cdot x_i$
2.2. $S_Y = S_Y + w_i \cdot P_i \cdot y_i$
3. $S_X = S_X mod P$
4. $S_Y = S_Y mod P$
5. **If** $S_X > S_Y$ **then return** '10'
6. **If** $S_X < S_Y$ **then return** '01'
7. **return** '00'

end.

Algorithm 1 has $4n$ multiplications, n additions, and 2 divisions by RNS range with remainder. Because of computational complexity of division by a big number P, researches proposed an alternative algorithm based on mixed radix conversion.

3.2. Mixed Radix Number System

Mixed Radix Number System (MRNS) allows number comparison without conversion to binary number system. A number in MRNS is defined by a tuple $[a_1, a_2, \ldots, a_n]$. Moduli of the system are p_1, p_1p_2, $p_1p_2p_3$, ..., $p_1p_2 \ldots p_{n-1}$, where $p_1, p_2, \ldots, p_n$ is the RNS moduli set. The following is the conversion of MRNS to binary representation:

$$X = a_1 + a_2p_1 + a_3p_1p_2 + \ldots + a_np_1 \ldots p_{n-1}$$

Since, MRNS is PNS, the number comparison in MRNS is equivalent to comparison of tuples $[a_1, a_2, \ldots, a_n]$ and $[b_1, b_2, \ldots, b_n]$.

To convert a number $X\underline{RNS}(x_1, x_2, \ldots, x_n)$ from RNS to MRNS $[a_1, a_2, \ldots, a_n]$ the following approach is used:

$$a_1 = x_1$$

$$a_2 = (x_2 - a_1) \cdot p_1^{-1}$$

$$\ldots$$

$$a_n = (x_n - a_1 - a_2p_1$$

$$-a_{n-1}p_1p_2 \ldots p_{n-2}) \cdot p_1^{-1} \cdot \ldots \cdot p_{n-1}^{-1}.$$

The following is an efficient implementation of the number comparison.

Algorithm 2. Number comparison in RNS with MRNS from [24]

Input: $X\underline{RNS}(x_1,\ldots,x_n)$, $Y\underline{RNS}(y_1,\ldots,y_n)$, $M_i = \prod_{j=1}^{i} p_j$, for each $i = \overline{1,n-2}$.

Output: $X > Y$-‘10’, $X < Y$-‘01’, $X = Y$–‘00’.

1. $a_1 = x_1$, $b_1 = y_1$,

2. $t_X = |x_2 - a_1|_{p_2}$, $t_Y = |y_2 - b_1|_{p_2}$,

3. $a_2 = |t_X / M_1|_{p_2}$, $b_2 = |t_Y / M_1|_{p_2}$,

4. $RX_{13} = |x_1|_{p_3}$, $RY_{13} = |y_1|_{p_3}$,

5. **For** $i = 3$ **to** n **do**:

5.1. **For** $j = i$ **to** n **do**:

5.1.1. $RX_{i-1,j} = |a_{i-1}|M_{i-2}|_{p_j} + RX_{i-2,j}|_{p_j}$

5.1.2. $RY_{i-1,j} = \left|b_{i-1}|M_{i-2}|_{p_j} + RY_{i-2,j}\right|_{p_j}$

5.2. $t_X = |x_i - RX_{i-1,i}|_{p_i}$,

5.3. $t_Y = |y_i - RY_{i-1,i}|_{p_i}$,

5.4. $a_i = |t_X / M_{i-1}|_{p_i}$, $b_i = |t_Y / M_{i-1}|_{p_i}$,

6. **For** $i = n$ **downto** 1 **do**:

6.1. **If** $a_i > b_i$ **then return** ‘10’

6.2. **Else If** $a_i < b_i$ **then return** ‘01’

7. **return** ‘00’

end.

Algorithm 2 has $n \cdot (n-1)$-modular multiplications and $2n$ modular subtractions. Thus, MRNS does not require division by a large number P, but leads to many modular operations.

3.3. Approximate Method

In order to avoid division by a large number, (Van, 1985) [25] proposes an approximate method based on transformation of [0, P) to [0, 2).

To do it, we rewrite (1) in the form

$$X = \sum_{i=1}^{n} P_i \cdot x_i \cdot \left|P_i^{-1}\right|_{p_i} - P \cdot r_x \qquad (2)$$

For a negative number, r_x is the rank of x. We divide (2) by $\frac{P}{2}$, and obtain

$$X_s = \left(\frac{2}{P}\right) \cdot X = \sum_{i=1}^{n} \frac{2}{p_i} \cdot x_i \cdot \left|P_i^{-1}\right|_{p_i} - 2r_x \qquad (3)$$

X_s can be computed as a sum of fractions minus a power of 2. This can be easily calculated since computations are done in binary number system.

We illustrate it with an example. For a number (2, 2, 3), according to (3), we have

$$X_s = \left|\frac{2}{3} \cdot 2 \cdot 2 + \frac{2}{5} \cdot 2 \cdot 1 + \frac{2}{7} \cdot 3 \cdot 1\right|_2 = \left|2\frac{34}{105}\right|_2 = \frac{34}{105}.$$

Note that this method works but the summands rarely can be represented as a finite fraction. Thus, each summand needs to be rounded.

If for each summand in (3) $N+1$ bits are allocated, $1-$ for the whole part and $N-$ for fractional and delete the rest bits then the error will satisfy the inequality $0 \le e_i < 2^{-N}$. And since there are n such summands, the error is upper bounded by $e = n2^{-t}$.

Since the numbers X_s are uniformly located in the interval, the distance between 2 neighbor numbers is $\frac{2}{P}$.

