computation

Article

An Efficient Method for Comparing Numbers and Determining the Sign of a Number in RNS for Even Ranges

Andrei Tchernykh [1,2,3,*], Mikhail Babenko [2,4], Egor Shiriaev [5], Bernardo Pulido-Gaytan [1], Jorge M. Cortés-Mendoza [3], Arutyun Avetisyan [2], Alexander Yu Drozdov [6] and Viktor Kuchukov [5]

1 Computer Science Department, CICESE Research Center, Ensenada 22860, Mexico; lpulido@cicese.edu.mx
2 Control/Management and Applied Mathematics, Institute for System Programming of the Russian Academy of Sciences, 109004 Moscow, Russia; mgbabenko@ncfu.ru (M.B.); arut@ispras.ru (A.A.)
3 School of Electronic Engineering and Computer Science, South Ural State University, 454080 Chelyabinsk, Russia; kortesmendosak@susu.ru
4 North-Caucasus Center for Mathematical Research, North-Caucasus Federal University, 355017 Stavropol, Russia
5 Department of Applied Mathematics and Mathematical Modeling, North-Caucasus Federal University, 355017 Stavropol, Russia; eshiriaev@ncfu.ru (E.S.); vkuchukov@ncfu.ru (V.K.)
6 School of Radio Engineering and Computer Technology, Moscow Institute of Physics and Technology, 141701 Moscow, Russia; alexander.y.drozdov@gmail.com
* Correspondence: chernykh@cicese.mx

Abstract: Fully Homomorphic Encryption (FHE) permits processing information in the form of ciphertexts without decryption. It can ensure the security of information in common technologies used today, such as cloud computing, the Internet of Things, and machine learning, among others. A primary disadvantage for its practical application is the low efficiency of sign and comparison operations. Several FHE schemes use the Residue Number System (RNS) to decrease the time complexity of these operations. Converting from the RNS to the positional number system and calculating the positional characteristic of a number are standard approaches for both operations in the RNS domain. In this paper, we propose a new method for comparing numbers and determining the sign of a number in RNS. We focus on the even ranges that are computationally simple due to their peculiarities. We compare the performance of several state-of-art algorithms based on an implementation in C++ and relatively simple moduli with a bit depth from 24 to 64 bits. The experimental analysis shows a better performance of our approach for all the test cases; it improves the sign detection between 1.93 and 15.3 times and the number comparison within 1.55–11.35 times with respect to all the methods and configurations.

Keywords: residue number system; weighted number system; approximate method; parity number; core functions; mixed number system; non-modular operations; determining the sign of a number

check for updates

Citation: Tchernykh, A.; Babenko, M.; Shiriaev, E.; Pulido-Gaytan, B.; Cortés-Mendoza, J.M.; Avetisyan, A.; Drozdov, A.Y.; Kuchukov, V. An Efficient Method for Comparing Numbers and Determining the Sign of a Number in RNS for Even Ranges. *Computation* **2022**, *10*, 17. https://doi.org/10.3390/computation10020017

Academic Editor: Demos T. Tsahalis

Received: 20 December 2021
Accepted: 19 January 2022
Published: 24 January 2022

1. Introduction

Number comparison and sign detection are simple and basic operations for calculation in computing systems. Both operations are necessary for almost any application area in computing technology: mathematical calculations, rendering, data storage, data transfer, data retrieval, encryption, machine learning, etc. Unfortunately, their application for encrypted data exhibits several limitations.

Sign detection and comparison of numbers in the Residue Number System (RNS) are based on the calculation of the positional characteristic. The algorithms for both operations in RNS are computationally complex because RNS is not a Weighted Number System (WNS). Various methods are used to reduce their computational complexity, such as the Pirlo and Impedovo function [1], Diagonal Function (DF) [2,3], Modified Diagonal Function (MDF) [4], Approximate Method (AM) [5,6], and Core Function (CF) [7].

Both operations are fundamental for the applicability of RNS for both hardware and software implementations. Reducing their computational complexity expands the scope of RNS for solving problems of computational mathematics and mathematical modeling. For instance, the use of RNS in software implementation on the GPU increases the performance of matrix processing [8,9]. Moreover, RNS is a main direction in the implementation of Homomorphic Encryption (HE) software.

Fully HE (FHE) has gained popularity for its applicability in cryptosystems [10]. Performing simple arithmetic operations (homomorphic addition and multiplication) over ciphertexts is its main feature. Hence, the processing of encrypted information does not require its preliminary decryption. This characteristic increases the confidentiality and reliability of the data processing [11,12], for example, in third-party systems such as cloud computing [13,14].

However, comparing ciphertext information is a laborious and computationally complex task [15,16]. Cheon–Kim–Kim–Song (CKKS) [17] and Brakerski/Fan–Vercauteren (BFV) [18] are FHE schemes that use RNS to increase the efficiency of sign detection and numbers' comparison [4,6,19–21].

The comparison operation can be performed in a relatively simple way: subtract one number from another, and if the result of the operation is negative, then the first number is greater than the second one and vice versa. Many methods in RNS work using this property. However, they are often ineffective, since they apply division by large numbers or use computationally complex algorithms to calculate the positional characteristic of the number.

In this paper, we consider a new algorithm for determining the sign of numbers in an even range with RNS. The content of the paper is structured as follows. Section 2 briefly introduces the residue number system. Section 3 describes several methods to determine the sing of numbers in RNS. The performance evaluation is described in Section 4. Finally, we conclude and discuss future works in Section 5.

2. Residue Number System

RNS is a not WNS based on Chinese Remainder Theorem (CRT) [22,23]. It is defined by a vector of coprime numbers called moduli. This vector, also called the RNS basis, can be denoted as $(p_1, p_2, \ldots, p_n)$. The basis is determined by the product of all the p_i elements: $P = \prod_{i=1}^{n} p_i$, where n is the length of the moduli vector. A positional number X is represented in RNS by the residues $(x_1, x_2, \ldots, x_n)$, where

$$x_1 \equiv X(mod\ p_1),$$

$$x_2 \equiv X(mod\ p_2),$$

$$\ldots$$

$$x_n \equiv X(mod\ p_n).$$

A restriction is imposed on the number X such that $X \in [0,\ P-1]$; otherwise, the residue of the number X will be obtained from the number X', where $X' = X - P$. Additionally, the CRT corollary guarantees the uniqueness of the representation of non-negative integers within the interval $[0,\ P-1]$. The representation of negative numbers in RNS considers the radix addition, where $-X$ is represented as $-X = P - X$. This property is used to determine the sign of a number based on converting a number from RNS to WNS, i.e., decimal, binary, etc.