Moreover, the interval between the greatest positive number and 1 is $\frac{2}{P}$ for even P and $\frac{1}{P}$ for odd P. Thus, for a rounded X_s to correspond to the true value of X_s, error must satisfy the following inequalities:

$$n \cdot 2^{-N} \le \frac{2}{P}, \ \textit{for even } P, \qquad (4)$$

$$n \cdot 2^{-N} \le \frac{1}{P}, \ \textit{for odd } P, \qquad (5)$$

or

$$N \ge \lceil \log_2(n \cdot P) \rceil - 1 \text{ for even } P,$$

$$N \ge \lceil \log_2(n \cdot P) \rceil \text{ for odd } P.$$

Even though fractional representation takes up approximately $\lceil \log_2 n \rceil$ bits more memory, the simplicity compensates for this redundancy. This method can be implemented with Look-Up Tables (LUTs), which have as input the residue $|X|_{p_i}$, and output the rounded value and then the rounded values are summed modulo 2, which is easily implemented in hardware.

Let us consider a numeric example. RNS moduli set is {3, 5, 7} and two numbers are $1\underline{RNS}(1,1,1)$ and $104\underline{RNS}(2,4,6)$. For this RNS, $N \ge \lceil \log_2(105 \cdot 3) \rceil = 9$. The first number`s summands are:

$$\left|\frac{2}{3} \cdot 1 \cdot 2\right|_2 = \frac{4}{3} = 1.010101010101011\ldots \approx 1.010101011,$$

$$\left|\frac{2}{5} \cdot 1 \cdot 1\right|_2 = \frac{2}{5} = 0.011001100110011\ldots \approx 0.011001101,$$

$$\left|\frac{2}{7} \cdot 1 \cdot 1\right|_2 = \frac{2}{7} = 0.010010010010010\ldots \approx 0.010010010.$$

Now, we sum modulo 2 and obtain 0.000001010.

The second number`s summands are:

$$\left|\frac{2}{3} \cdot 2 \cdot 2\right|_2 = \frac{2}{3} = 0.101010101010101\ldots$$

$$\approx 0.101010101,$$

$$\left|\frac{2}{5}\cdot 1\cdot 4\right|_2 = \frac{8}{5} = 1.100110011001101\ldots$$

$$\approx 1.100110011,$$

$$\left|\frac{2}{7}\cdot 1\cdot 6\right|_2 = \frac{12}{7} = 1.101101101101110\ldots$$

$$\approx 1.101101110.$$

Now, we sum modulo 2 and obtain 1.111110110.

Comparing the obtained values, we conclude that $(1,1,1) < (2,4,6)$.

This method is more efficient than Chinese remainder theorem, but there is a question whether the accuracy is sufficient or redundant according to (4)–(5).

It is worth noting that this approach computes the positional characteristic using the following equation:

$$V(X) = \left|\sum_{i=1}^{n}\left\lceil \frac{2}{p_i}\|P_i^{-1}|_{p_i}\cdot\cdot x_i|_{p_i}\right\rceil_{2^{-N}}\right|_2, \qquad (6)$$

where $\lceil x\rceil_{2^{-N}} = \lceil 2^N x\rceil/2^N$.

The Algorithm 3 of the number comparison with approximate method is from work (Van, 1985) [25].

Algorithm 3. Number comparison in RNS with approximate method [25]

Input: $X\underline{RNS}(x_1,\ldots,x_n)$, $Y\underline{RNS}(y_1,\ldots,y_n)$, $(p_1, p_2, \ldots, p_n)$, $w_i = |P_i^{-1}|_{p_i}$ for each $i = \overline{1,n}$, where $N = \lceil \log_2(n\cdot P)\rceil$.

Output: $X > Y$-‘10’, $X < Y$-‘01’, $X = Y$ – ‘00’.

1. $S_X = 0$, $S_Y = 0$,
2. **For** $i = 1$ **to** n **do**:
2.1. $S_X = S_X + \lceil 2^{N+1}\cdot w_i\cdot x_i/p_i\rceil/2^N$
2.2. $S_Y = S_Y + \lceil 2^{N+1}\cdot w_i\cdot y_i/p_i\rceil/2^N$
3. $S_X = S_X mod 2$
4. $S_Y = S_Y mod 2$
5. **If** $S_X > S_Y$ **then return** '10'
6. **If** $S_X < S_Y$ **then return** '01'
7. **return** '00'

end.

To compare numbers with Algorithm 3, it is required to perform $2n$ multiplications, $2n$ divisions, $2n$ ceiling, and 2 divisions with remainder.

In order to reduce the number of operations in the approximate method, (Chervyakov et al., 2017) [19] proposed to use the following:

$$C(X) = \left|\sum_{i=1}^{n} W_i x_i\right|_1, \qquad (7)$$

where $W_i = \left\lfloor \frac{2^N|P_i^{-1}|_{p_i}}{p_i}\right\rfloor/2^N$, $|x|_1$ – is the fractional part of x, $N = \lceil \log_2(\rho\cdot P)\rceil$ and $\rho = -n + \sum_{i=1}^{n} p_i$.

Algorithm 4. Number comparison in RNS with approximate method [19]

Input: $X\underline{RNS}(x_1,\ldots,x_n)$, $Y\underline{RNS}(y_1,\ldots,y_n)$, $(p_1, p_2, \ldots, p_n)$, $W_i = \left\lfloor \frac{2^N|P_i^{-1}|_{p_i}}{p_i}\right\rfloor/2^N$ for each $i = \overline{1,n}$, where $N = \lceil \log_2(\rho\cdot P)\rceil$.

Output: $X > Y$-‘10’, $X < Y$-‘01’, $X = Y$ – ‘00’.

1. $S_X = 0$, $S_Y = 0$,
2. **For** $i = 1$ **to** n **do**:
2.1. $S_X = (S_X + W_i\cdot x_i) mod 1$
2.2. $S_Y = (S_Y + W_i\cdot y_i) mod 1$
3. **If** $S_X > S_Y$ **then return** '10'
4. **If** $S_X < S_Y$ **then return** '01'
5. **return** '00'

end.

Algorithm 4 has $2n$ multiplications and n additions. Its advantage over Algorithm 3 is that it does not require ceiling, but the size of operands is increased.

4. METHODS OF NUMBER COMPARISON IN RNS WITH PC

To reduce the computational complexity of the comparison, Dimauro et al., 1993 [15] proposed to use monotonic diagonal function.

4.1. Diagonal Function

The method based on the special diagonal function, which is defined as the sum of the corresponding coefficients P_i, called the Sum of Quotients Technique (SQT) method [15]. The diagonal function is a monotonically increasing function on the basis of which it is possible to compare numbers.

The diagonal function has the form:

$$D(X) = \left\lfloor \frac{X}{p_1}\right\rfloor + \left\lfloor \frac{X}{p_2}\right\rfloor + \ldots + \left\lfloor \frac{X}{p_n}\right\rfloor. \qquad (8)$$

However, (8) is not used in practice. Therefore, (Dimauro et al., 1993) [15] proposed another expression to compute diagonal function:

$$D(X) = \left|\sum_{i=1}^{n} k_i\cdot x_i\right|_{SQ} \qquad (9)$$

where $k_i = |-p_i^{-1}|_{SQ}$, where $i = \overline{1,n}$, $SQ = P_1 + P_2 + \ldots + P_n$.

Since diagonal function (8) increases monotonically, it can be used to compare numbers, i.e. if $D(X) < D(Y)$, then $X < Y$. However, there are cases when $D(X) = D(Y)$, and then $X < Y$, if $x_i < y_i$, $i = \overline{1,n}$.

Algorithm 5. Number comparison in RNS with diagonal function [15]

Input: $X\underline{RNS}(x_1,\ldots,x_n)$, $Y\underline{RNS}(y_1,\ldots,y_n)$, $SQ = \sum_{i=1}^{n} P_i$, $k_i = \left|-p_i^{-1}\right|_{SQ}$ for each $i = \overline{1,n}$.

Output: $X > Y$-'10', $X < Y$-'01', $X = Y$ – '00'.

1. $S_X = 0$, $S_Y = 0$,
2. **For** $i = 1$ **to** n **do**:
2.1. $S_X = S_X + k_i \cdot x_i$
2.2. $S_Y = S_Y + k_i \cdot y_i$
3. $S_X = S_X mod SQ$
4. $S_Y = S_Y mod SQ$
5. **If** $S_X > S_Y$ **then return** '10'
6. **If** $S_X < S_Y$ **then return** '01'
7. **If** $x_1 > y_1$ **then return** '10'
8. **If** $x_1 < y_1$ **then return** '01'
9. **return** '00'

end.

We illustrate it with an example. We take numbers $X\underline{RNS}(2,2,3)$ and $Y\underline{RNS}(1,3,4)$. First, we compute

$$SQ = 35 + 21 + 15 = 71,$$
$$k_1 = \left|-p_1^{-1}\right|_{71} = \left|-3^{-1}\right|_{71} = 47,$$
$$k_2 = \left|-p_2^{-1}\right|_{71} = \left|-5^{-1}\right|_{71} = 14,$$
$$k_3 = \left|-p_3^{-1}\right|_{71} = \left|-7^{-1}\right|_{71} = 10.$$

Then, we compute diagonal function:

$$D(X) = \left|2 \cdot 47 + 2 \cdot 14 + 3 \cdot 10\right|_{71} = 10,$$
$$D(Y) = \left|1 \cdot 47 + 2 \cdot 14 + 3 \cdot 10\right|_{71} = 34.$$

Since $D(X) < D(Y)$, then X < Y.

Algorithm 5 has $2n$ multiplications, n additions, and 2 divisions by a large number SQ with remainder.

4.2. Diagonal Function

(Pirlo and Impedovo, 2013) [18] proposed to use minimal Akushsky core function without critical core. This approach is similar to the method of the diagonal function. The Pirlo is:

$$Pi(X) = \left\lfloor \frac{X}{p_n} \right\rfloor \tag{10}$$

However, (10) is not used in practice and another function was proposed to computer Pirlo function:

$$Pi(X) = \left| \sum_{i=1}^{n} k_i x_i \right|_{Pi(P)} \tag{11}$$

where $k_i = \left\lfloor \frac{|P_i^{-1}|_{p_i} P_i}{p_n} \right\rfloor$.

Since Pirlo function (10) is monotonic and increasing, it can be used to compare numbers, i.e. if $Pi(X) < Pi(Y)$, then $X < Y$. However, there are cases when $Pi(X) = Pi(Y)$, and then $X < Y$, if $x_n < y_n$.

We illustrate this with an example. We use $X\underline{RNS}(2,2,3)$ and $Y\underline{RNS}(1,3,4)$. First, we compute:

$$P_3 = 15,$$
$$k_1 = \left\lfloor \frac{|P_1^{-1}|_{p_1} P_1}{p_3} \right\rfloor = \left\lfloor \frac{2 \cdot 35}{7} \right\rfloor = 10,$$
$$k_2 = \left\lfloor \frac{|P_2^{-1}|_{p_2} P_2}{p_3} \right\rfloor = \left\lfloor \frac{1 \cdot 21}{7} \right\rfloor = 3$$
$$k_3 = \left\lfloor \frac{|P_3^{-1}|_{p_3} P_3}{p_3} \right\rfloor = \left\lfloor \frac{1 \cdot 15}{7} \right\rfloor = 2$$

We compute Pirlo function:

$$Pi(X) = \left|2 \cdot 10 + 2 \cdot 3 + 3 \cdot 2\right|_{15} = 2,$$
$$Pi(Y) = \left|1 \cdot 10 + 3 \cdot 3 + 4 \cdot 2\right|_{15} = 12.$$

and since $Pi(X) < Pi(Y)$, then $X < Y$.

As shown in (Mohan, 2016) [26], Pirlo function is less efficient than CRT since it requires additional comparison.