The simplicity to perform basic arithmetic operations is the main advantage of RNS for computing. Addition, multiplication, and subtraction operations are performed according to the following general formula:

$$A * B = (a_1, a_2, \ldots, a_n) * (b_1, b_2, \ldots, b_n) = ((a_1 * b_1) mod\ p_1), \ldots, \\ ((a_n * b_n) mod\ p_n), \tag{1}$$

where $*$ denotes operations such as addition, multiplication, and subtraction.

The modular arithmetic, or operations in RNS, has several particularities: the value of the number in each residue does not depend on other residues, and one cycle of the processor's numerical processing is enough to perform such operations, since this processing can be performed in parallel.

Let us prove that the ranges of numbers in RNS and WNS are equal. For example, for moduli RNS $p_1 = 3$, $p_2 = 5$, $p_3 = 17$, $p_4 = 257$, and $p_5 = 65537$, the RNS range is $P = \prod_{i=1}^{5} p_i = 2^{32} - 1$ and WNS range is $2^{32} - 1$.

Let RNS be given by the basis $(p_1, p_2, \ldots, p_n)$ and the number with residues $X = (x_1, x_2, \ldots, x_n)$. Then, the number X can be represented in the form.

$X = (y_n \cdot p_1 \cdot p_2 \cdot \ldots \cdot p_{n-1} + y_{n-1} \cdot p_1 \cdot p_2 \cdot \ldots \cdot p_{n-2} + \ldots + y_3 \cdot p_1 \cdot p_2 + y_2 \cdot p_1 + y_1)$, where y_i is from 0 to p_k, and $0 \le x_k < p_1 \cdot p_2 \cdot \ldots \cdot p_{k-1} (k = 1, \ldots, n)$ are the coefficients of RNS.

The ranges of numbers represented in RNS and WNS coincide; i.e., we can talk about the presence of a one-to-one correspondence between the set of number representations in RNS and WNS.

The previous equality can be rewritten as:

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

thereby translating numbers into WNS. It establishes the basic concepts of RNS, operations, and translating numbers from WNS to RNS and vice versa.

3. Methods for Determining the Sign of a Number in RNS

3.1. Chinese Remainder Theorem

The following formula is used to convert numbers from RNS to WNS with Chinese Remainder Theorem (CRT)

$$X = \left| \sum_{i=0}^{n} P_i \cdot x_i \cdot |P_i^{-1}|_{p_i} \right|_P \tag{2}$$

where $P_i = \frac{P}{p_i}$ and $|P_i^{-1}|_{p_i}$ is a multiplicative inverse of P_i modulo p_i.

The restored number must be compared with $P/2$. If $X < P/2$, then the number is greater than zero. Two's complement is the most common method of representing signed integers. Its half of the range (the upper half) denotes a negative number and vice versa. To estimate the time complexity, we assume that all RNS moduli are composed of l-bits bit words, so the time complexity is defined by $O(n^2 \cdot l^2)$.

Let us consider an example based on this method.

Example 1. *Let the number X be represented in RNS with moduli* $(3, 5, 7)$ *and residues* $(2, 2, 3)$. *Initially, we calculate the dynamic range* $P = 3 * 5 * 7 = 105$ *and the* P_i *elements:*

$$P_1 = \frac{P}{p_1} = \frac{105}{3} = 35,$$

$$P_2 = \frac{P}{p_2} = \frac{105}{5} = 21,$$

$$P_3 = \frac{P}{p_3} = \frac{105}{7} = 15.$$

Then, we compute the multiplicative inverse, which consists in finding P_i^{-1}, such that $P_i^{-1} \cdot P_i \equiv 1 \; mod \; p_i$. Therefore,

$$|P_1^{-1}|_3 = 2,$$

$$|P_2^{-1}|_5 = 1,$$

$$|P_3^{-1}|_7 = 1.$$

Next, we use Formula (2) to calculate the value of X:

$$X = |35{\cdot}2{\cdot}2 + 21{\cdot}2{\cdot}1 + 15{\cdot}3{\cdot}1|_{105} = |227|_{105} = 17$$

Finally, we compare X with $P/2$; then, $17 < \frac{105}{2}$ or $17 < 52.5$. Therefore, the number is positive.

Algorithm 1 presents the pseudocode of the CRT method.

Algorithm 1: CRT Method.

```
Input: $P$, $(x_1, x_2, \dots, x_n)$, $(p_1, p_2, \dots, p_n)$, and $(P_1, P_2, \dots, P_n)$
Output: $S$
1. $sum = 0$
2. for $i = 1$ to $n$ do:
    2.1 $sum += x_i \left| P_i^{-1} \right|_{p_i} P_i$
3. $X = sum \bmod P$
4. if $X < P/2$:
    4.1 $S = 1$
5. else:
    5.1 $S = 0$
```

However, this method has a significant drawback due to the divisions to obtain X. The division of large numbers by a large P increases the time required for calculations.

3.2. Mixed Radix Conversion Method

The Mixed Radix Conversion (MRC) method consists of a consecutive translation of a number from RNS to WNS. In addition, a pattern in the number in a generalized form is traced by $y_i = (U_i{\cdot}V_i)_{\rho_i}$, where U_i is known as the conversion coefficient and defines an increasing series

$$U_1 = x_1,$$
$$U_2 = |x_2 - x_1|_{p_1},$$
$$U_3 = |x_3 - x_2 - x_1 U_2|_{p_1}.$$

Yassin and Moore [24] found that

$$V_1 = 1,$$
$$V_2 = \left|\frac{1}{p_1}\right|_{p_2} = 1,$$
$$V_3 = \left|\frac{1}{p_1 p_2}\right|_{p_3} = 1.$$

Example 2. *The initial data are a set of moduli* $(127,\ 63,\ 50,\ 13)$, *a set of residues* $(78,\ 41,\ 47,\ 7)$, $V_1 = 1,\ V_2 = 1,\ V_3 = 1,$ *and* $V_4 = 1$.