5. NUMBER COMPARISON BASED ON DETERMINING THE SIGN OF A NUMBER

In order to optimize the algorithm for number comparison, sometimes it is advisable to use the algorithm for determining the sign of a number in the second step instead of the comparison.

Some applications in RNS use negative numbers. To determine the sign of a number in RNS with negative numbers, it is necessary to compare this number with the middle point of the range. It should also be noted that, in this case, negative numbers follow positive ones, and to compare numbers, you first need to determine their sign.

In RNS with moduli set $\{p_1, p_2, \ldots, p_n\}$ and dynamic range $P = \prod_{i=1}^{n} p_i$, a number X can be represented if it satisfies the following inequalities:

$$\frac{-P-1}{2} \le X \le \frac{P-1}{2}, \text{ if } P \text{ is odd},$$

$$\frac{-P}{2} \le X \le \frac{P}{2} - 1, \text{ if } P \text{ is even.}$$

Then according to (Omondi & Premkumar, 2007) [21], if $X\underline{RNS}(x_1, x_2, \ldots, x_n)$, then a negative number would be $-X\underline{RNS}(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n)$, where $\overline{x}_i$ is inverse of x_i modulo m_i. For example, for RNS {3, 5, 7} and number $X = 17\underline{RNS}(2,2,3)$ we obtain $-X = (3-2, 5-2, 7-3)\underline{RNS}(1, 3, 4)$. It is obvious that to obtain a negative number it is required to subtract the value of dynamic range, i.e. $(1,3,4) = 88 = 88 - 105 = -17$. If we look at the whole range, we can see that numbers are allocated as follows: $\underline{0,1,\ldots,52,}\overline{-52,-51,\ldots,-1}$.

Now, when we defined negative numbers, there is need to determine the sign of a number. There are several approaches to determining the sign of numbers in RNS: conversion to PNS with CRT, MRNS and others.

The problem with CRT is the need to find the remainder from division by a large modulus P, which is a resource consuming operation, and the subsequent comparison with the constant.

We introduce the sign function $S(X)$ for RNS with an even dynamic range P:

$$S(X) = \begin{cases} 0, & \text{if} \quad 0 \le X < \dfrac{P}{2}, \\ 1, & \text{if} \quad \dfrac{P}{2} \le X < P. \end{cases} \tag{12}$$

Approximate method described in Section 3.3 simplifies (12):

$$S(X) = \begin{cases} 0, & \text{if} \quad 0 \le X_s < 1, \\ 1, & \text{if} \quad 1 \le X_s < 2. \end{cases} \tag{13}$$

It should be noted that for correct operation of the algorithm for number comparing with determining the sign of a number, doubling the range of RNS is required, which leads to additional computational loads during data processing.

6. MODIFICATION OF THE NUMBER COMPARISON ALGORITHM IN RNS

We take the following function as a positional characteristic:

$$f(X) = \left| \sum_{i=1}^{n} \overline{k}_i x_i \right|_{2^N}$$

where $\overline{k}_i = \left\lfloor \dfrac{2^N |P^{-1}|_{p_i}}{p_i} \right\rfloor$.

6.1. Number Comparison in RNS with Odd Range

Lemma 1. If $N = \lceil \log_2(-n + n \cdot P) \rceil$, then the following holds:

$$\left\lfloor \frac{\sum_{i=1}^{n} \overline{k}_i x_i}{2^N} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^{n} k_i x_i}{P} \right\rfloor, \tag{14}$$

where $\overline{k}_i = \left\lfloor \dfrac{2^N |P^{-1}|_{p_i}}{p_i} \right\rfloor$ and $k_i = \left| P^{-1} \right|_{p_i} P_i$.

Proof.

Since k_i and $\overline{k}_i$ satisfy the equation $\overline{k}_i = \dfrac{2^N k_i}{P} - \dfrac{|2^N k_i|_P}{P}$, the the expression $\left\lfloor \dfrac{\sum_{i=1}^{n} \overline{k}_i x_i}{2^N} \right\rfloor$ can be rewritten as:

$$\left\lfloor \frac{\sum_{i=1}^{n} \overline{k}_i x_i}{2^N} \right\rfloor = \left\lfloor \frac{1}{P} \sum_{i=1}^{n} k_i x_i - \frac{1}{P \cdot 2^N} \sum_{i-1}^{n} |2^N k_i|_P \cdot x_i \right\rfloor \tag{15}$$

We use $\dfrac{1}{P}\sum_{i=1}^{n} k_i x_i = \left\lfloor \dfrac{1}{P}\sum_{i=1}^{n} k_i x_i \right\rfloor + \dfrac{X}{P}$ in (15), and obtain:

$$\left\lfloor \frac{\sum_{i=1}^{n} \overline{k}_i x_i}{2^N} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^{n} k_i x_i}{2^N} \right\rfloor + \left\lfloor \frac{X}{P} - \frac{1}{P \cdot 2^N} \cdot \sum_{i=1}^{n} |2^N k_i|_P x_i \right\rfloor \tag{16}$$

From (16), it follows that the condition of Lemma 1 is equivalent to:

$$0 \le \frac{X}{P} - \frac{1}{P \cdot 2^N} \cdot \sum_{i=1}^{n} |2^N k_i|_P x_i < 1 \tag{17}$$

According to CRT, X satisfies the inequality $0 \le X < P$, hence, $0 \le \dfrac{X}{P} < 1$. Considering that $\dfrac{1}{P \cdot 2^N} \cdot \sum_{i=1}^{n} |2^N k_i|_P x_i \ge 0$, we obtain that the right-hand side of (17) is true for all N.