First, we find the y_i coefficients:

$$y_1 = U_1{\cdot}V_1 = 78,$$
$$y_2 = U_2{\cdot}V_2 = |41 - 78|_{63}{\cdot}1 = 26,$$
$$y_3 = U_3{\cdot}V_3 = |47 - 78 - 127{\cdot}26|_{50}{\cdot}1 = 17,$$
$$y_4 = U_4{\cdot}V_4 = |7 - 78 - 127{\cdot}26 - 127{\cdot}63{\cdot}17|_{13}{\cdot}1 = 9.$$

Second, the number is restored by (2)

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

Finally, we compare X with $P/2$; then, $3,739,847 < 5,200,650$. Therefore, the number is positive. Time complexity is $O(n^2 \cdot l^2)$.

Algorithm 2 describes the pseudocode of the MRC method.

Algorithm 2: MRC Method.

```
Input: $(x_1, x_2, \ldots, x_n)$ and $(p_1, p_2, \ldots, p_n)$
Output: $S$
1. $U_1 = x_1, k_1 = 0, h = 1$, and $P = 1$
2. for $i = 2$ to $n$ do
2.1 $h * = n_{i-1}$
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. $sum + = U_i\ mod\ p_i$
4. for $i = 1$ to $n$ do
4.1 $P* = p_i$
4.2 $sum + = P * (U_i)\ mod\ p_i$
5 $X = sum$
6 if $X < P/2$:
6.1 $S = 1$
7. else:
7.1  $S = 0$
```

Effective implementation of this method allows avoiding division by a large P. However, the method uses many modular operations related to the vector $(p_1,\ p_2,\ \ldots, p_n)$.

3.3. *Approximate Method*

The algorithm for finding the Euclidean division by the range RNS is computationally complex. To reduce the computational complexity of the Approximate Method (AM), it is proposed to use weighted coefficients, which allow replacing the operation of Euclidean division with the operation of taking a fractional part.

AM is based on the mapping from $[0, P-1]$ to $[0, 2)$ to avoid the division by a large P.

The mapping allows calculating the positional characteristic of the number, hence, to determine the sign of the number or compare numbers with each other. The positional characteristic is originally calculated as:

$$V(X) = \left| \sum_{i=1}^{n} \left(\frac{2}{p_i} \left| |P_i^{-1}|_{p_i} \cdot x_i \right| \right)_{2^{-N}} \right|_2. \tag{3}$$

However, the use of (3) as a positional characteristic requires the use of rational fractions or their approximation using numbers with a fixed precision, which increases the computational complexity of the algorithm. In [25], a modification of the methods from [5] was proposed, which allows retaining the advantages of the method from [5] but at the same time avoiding the use of rational numbers.

$$VC(X) = \left| \sum_{i=1}^{n} W_i \cdot x_i \right|_{2^N}, \tag{4}$$

where $W_i = \lceil 2^N |P_i^{-1}|_{p_i} / p_i \rceil$, with $N = \lceil \log_2(P\rho) \rceil$, and $\rho = -n + \sum_{i=1}^{n} p_i$. If $VC(X) < 2^{N-1}$ then $X > 0$ else $X < 0$. Time complexity is $O((n+1)l \cdot \log_2((n+1)l))$.

Example 3. $X = (1,\ 2,\ 3)$, *moduli RNS:* $(11,\ 12,\ 13)$, $P = p_1 \cdot p_2 \cdot p_3 = 1716$, $P_1 = \frac{P}{p_1} = 156$, $P_2 = \frac{P}{p_2} = 143$, $P_3 = \frac{P}{p_3} = 132$, $\rho = -3 + 11 + 12 + 13 = 33$, *and* $N = \lceil log_2(331716) \rceil = 16$.

We calculate the coefficients W_i:

$$W_1 = \left\lceil \frac{2^N |P_1^{-1}|_{p_1}}{p_i} \right\rceil = \left\lceil \frac{2^{16}|156^{-1}|_{11}}{11} \right\rceil = 35,747,\ W_2 = \left\lceil \frac{2^N |P_2^{-1}|_{p_2}}{p_2} \right\rceil = \left\lceil \frac{2^{16}|143^{-1}|_{12}}{12} \right\rceil = 60,075,$$

$$W_3 = \left\lceil \frac{2^N |P_3^{-1}|_{p_3}}{p_3} \right\rceil = \left\lceil \frac{2^{16}|132^{-1}|_{13}}{13} \right\rceil = 35,289$$

Thus, we can translate the number based on the coefficients

$$VC(X) = |35,747 \cdot 1 + 60,075 \cdot 2 + 35,289 \cdot 3|_{2^{16}} = 65,156$$

Therefore, the number is negative because $65,156 > 2^{15}$.
Algorithm 3 shows the pseudocode of the AM.

Algorithm 3: AM Method.

```
Input: $(x_1, x_2, \ldots, x_n)$, $(W_1, W_2, \ldots, W_n)$, and N
Output: $S$
1. for $i = 1$ to $n$ do
1.1 $sum += W_i \cdot x_i$
2 $X = sum\ mod\ 2^N$
3. if $(X \gg (N-1)) == 1$:
3.1 $S = 1$
4. else:
4.1 $S = 0$
```

3.4. Diagonal Function

The positional characteristic of a number in RNS can be also determined by the so-called Diagonal Function (DF), which is defined by $D(X) = \sum_{i=1}^{n} \left\lfloor \frac{X}{p_i} \right\rfloor$ [2]. To calculate $D(X)$ using moduli RNS, use the following formula:

$$D(X) = \left| \sum_{i=1}^{n} k_i^* \cdot x_i \right|_{SQ}, \tag{5}$$

where $k_i^* = |-P_i^{-1}|_{SQ}$, $SQ = P_1 + P_2 + \ldots + P_n$.

The sign of the number depends on the comparison of $D(X)$ and $SQ/2$. Let us look at an example of comparing numbers. The time complexity of the algorithm is $O\left(((n-1)l + \log_2 n)^2\right)$.

Example 4. *We consider the numbers of previous examples,* $X = (2,\ 2,\ 3)$ *and moduli* $(3,\ 5,\ 7)$. *First, we calculate the values* $SQ = 35 + 21 + 15 = 71$ *and* k_i^*.

$$k_1^* = |-p_1^{-1}|_{71} = |-3^{-1}|_{71} = 47,$$

$$k_2^* = |-p_2^{-1}|_{71} = |-5^{-1}|_{71} = 14,$$

$$k_3^* = |-p_3^{-1}|_{71} = |-7^{-1}|_{71} = 10.$$

Then, we find the value of the DF:

$$D(X) = |2 \cdot 47 + 2 \cdot 14 + 3 \cdot 10|_{71} = 10.$$

Since $10 < 71/2$, then X is positive.

Algorithm 4 describes the pseudocode of the DF method.

Algorithm 4: DF Method.

```
Input: $SQ$, $(x_1, x_2, \ldots, x_n)$, and $(k_1^*, k_2^*, \ldots, k_n^*)$
Output: $S$
1. for $i = 1$ to $n$ do
1.1 $sum+ = k_i^* \cdot x_i$
2 $X = sum \ mod \ SQ$
3. if $X < SQ/2$:
3.1 $S = 1$
4. else:
4.1 $S = 0$
```

3.5. Core Function

Burgess [26] proposed the minimal Akushsky Core Function (CF) based on the result obtained in [27,28]. This approach is similar to the DF method, and it does not need to calculate the critical cores.

The analytical form of the Pirlo function is calculated as:

$$Pi(X) = \left| \sum_{i=1}^{n} k_i^{**} \cdot x_i \right|_{P_n}, \quad (6)$$

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

If $Pi(X) < P_n/2$, then the number is positive, since the function under study is monotonically increasing. Time complexity is $O((n-1)^2 l^2)$.

Let us consider an example of comparing numbers.

Example 5. *We take the numbers of previous examples,* $X = (2,\ 2,\ 3)$ *and moduli* $(3,\ 5,\ 7)$*. First, let us calculate the values* $P_3 = 15$ *and* k_i^{**} *as:*

$$k_1^{**} = \left| \frac{|P_1^{-1}|_{p_1} \cdot P_1}{p_3} \right| = \left| \frac{2 \cdot 35}{7} \right| = 10,$$

$$k_1^{**} = \left| \frac{|P_2^{-1}|_{p_2} \cdot P_2}{p_3} \right| = \left| \frac{1 \cdot 21}{7} \right| = 3,$$

$$k_1^{**} = \left| \frac{|P_3^{-1}|_{p_3} \cdot P_3}{p_3} \right| = \left| \frac{1 \cdot 15}{7} \right| = 2.$$

Then, we compute the Pirlo function:

$$Pi(X) = |2 \cdot 10 + 2 \cdot 3 + 3 \cdot 2|_{15} = 2.$$

Since $2 < 15/2$, the number is positive.

Algorithm 5 describes the pseudocode of the CF method.

Algorithm 5: CF Method.

```
Input: $P_n$, $(x_1, x_2, \ldots, x_n)$, and $(k_1^{**}, k_2^{**}, \ldots, k_n^{**})$
Output: $S$
1. for $i = 1$ to $n$ do
    1.1 $sum+ = k_i^{**} \cdot x_i$
2. $X = sum \ mod \ P_n$
3. if $X < P_n/2$:
    3.1 $S = 1$
4. else:
    4.1 $S = 0$
```

3.6. Determining the Sign of a Number

Knowing the sign of a number can increase the performance of number comparison in RNS; the difference between numbers X and Y provides information about their comparison. Determining the Sign of a Number (DSN) in RNS is based on the fact that the residues $\overline{x_i}$ of the negative number X are the complement of x_i to the modulus p_i. Hence, the following relationship holds:

$$S(X) = \begin{cases} 0, \ if \ 0 \leq X < \frac{P-1}{2}, \\ 1, \ if \frac{P-1}{2} \leq X < P. \end{cases} \tag{7}$$

The formula based on CRT and an AM allows determining the sign of the number based on the following equation [29].

$$\frac{X}{P} = \left| \sum\nolimits_{i=1}^{n} \frac{x_i \cdot |P_i^{-1}|_{p_i}}{p_i} \right|_1 < \frac{1}{2}. \tag{8}$$

$X > 0$ if the equation holds; otherwise, $X < 0$.

Let us consider the algorithm for comparing numbers in RNS based on DSN.

Example 6. *Let the number in RNS be* $(1, \ 0, \ 4)$ *with the basis* $(3, \ 5, \ 7)$*. We determine the sign of the number by calculating Formula (8):*

$$\frac{X}{P} = \left| \sum_{i=1}^{3} \frac{x_i \cdot |P_i^{-1}|_{p_i}}{p_i} \right|_1 = \left| \frac{2}{3} + \frac{4}{7} \right|_1 = \frac{5}{21} < \frac{1}{2}.$$

So, the number is positive.

Algorithm 6 shows the pseudocode of the DSN method.

Algorithm 6: DSN Method.

```
Input: $(x_1, x_2, \ldots, x_n)$, $(p_1, p_2, \ldots, p_n)$, and $(|P_1^{-1}|_{p_1}, |P_2^{-1}|_{p_2}, \ldots, |P_n^{-1}|_{p_n})$
Output: $S$
1. for $i = 1$ to $n$ do:
    1.1 $S+ = \frac{x_i \cdot (P_i^{-1})_{p_i}}{p_i}$
2. $S = S \ mod \ 1$
3. if $S < \frac{1}{2}$:
    3.1 $S = 1$
4. else:
    4.1 $S = 0$
```

3.7. Modified Diagonal Function

The Modified Diagonal Function (MDF) method reduces the computational complexity of comparing numbers in the RNS [5]. It proposes a new positional characteristic based on the DF and an AM. The essence of the MDF method is to calculate the relative value of the

DF to SQ, which allows replacing the computationally complex operation of finding the residue of division by SQ by taking the fractional part of the number, and the coefficients k_i^* are replaced by the relative value k_i^* of SQ in Formula (5).

$$D'(X) = \left| \sum_{i=1}^{n} \hat{k}_i \cdot x_i \right|_{2^{N_M}}, \tag{9}$$

where $\hat{k}_i = \left[k_i^* \cdot \frac{2^{N_M}}{SQ} \right]$, $N_M > [\log_2(SQ \cdot (m-1))]$, and $m = \max\limits_{1 \le i \le n} p_i$.