We now consider the left-hand side of (17). For $X = 0$, it is true for all N. Let X satisfies $1 \le X < P$, then left hand side of (17) can be rewritten as:

Table 1. Properties of number comparison methods

Method		$\lceil\cdot\rceil$	Number of operations '×'	Size of the modulus	Form of the modulus
CRT [23]			$2n$	$\lfloor n \cdot \log_2 p_n \rfloor$	P
Diagonal function [15]			$2n$	$\lfloor (n-1) \cdot \log_2 p_n + \log_2 n \rfloor$	SQ
MRNS [24]			$n \cdot (n-1)/2$	$\lceil \log_2 p_n \rceil$	p_i
Approximate method [25]	$\lvert P \rvert_2 = 1$	n	$2n$	$\lceil \log_2 (n \cdot P) \rceil$	2^N
Approximate method (AM) [19]			$2n$	$\lceil \log_2 (\rho \cdot P) \rceil$	2^N
Proposed method			$2n$	$\lceil \log_2 (n \cdot P - n) \rceil$	2^N
Approximate method [25]	$\lvert P \rvert_2 = 0$	n	$2n$	$\lceil \log_2 (n \cdot P) \rceil - 1$	2^N
Approximate method (AM) [19]			$2n$	$\lceil \log_2 (\rho \cdot P) \rceil - 1$	2^N
Proposed method			$4n$	$\lceil \log_2 (-n + n \cdot P_n) \rceil$	2^N

Table 2. Sample sets of moduli and their characteristics

No.	Moduli set	Size of moduli (bits)	Size of dynamic range (bits)
1	21, 23, 25, 29	5	18
2	263, 269, 271, 277	8	32
3	4099, 4111, 4127, 4129	12	48
4	65537, 65539, 65543, 65551	16	64
5	1048583, 1048589, 1048601, 1048609	20	80
6	16777259, 16777289, 16777291, 16777331	24	96

$$2^N \geq \frac{1}{X}\sum_{i=1}^{n} \left|2^N k_i\right|_P x_i \tag{18}$$

Since $|2^N k_i|_P \leq P-1$, then $\sum_{i=1}^{n} |2^N k_i|_P x_i \leq (P-1)\sum_{i=1}^{n} x_i$, hence, for all $1 \leq X < P$ the following holds:

$$\frac{1}{X}\sum_{i=1}^{n} |2^N k_i|_P x_i \leq n \cdot (P-1) \tag{19}$$

From (18) and (19), it follows that if $N = \lceil \log_2(-n + n \cdot P) \rceil$, then left hand side of (17) is true, hence, (14) holds. Lemma is proved.

Theorem 1. If $N = \lceil \log_2 (-n + n \cdot P) \rceil$, then function $f(X)$ is increasing.

Proof.

For $f(X)$ to be increasing it is necessary and sufficient that for all integers $1 \leq X \leq P-1$ the following holds:

$$f(X) - f(X-1) > 0 \tag{20}$$

Table 3. Number of slices for 4-moduli sets of RNS from Table 2

Size of dynamic range (bits)	Number of slices				Reduction, %		
	1	2	3	4			
	CRT	MRNS	AM	New	(1-4)/4	(2-4)/4	(3-4)/4
18	138	138	138	96	43.7	43.7	43.7
32	467	381	307	212	120.2	79.7	44.8
48	896	755	511	367	144.1	105.7	**39.2**
64	1457	1017	751	515	182.9	97.4	45.8
80	2165	1602	1102	751	188.2	113.3	46.7
96	3160	2434	2031	1433	120.5	69.8	41.7

Since $|X|_{2^N} = X - \left\lfloor \frac{X}{2^N} \right\rfloor \cdot 2^N$, then $f(X)$ can be rewritten as:

$$f(X) = \sum_{i=1}^{n} \overline{k_i} x_i - \left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} x_i}{2^N} \right\rfloor \cdot 2^N \tag{21}$$

Since $\sum_{i=1}^{n} \overline{k_i}(x_i - |x_i - 1|_{p_i}) = \sum_{i=1}^{n} \overline{k_i} - \sum_{x_i=0} \overline{k_i} p_i$, we obtain

$$f(X) - f(X-1) = \sum_{i=1}^{n} \overline{k_i} - \sum_{x_i=0} \overline{k_i} p_i - \left(\left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} x_i}{2^N} \right\rfloor - \left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} |x_i - 1|_{p_i}}{2^N} \right\rfloor \right) \cdot 2^N \tag{22}$$

Since condition of Lemma 1 is satisfied,

$$\left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} x_i}{2^N} \right\rfloor - \left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i} |x_i - 1|_{p_i}}{2^N} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^{n} k_i x_i}{P} \right\rfloor - \left\lfloor \frac{\sum_{i=1}^{n} k_i |x_i - 1|_{p_i}}{P} \right\rfloor \tag{23}$$

Using theorem from (Chervyakov, et. al., 2017) [4] and Lemma 1, Eq. (23) becomes:

$$\left\lfloor \frac{\sum_{i=1}^{n} k_i x_i}{P} \right\rfloor - \left\lfloor \frac{\sum_{i=1}^{n} k_i |x_i - 1|_{p_i}}{P} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^{n} k_i}{P} \right\rfloor - \sum_{x_i=0} |P_i^{-1}|_{p_i} = \left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i}}{P} \right\rfloor - \sum_{x_i=0} |P_i^{-1}|_{p_i} \tag{24}$$

We use (24) in (22), obtain:

$$f(X) - f(X-1) = \sum_{i=1}^{n} \overline{k_i} - \sum_{x_i=0} \overline{k_i} p_i - \left(\left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i}}{2^N} \right\rfloor - \sum_{x_i=0} |P_i^{-1}|_{p_i} \right) \cdot 2^N \tag{25}$$

Since $\sum_{i=1}^{n} \overline{k_i} - \left\lfloor \frac{\sum_{i=1}^{n} \overline{k_i}}{2^N} \right\rfloor \cdot 2^N = \left| \sum_{i=1}^{n} \overline{k_i} \right|_{2^N}$ and $|P_i^{-1}|_{p_i} \cdot 2^N - \overline{k_i} p_i = \frac{|2^N k_i|_P}{P_i}$ then for each $i = \overline{1,n}$ (25) becomes:

$$f(X) - f(X-1) = \left| \sum_{i=1}^{n} \overline{k_i} \right|_{2^N} + \sum_{x_i=0} \frac{|2^N k_i|_P}{P_i} \tag{26}$$

Since $\left| \sum_{i=1}^{n} \overline{k_i} \right|_{2^N} > 0$, then from (26) it follows that $f(X) - f(X-1) > 0$ hence, $f(X)$ is increasing. Theorem is proven.