Let us look at an example of comparing numbers. Time complexity is

$$O\Big(((n-1)l + \log_2 n) \log_2((n-1)l + \log_2 n) \Big).$$

Example 7. *We use the previously numbers* $X = (2, 2, 3)$ *and moduli* $(3, 5, 7)$. *Then, we take* $2^{N_M} = 8.755$ *and value* k_i^* *from the diagonal function:*

$$SQ = 35 + 21 + 15 = 71,$$

$$k_1^* = |-p_1^{-1}|_{71} = |-3^{-1}|_{71} = 47,$$

$$k_2^* = |-p_2^{-1}|_{71} = |-5^{-1}|_{71} = 14.$$

We calculate $\hat{k}_i$

$$\hat{k}_1 = 47 \cdot \frac{8.755}{71} = 5.795,$$

$$\hat{k}_2 = 14 \cdot \frac{8.755}{71} = 1.722,$$

$$\hat{k}_3 = 10 \cdot \frac{8.755}{71} = 1.23.$$

We find the value of the MDF:

$$D'(X) = |2 \cdot 5.795 + 2 \cdot 1.722 + 3 \cdot 1.23|_{8.755} = 1.214.$$

The number is positive, since $1.214 < 8.755/2$. Algorithm 7 shows the pseudocode of the MDF method.

Algorithm 7: MDF Method.

```
Input: $(x_1, x_2, \ldots, x_n)$, $\left(\hat{k}_1, \hat{k}_2, \ldots, \hat{k}_n\right)$, and $2^{N_M}$
Output: $S$
1. for $i = 1$ to $n$ do
    1.1 $sum += \hat{k}_i \cdot x_i$
2. $X = sum \bmod 2^{N_M}$
3. if $X < 2^{N_M}/2$:
    3.1 $S = 1$
4. else :
    4.1 $S = 0$
```

3.8. Determining the Sign of a Number in RNS with an Even Range

Here, we propose the efficient method Determining the Sign of a Number (DSN) in RNS with an Even Range (DSN-EN) for comparing numbers and determining the sign of a number for the case of an even RNS range.

Since the RNS range is an Even Number (EN), one of the RNS moduli is an EN. Without loss of generality, we assume that the module p_n is even; then, using the property $\left\lfloor \frac{\left\lfloor \frac{X}{a} \right\rfloor}{b} \right\rfloor = \left\lfloor \frac{X}{a \cdot b} \right\rfloor$, the determination of a sign is reduced to a two-stage algorithm.

The first stage is dividing by an integer $P_n = \frac{P}{p_n}$. The second stage is dividing by an integer $\frac{p_n}{2}$.

Formally, it is determined by the following formula:

$$S\,(X) = \left\lfloor \frac{2 \cdot X}{P} \right\rfloor = \left\lfloor \left\lfloor \frac{X}{P_n} \right\rfloor \cdot \frac{2}{p_n} \right\rfloor. \tag{10}$$

The correctness of (10) follows from the proof of the following Theorem 1.

Theorem 1. *If X, a, and b are integer numbers with $a, b \geq 2$, then the equality holds:*

$$\left\lfloor \left\lfloor \frac{X}{a} \right\rfloor \cdot \frac{1}{b} \right\rfloor = \left\lfloor \frac{X}{a \cdot b} \right\rfloor. \tag{11}$$

Proof. Due to $\frac{X}{a} = \frac{X - |X|_a}{a}$, Equality (11) takes the form:

$$\left\lfloor \left\lfloor \frac{X}{a} \right\rfloor \cdot \frac{1}{b} \right\rfloor = \left\lfloor \frac{X - |X|_a}{a \cdot b} \right\rfloor. \tag{12}$$

We represent X in the form $X = a \cdot b \cdot \left\lfloor \frac{X}{a \cdot b} \right\rfloor + |X|_{a \cdot b}$. Substituting it in (12), we have:

$$\left\lfloor \frac{X - |X|_a}{a \cdot b} \right\rfloor = \left\lfloor \frac{X}{a \cdot b} \right\rfloor + \left\lfloor \frac{|X|_{a \cdot b} - |X|_a}{a \cdot b} \right\rfloor. \tag{13}$$

Let $X = \omega \cdot a + \gamma$, where $0 \leq \gamma \leq a - 1$, we obtain:

$$|\omega \cdot a|_{a \cdot b} = \omega \cdot a - \left\lfloor \frac{\omega \cdot a}{a \cdot b} \right\rfloor \cdot a \cdot b = a \Big(\omega - \left\lfloor \frac{\omega}{b} \right\rfloor \cdot b \Big) = a \cdot |\omega|_b. \tag{14}$$

Considering that $0 \leq |\omega|_b \leq b - 1$, then

$$0 \leq a \cdot |\omega|_b + \gamma \leq a \cdot (b-1) + a - 1 = a \cdot b - 1$$

and

$$|X|_{a \cdot b} = |\omega \cdot a + \gamma|_{a \cdot b} = \left| |\omega \cdot a|_{a \cdot b} + \gamma \right|_{a \cdot b} = |a \cdot |\omega|_b + \gamma|_{a \cdot b} = a \cdot |\omega|_b + \gamma.$$

Hence, $|X|_{a \cdot b} - |X|_a = a \cdot |\omega|_b + \gamma - \gamma = a \cdot |\omega|_b$ and

$$\left\lfloor \frac{|X|_{a \cdot b} - |X|_a}{a \cdot b} \right\rfloor = \left\lfloor \frac{|\omega|_b}{b} \right\rfloor = 0.$$

Substituting this result in (13), we have:

$$\left\lfloor \frac{X - |X|_a}{a \cdot b} \right\rfloor = \left\lfloor \frac{X}{a \cdot b} \right\rfloor.$$

The theorem is proved. □

Corollary 1. *Equation (10) is correct.*

Proof. Let $a = P_n$ and $b = \frac{p_n}{2}$. Since p_n is an even number, then $\frac{p_n}{2}$ is an integer number. Therefore, the conditions of Theorem 1 are satisfied and the corollary is proved. The time complexity of the algorithm is $O\left(n^2 l\right)$. $\square$

Example 8. *Let the RNS be given by the moduli* $p_1 = 17,\ p_2 = 19,\ p_3 = 23,$ *and* $p_4 = 32$. *We determine the sign of a number* $X = (16,\ 18,\ 22,\ 15)$ *and* $Y = (0,\ 0,\ 0,\ 16)$. *Let us calculate the synoptic weights:*

$$w_{1,2} = |p_1^{-1}|_{p_2} = \left|\frac{1}{17}\right|_{19} = 9,$$

$$w_{1,3} = |p_1^{-1}|_{p_3} = \left|\frac{1}{17}\right|_{23} = 19,$$

$$w_{1,4} = |p_1^{-1}|_{p_4} = \left|\frac{1}{17}\right|_{32} = 17,$$

$$w_{2,3} = |p_2^{-1}|_{p_3} = \left|\frac{1}{19}\right|_{23} = 17,$$

$$w_{2,4} = |p_2^{-1}|_{p_4} = \left|\frac{1}{19}\right|_{32} = 27,$$

$$w_{3,4} = |p_3^{-1}|_{p_4} = \left|\frac{1}{23}\right|_{32} = 7.$$

The number X is positive since

$$S(X) = \left\lfloor 2\cdot\frac{x_4^{(3)}}{p_4} \right\rfloor = \left\lfloor 2\cdot\frac{15}{32} \right\rfloor = 0 < 1,$$

and Y is negative because

$$S(Y) = \left\lfloor 2\cdot\frac{y_4^{(3)}}{p_4} \right\rfloor = \left\lfloor 2\cdot\frac{16}{32} \right\rfloor = 1.$$

Algorithm 8 shows the pseudocode of the DSN with the EN method.

Algorithm 8: DSN-EN.

```
Input: $(x_1, x_2, \ldots, x_n)$, $(p_1, p_2, \ldots, p_n)$, and $(w_{1,2}, w_{1,3}, \ldots, w_{n,n})$
Output: $S$
1 for $i = 1$ to $n$ do
  1.1 for $j = i + 1$ to $n$ do
    1.1.1 $x_j = \left|\left(x_j - x_i\right)\cdot w_{i,j}\right|_{p_j}$
2. if $\left|2\cdot\frac{x_n}{p_n}\right| < 1$:
  2.1 $S = 1$
4. else :
  4.1 $S = 0$
```

4. Performance Evaluation

The proposed algorithm described in Section 3.8 has the advantage among the methods described in Section 3.

To confirm the properties, we implement all the algorithms using the high-level language C++ and compare their performance. The experiments are carried out on operating systems Windows 10 Home Edition on a computer with Intel Core i5-8250U 1.60 GHz, RAM DDR4 8 GB 1196 MHz, and SSD 512 GB.

For the analysis, we use a 128-bit number transferred to RNS, a set of six RNS moduli, and a width of 24 to 64 bits. We vary a vector length of 3 to 8 moduli with a 32-bit width.

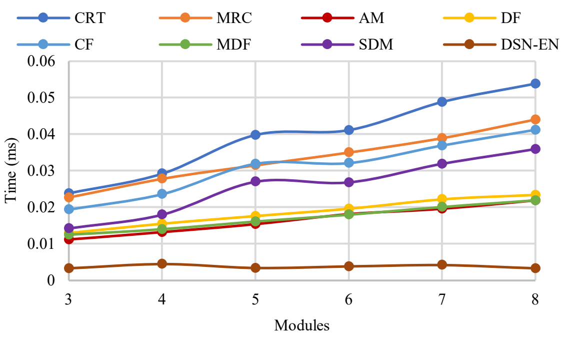
The experimental results report the maximum, minimum, and average of 10,000 runs of each method. Additionally, we incorporate the average calculated based on the maximum and minimum values.

The information of the execution is presented in Appendix A. For simplicity, we only analyze the significant discoveries of the results. The average values have stable linear growth. The minimum values are prevailing because the average values of each method are very close to them. Hence, the maximum values are not common causes.

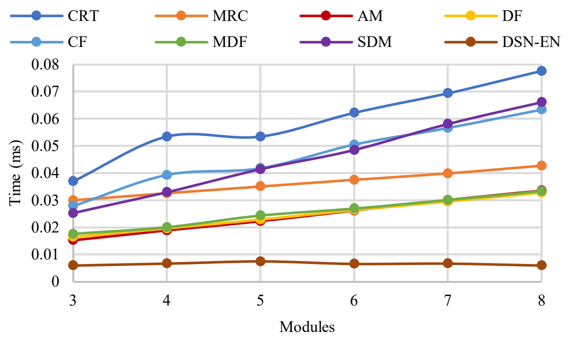
To avoid time measuring biases, the mean values are calculated as the average of 10,000 measuring of the maximum values and 10,000 measuring of the minimum values. We also consider the medium values of all methods to determine the most efficient ones.

Comparative Analysis

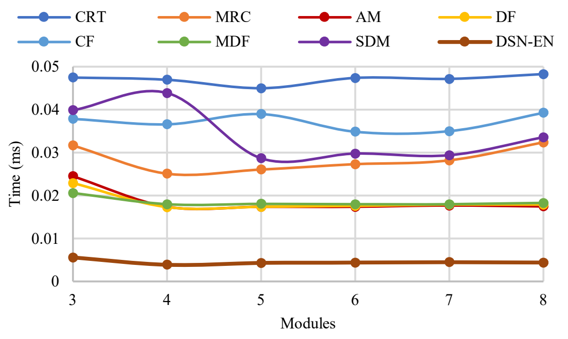
The performance of all methods for the operations of determining the sign of a number and number comparison is presented in Figure 1. DSN-EN has the highest performance for both operations with different numbers of modules p and dynamic bit width. AM, DF, and MDF are in the group of the second most efficient methods for all operations.