From Theorem 1, it follows that the introduced function is monotonic. Hence, it can be used as a positional characteristic for comparing numbers in RNS. We propose an algorithm for comparing numbers based on Theorem 1.

Algorithm 5. Number comparison in RNS with odd range with $f(X)$

Input: $X \underline{RNS}(x_1,\ldots,x_n)$, $Y \underline{RNS}(y_1,\ldots,y_n)$, $\overline{k_i} = \left\lfloor 2^N \cdot |P_i^{-1}|_{p_i} / p_i \right\rfloor$, for each $i = \overline{1,n-1}$, where $N = \lceil \log_2(-n + n \cdot P) \rceil$.

Output: $X > Y$-‘10’, $X < Y$-‘01’, $X = Y$ – ‘00’.

1. $f_X = 0$; $f_Y = 0$;
2. **For** $i := 1$ **to** n **do**:
2.1. $f_X = |f_X + \overline{k_i} \cdot x_i|_{2^N}$;
2.2. $f_Y = |f_Y + \overline{k_i} \cdot y_i|_{2^N}$;
3. **IF** $f_X > f_Y$ **then return** ‘10’
4. **IF** $f_X < f_Y$ **then return** ‘01’
5. **return** «00»

end.

Algorithm 5 has $2n$ multiplications and n additions.

The proposed approach reduces the computational complexity of the algorithm for comparing numbers in RNS. Efficient implementation of the operation $|x \cdot y|_{2^N}$ on FPGA reduces the logic scheme for hardware implementation in comparison with the classical multiplication of two numbers $x \cdot y$.

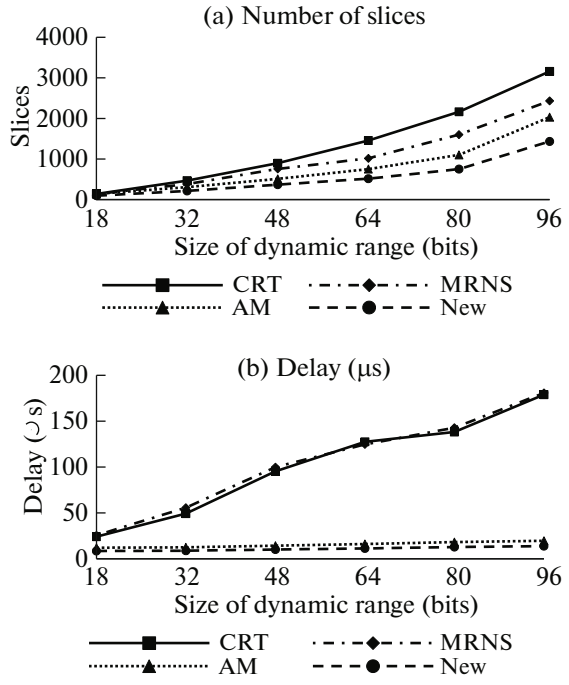

Fig. 2. Complexity characteristics of RNS comparators for 4-moduli sets RNS from Table 2.

6.2. Number Comparison in RNS, if one of the Moduli is Equal to the Power of Two

Since RNS modules are pairwise coprime numbers, there is only one even module. Hence, without loss of generality, we assume that the n-th module has the following form $p_n = 2^t$. Then using the property of RNS, the numbers X, Y can be represented in the following form:

$$X = A \cdot 2^t + x_n, \quad Y = B \cdot 2^t + y_n. \tag{27}$$

To compare numbers defined by (27), we use Algorithm 6.

Since n-th RNS modulus is even, $p_1, p_2, \ldots, p_{n-1}$ are odd and P_n is odd. Coefficients A and B satisfy: $0 \le A < P_n$ and $0 \le B < P_n$. With remaiders of A and B modulo $p_1, p_2, \ldots, p_{n-1}$, we can compare them with Algorithm 5.

Algorithm 6. Comparison of X, Y of the form: $X = A \cdot p_n + x_n$, $Y = B \cdot p_n + y_n$.

Input: $X = (A, x_n)$ and $Y = (B, y_n)$.

Output: $X > Y$-‘10’, $X < Y$-‘01’, $X = Y$ – ‘00’.

1. **IF** $A > B$ **then return** ‘10’;
2. **IF** $A < B$ **then return** ‘01’;
3. **IF** $x_n > y_n$ **then return** ‘10’;
4. **IF** $x_n < y_n$ **then return** ‘01’;
5. **return** ‘00’.

end.

Algorithm 7. Algorithm to compare X and Y.

Input: $X \underline{RNS}(x_1, \ldots, x_n)$, $Y \underline{RNS}(y_1, \ldots, y_n)$, $p_1, p_2, \ldots, p_{n-1}, p_n$, $I_i = \left|\frac{1}{p_n}\right|_{p_i}$ for each $i = \overline{1, n-1}\,\overline{k_i} = \left\lfloor 2^N \cdot \left|1/P_i\right|_{p_i} / p_i \right\rfloor$ for each $i = \overline{1, n-1}$, where $N = \lceil \log_2(-n + n \cdot P_n) \rceil$ and $P_i = P_n / p_i$ for each $i = \overline{1, n-1}$.

Output: $X > Y$-‘10’, $X < Y$-‘01’, $X = Y$ – ‘00’.