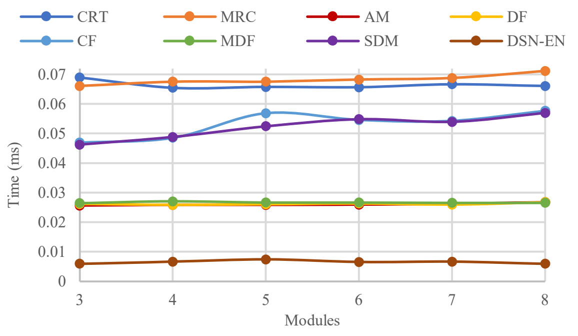
(**a**) Sign



(**b**) Comparison



(**c**) Sign



(**d**) Comparison

Figure 1. Performance of all methods to (**a**) determine the number's sign with the dynamic number of modules p, (**b**) compare dynamic numbers of modules p, (**c**) determine the number's sign with a dynamic bit width, and (**d**) compare numbers with a dynamic bit width.

The results with a different number of modules p show: Considering the sign detection with respect to DSN-EN, the efficiency of AM is between 1.93 and 5.64 times worst. Likewise, DF and MDF are from 2.44 to 6.1 and from 2.11 to 5.64 times worst, respectively. Similar results present the comparison of numbers, DSN-EN improves AM within 1.55–4.6 times, DF among 1.75–4.47 times, and MDF within 1.93–4.57 times.

The results of comparison with different dynamic bit widths exhibit that DSN-EN is more efficient than MDF from 2.68 to 3.61 times, AM within 2.93–3.46 times, and MD between 2.97 and 3.44 times. For comparison of numbers, the differences are in 2.45–3.47, 2.48–3.47, and 2.56–3.43 for AM, DF, and MDF, respectively.

Figure 2 present a comparison between DSN-EN and AM, both methods with the best overall performance. From Figure 2a, we observe that the time difference is increased with respect to the number of modules for both operations. Figure 2b shows the execution time varying a dynamic bit width.

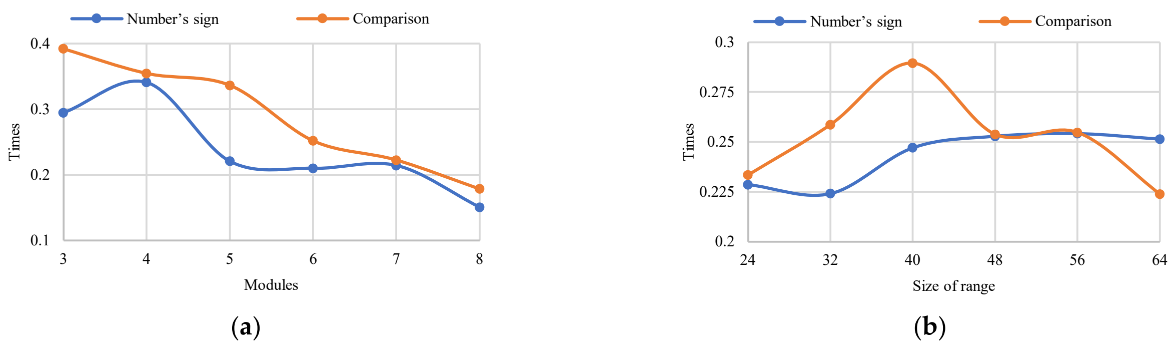


Figure 2. Comparison of DSN-EN and AM methods for determination of the number's sign and comparison with (**a**) a dynamic number of modules p and (**b**) a dynamic bit width.

In general, the proposed method is several times more productive than AM, which was determined as the most efficient of the existing state-of-the-art ones. We consider that the DSN-EN can be applied efficiently in systems that use RNS.

5. Conclusions

Determining the sign of a number is fundamental for the implementation of several techniques with privacy preserving in untrusted environments. Current homomorphic encryption approaches try to make the implementation of this operation more efficient by using a residual number system. Unfortunately, the methods based on converting a number to a weight number system and determining the sign are inefficient and slow. The calculation of positional characteristics is better in performance but not efficient.

We propose a new method to determine the sign of a number in RNS and provide a theoretical foundation for when one modulo is an even number. We provide its performance evaluation compared with other determining the sign methods. The results show the advantages of our approach for different modules numbers and dynamic bit widths. In the future, we will study determining the sign of a number and comparing numbers in RNS for odd ranges.

Author Contributions: The authors have contributed in different parts of the paper preparation, as follows: Conceptualization, A.T. and M.B.; methodology, A.T., M.B., E.S. and A.A.; software, E.S., B.P.-G. and J.M.C.-M.; validation, E.S., B.P.-G., J.M.C.-M. and V.K.; formal analysis, A.T., M.B., E.S. and A.Y.D.; investigation, A.T., M.B., E.S. and A.A.; resources, E.S., and B.P.-G.; data curation, B.P.-G. and J.M.C.-M.; writing—original draft preparation, A.T., M.B., E.S., A.Y.D. and V.K.; writing—review and editing, A.T., B.P.-G., J.M.C.-M. and A.A.; visualization, B.P.-G. and J.M.C.-M.; supervision, A.T. and M.B.; project administration, A.T. and M.B.; funding acquisition, M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Education and Science of the Russian Federation (Project 075-15-2020-915).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1 presents the time in milliseconds (ms) needed to determine the number's sign and comparison of numbers for considered methods with a dynamic number of modules p based on the 10,000 measurements.

Table A1. Operation time with a dynamic number of modules p (ms).