1. **For** $i := 1$ **to** $n-1$ **do**:

1.1. $a_i := \left|x_i - x_n\right|_{p_i}$; \\Parallel processing

1.2. $b_i := \left|y_i - y_n\right|_{p_i}$; \\Parallel processing

2. **For** $i := 1$ **to** $n-1$ **do**:

2.1. $a_i := \left|a_i \cdot I_i\right|_{p_i}$; \\ Parallel processing

2.2. $b_i := \left|b_i \cdot I_i\right|_{p_i}$; \\Parallel processing

3. $S_A = 0$; $S_B = 0$;

4. **For** $i := 1$ **to** $n-1$ **do**:

4.1. $S_A = \left|S_A + \overline{k_i} \cdot a_i\right|_{2^N}$;

4.2. $S_B = \left|S_B + \overline{k_i} \cdot b_i\right|_{2^N}$;

5. **IF** $S_A > S_B$ **then return** ‘10’

6. **IF** $S_A < S_B$ **then return** ‘01’

7. **IF** $a_n > b_n$ **then return** ‘10’

8. **IF** $a_n < b_n$ **then return** ‘01’

9. **return** ‘01’

end.

The Algorithm 7 compares X and Y.

It has $4n$ multiplications, $2n$ subtractions, and n additions.

Table 1 shows the properties of different methods to compute positional characteristics. The proposed method has reduced size of the moduli.

7. SIMULATION

Simulation of the number comparison methods was conducted in the ISE Design Suite environment on the FPGA Xilinx Spartan-6 SP605. Two numbers X and Y represented in RNS were used as the input. The output is one of the three values: “00” – the numbers are equal, “01” – $X < Y$, and “10” – $X > Y$.

Algorithms based on the Chinese remainder theorem, approximate method, MRNS, and proposed method were compared with five RNS parameters (Table 2).

Hardware resources on an FPGA are characterized by the number of slices that are comprised of LUTs and flip flops. Another important characteristic is the delay or depth of the obtained circuit. For each pair of numbers, we consider all paths from the primary input to the exit. The corresponding delays are proportional to the number of LUTs on the path. The delay of the

Table 4. Delay (μs) for 4-moduli sets of RNS from Table 2

Size of dynamic range (bits)	Delay, μs				Reduction, %		
	1	2	3	4			
	CRT	MRNS	AM	New	(1-4)/4	(2-4)/4	(3-4)/4
18	24.1	25.6	12.1	8.5	183.5	201.1	42.3
32	49.3	55.7	12.5	8.8	460.2	532.9	42.0
48	95.4	100	14.3	10.1	844.5	890.1	**41.5**
64	127.6	124.7	16.1	11.3	1029.2	1003.5	42.4
80	138.3	142.9	18.2	12.8	980.4	1016.4	42.1
96	178.9	180.8	19.6	13.8	1196.3	1210.1	42.0

solutions is defined as the maximum of all delays measured for input numbers from the dynamic range. The criteria for comparison of different number comparators are the number of slices and delay.

From the simulation results presented in Fig. 2, we can conclude that the proposed method shows better characteristics gaining more than 39.2% in slices (Table 3, Size of dynamic range = 48 bits), and 41.5% in delay (Table 4, Size of dynamic range = 48 bits).

8. CONCLUSION

Homomorphic encryption potentially supports computation on encrypted data and ensures data confidentiality simultaneously. However, fully homomorphic encryptions schemes suffer from performances problems. In this paper, we present new efficient method of the number comparison in RNS and provide its experimental analysis in the ISE Design Suite environment on FPGA Xilinx Spartan-6 SP605. It shows that the proposed approximate method is fastest among state-of-the-art ones. The worst result is shown by the algorithm based on the MRNS. The diagonal function requires finding the remainder from division by a smaller modulus, but in the case of equality of the diagonal functions, an additional refinement is required, which takes additional time.

Performance analysis shows that the new circuits have delay reduction ranging from over 41.5% to over 42.4%, compared to the fastest circuits designed using existing methods. Moreover, it is implemented with less slices, whose reduction can attain over 39.2%. Therefore, the presented method leads to designing the most efficient hardware comparators of numbers using a general RNS moduli set. It is important to increase efficiency of processing of encrypted data without decrypting it first when the data becomes vulnerable to unauthorized access and theft.

Future research will include extensions of the proposed approach to handle other difficult non-modular RNS operations like sign and overflow detection.

ACKNOWLEDGMENTS

The work is partially supported by Russian Foundation for Basic Research (RFBR) 18-07-00109, 18-07-01224, and 19-07-00856, State task nos. 2.6035.2017 and 2019-1105, Russian Federation President Grant MK-341.2019.9, and SP-2236.2018.5.

REFERENCES

1. Massobrio, R., Nesmachnow, S., Tchernykh, A., Avetisyan, A., and Radchenko, G., Towards a cloud computing paradigm for big data analysis in smart cities, *Program. Comput. Software*, 2018, vol. **44**, no. 3, pp. 181–189.
2. Varnovskiy, N.P., Martishin, S.A., Khrapchenko, M.V., and Shokurov, A.V., Secure cloud computing based on threshold homomorphic encryption, *Program. Comput. Software*, 2015, vol. **41**, no. 4, pp. 215–218.
3. Chang, C.H., Molahosseini, A.S., Zarandi, A.A.E., and Tay, T.F., Residue number systems: a new paradigm to datapath optimization for low-power and high-performance digital signal processing applications, *IEEE Circuits Syst. Mag.*, 2015, vol. **15**, no. 4, pp. 26–44.
4. Chervyakov, N., Babenko, M., Tchernykh, A., Kucherov, N., Miranda-López, V., and Cortés-Mendoza, J.M., AR-RRNS: configurable reliable distributed data storage systems for Internet of things to ensure security, *Future Gener. Comput. Syst.*, 2019, vol. **92**, pp. 1080–1092. https://doi.org/10.1016/j.future.2017.09.061
5. Sousa, L., Antao, S., and Martins, P., Combining residue arithmetic to design efficient cryptographic circuits and systems, *IEEE Circuits Syst. Mag.*, 2016, vol. **16**, no. 4, pp. 6–32.
6. Chervyakov, N.I., Lyakhov, P.A., and Babenko, M., Digital filtering of images in a residue number system using finite-field wavelets, *Autom. Control Comput. Sci.*, 2014, vol. **48**, no. 3, pp. 180–189.
7. Ye, R., Boukerche, A., Wang, H., Zhou, X., and Yan, B., RESIDENT: a reliable residue number system-based data transmission mechanism for wireless sensor networks, *Wireless Networks*, 2018, vol. **24**, no. 2, pp. 597–610.

8. Tchernykh, A., Schwiegelsohn, U., Talbi, E.G., and Babenko, M., Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability, *J. Comput. Sci.*, 2019, vol. **36**, p. 100581.
9. Miranda-López, V., Tchernykh, A., Cortés-Mendoza, J.M., Babenko, M., G. Radchenko, Nesmachnow, S., and Du, Z., Experimental analysis of secret sharing schemes for cloud storage based on RNS, *Proc. Latin American High Performance Computing Conf.*, Buenos Aires, 2017, pp. 370–383.
10. Tchernykh, A., Babenko, M., Chervyakov, N., Cortés-Mendoza, J.M., Kucherov, N., Miranda-López, V., Deryabin, M., Dvoryaninova, I., and Radchenko, G., Towards mitigating uncertainty of data security breaches and collusion in cloud computing, *Proc. 28th Int. Workshop on Database and Expert Systems Applications (DEXA)*, Lyon, 2017, pp. 137–141.
11. Babenko, M., Chervyakov, N., Tchernykh, A., Kucherov, N., Shabalina, M., Vashchenko, I., Radchenko, G., and Murga, D., Unfairness correction in P2P grids based on residue number system of a special form, *Proc. 28th Int. Workshop on Database and Expert Systems Applications (DEXA)*, Lyon, 2017, pp. 147–151.
12. Szabo, N.S. and Tanaka, R.I., *Residue Arithmetic and Its Applications to Computer Technology*, New York: McGraw-Hill, 1967.
13. Bi, S. and Gross, W.J., The mixed-radix Chinese remainder theorem and its applications to residue comparison, *IEEE Trans. Comput.*, 2008, vol. 57, no. 12), 1624–1632.
14. Wang, Y., Residue-to-binary converters based on new Chinese remainder theorems, *IEEE Trans. Circuits Syst.*, 2000, vol. **47**, no. 3, pp. 197–205.
15. Dimauro, G., Impedovo, S., and Pirlo, G., A new technique for fast number comparison in the residue number system, *IEEE Trans. Comput.*, 1993, vol. **42**, no. 5, pp. 608–612.
16. Burgess, N., Scaling an RNS number using the core function, *Proc. 16th IEEE Symp. on Computer Arithmetic*, Santiago de Compostela, 2003, pp. 262–269.
17. Dimauro, G., Impedovo, S., Modugno, R., Pirlo, G., and Stefanelli, R., Residue-to-binary conversion by the "quotient function", *IEEE Trans. Circuits Syst.*, 2003, vol. **50**, no. 8, pp. 488–493.
18. Pirlo, G. and Impedovo, D., A new class of monotone functions of the residue number system, *Int. J. Math. Models Methods Appl. Sci.*, 2013, vol. **7**, no. 9, pp. 803–809.
19. Chervyakov, N.I., Molahosseini, A.S., Lyakhov, P.A., Babenko, M.G., and Deryabin, M.A., Residue-to-binary conversion for general moduli sets based on approximate Chinese remainder theorem, *Int. J. Comput. Math.*, 2017, vol. **94**, no. 9, pp. 1833–1849.
20. Patronik, P. and Piestrak, S.J., Design of reverse converters for general RNS moduli sets $\{2^k, 2^n - 1, 2^n + 1, 2^n + 1 - 1\}$ and $\{2^k, 2^n - 1, 2^n + 1, 2^n - 1 - 1\}$ (n even), *IEEE Trans. Circuits Syst.*, 2014, vol. **61**, no. 6, pp. 1687–1700.
21. Phatak, D.S. and Houston, S.D., New distributed algorithms for fast sign detection in residue number systems (RNS), *J. Parallel Distrib. Comput.*, 2016, vol. **97**, pp. 78–95.
22. Akushskii, I.Ya. and Yuditskii, D.I., *Mashinnaya arifmetika v ostatochnykh protsessakh* (Machine Arithmetic in Residual Classes), Moscow: Sovetskoe Radio, 1968.
23. Omondi, A.R. and Premkumar, B., *Residue Number Systems: Theory and Implementation*, London: Imperial College Press, 2007.
24. Isupov, K., "An algorithm for magnitude comparison in RNS based on mixed-radix conversion II, *Int. J. Comput. Appl.*, 2016, vol. **141**, no. 5.
25. Van Vu, T., Efficient implementations of the Chinese remainder theorem for sign detection and residue decoding, *IEEE Trans. Comput.*, 1985, vol. **100**, no. 7, pp. 646–651.
26. Mohan, P.A., RNS to binary conversion using diagonal function and Pirlo and Impedovo monotonic function, *Circuits, Syst., Signal Process.*, 2016, vol. **35**, no. 3, pp. 1063–1076.
27. Tchernykh, A., Babenko, M., Chervyakov, N., Miranda-López, V., Kuchukov, V., Cortés-Mendoza, J.M., Deryabin, M., Kucherov, N., Radchenko, G., and Avetisyan, A., AC-RRNS: anti-collusion secured data sharing scheme for cloud storage, *Int. J. Approx. Reason.*, 2018, vol. **102**, pp. 60–73.