Method	Modules	Number's Sign				Comparison			
		Max	Mean	Min	Mean–Min	Max	Mean	Min	Mean–Min
CRT	3	0.1130	0.0238	0.0229	0.0009	0.1876	0.0370	0.0355	0.0015
	4	0.1451	0.0292	0.0287	0.0005	0.1562	0.0534	0.0434	0.0100
	5	0.1547	0.0398	0.0381	0.0017	0.2381	0.0534	0.0510	0.0024
	6	0.2355	0.0411	0.0404	0.0007	0.2471	0.0622	0.0587	0.0035
	7	0.2090	0.0488	0.0467	0.0021	0.2725	0.0694	0.0665	0.0029
	8	0.2499	0.0538	0.0520	0.0018	0.4214	0.0776	0.0741	0.0035
MRC	3	0.0895	0.0227	0.0127	0.0100	0.0702	0.0300	0.0100	0.0200
	4	0.0988	0.0278	0.0172	0.0106	0.0927	0.0326	0.0124	0.0202
	5	0.0839	0.0315	0.0208	0.0107	0.0700	0.0351	0.0146	0.0205
	6	0.1208	0.0350	0.0245	0.0105	0.0933	0.0375	0.0169	0.0206
	7	0.1400	0.0389	0.0282	0.0107	0.1394	0.0399	0.0193	0.0206
	8	0.1525	0.0440	0.0329	0.0111	0.1147	0.0427	0.0221	0.0206
AM	3	0.0657	0.0112	0.0109	0.0003	0.0617	0.0153	0.0148	0.0005
	4	0.0917	0.0132	0.0130	0.0002	0.0921	0.0189	0.0183	0.0006
	5	0.0902	0.0154	0.0151	0.0003	0.1278	0.0223	0.0217	0.0006
	6	0.0654	0.0181	0.0173	0.0008	0.1244	0.0262	0.0252	0.0010
	7	0.0935	0.0196	0.0193	0.0003	0.1151	0.0301	0.0286	0.0015
	8	0.1110	0.0219	0.0214	0.0005	0.1867	0.0336	0.0320	0.0016
DF	3	0.0598	0.0130	0.0114	0.0016	0.0819	0.0165	0.0157	0.0008
	4	0.0685	0.0155	0.0134	0.0021	0.0833	0.0196	0.0189	0.0007
	5	0.0835	0.0176	0.0153	0.0023	0.1270	0.0229	0.0221	0.0008
	6	0.1092	0.0196	0.0173	0.0023	0.1365	0.0264	0.0252	0.0012
	7	0.1061	0.0222	0.0193	0.0029	0.1575	0.0295	0.0284	0.0011
	8	0.0948	0.0234	0.0211	0.0023	0.1969	0.0328	0.0315	0.0013
CF	3	0.1025	0.0194	0.0190	0.0004	0.1187	0.0279	0.0271	0.0008
	4	0.1210	0.0236	0.0230	0.0006	0.1910	0.0393	0.0372	0.0021
	5	0.1386	0.0319	0.0312	0.0007	0.1840	0.0418	0.0401	0.0017
	6	0.1769	0.0321	0.0312	0.0009	0.2156	0.0505	0.0466	0.0039
	7	0.1831	0.0369	0.0361	0.0008	0.3287	0.0566	0.0546	0.0020
	8	0.1984	0.0412	0.0399	0.0013	0.2907	0.0634	0.0609	0.0025
MDF	3	0.0817	0.0125	0.0118	0.0007	0.0942	0.0176	0.0163	0.0013
	4	0.0660	0.0140	0.0138	0.0002	0.0940	0.0201	0.0195	0.0006
	5	0.0909	0.0161	0.0158	0.0003	0.1493	0.0244	0.0227	0.0017
	6	0.0982	0.0180	0.0178	0.0002	0.1258	0.0270	0.0258	0.0012
	7	0.1013	0.0201	0.0197	0.0004	0.1293	0.0301	0.0290	0.0011
	8	0.1100	0.0219	0.0217	0.0002	0.1768	0.0334	0.0322	0.0012
SDM	3	0.0983	0.0142	0.0138	0.0004	0.1164	0.0253	0.0241	0.0012
	4	0.1188	0.0180	0.0178	0.0002	0.1531	0.0330	0.0316	0.0014
	5	0.1186	0.0270	0.0256	0.0014	0.1809	0.0414	0.0390	0.0024
	6	0.1137	0.0268	0.0262	0.0006	0.2073	0.0485	0.0470	0.0015
	7	0.1652	0.0319	0.0307	0.0012	0.2289	0.0581	0.0551	0.0030
	8	0.1780	0.0359	0.0349	0.0010	0.3036	0.0661	0.0632	0.0029
DSN-EN	3	0.0487	0.0033	0.0033	0.0000	0.0428	0.0060	0.0059	0.0001
	4	0.0359	0.0045	0.0042	0.0003	0.0596	0.0067	0.0066	0.0001
	5	0.0247	0.0034	0.0033	0.0001	0.0443	0.0075	0.0074	0.0001
	6	0.0384	0.0038	0.0038	0.0000	0.0675	0.0066	0.0065	0.0001
	7	0.0668	0.0042	0.0042	0.0000	0.0400	0.0067	0.0065	0.0002
	8	0.0343	0.0033	0.0033	0.0000	0.0218	0.0060	0.0058	0.0002

Table A2 shows the time (ms) needed to determine the number's sing and comparison of numbers for with a dynamic bit width based on the 10,000 measurements.

Table A2. Operation time with a dynamic bit width (ms).

Method	**Size**	**Number's Sign**				**Comparison**			
		Max	**Mean**	**Min**	**Mean–Min**	**Max**	**Mean**	**Min**	**Mean–Min**
CRT	24	0.2849	0.0475	0.0415	0.0060	0.2974	0.0690	0.0625	0.0065
	32	0.2035	0.0470	0.0442	0.0028	0.2781	0.0655	0.0627	0.0028
	40	0.1942	0.0450	0.0433	0.0017	0.2959	0.0658	0.0626	0.0032
	48	0.2344	0.0474	0.0456	0.0018	0.3104	0.0657	0.0626	0.0031
	56	0.2078	0.0472	0.0455	0.0017	0.3846	0.0667	0.0629	0.0038
	64	0.2626	0.0483	0.0465	0.0018	0.3443	0.0661	0.0630	0.0031
MRC	24	0.1504	0.0317	0.0220	0.0097	0.0761	0.0661	0.0155	0.0506
	32	0.1359	0.0251	0.0244	0.0007	0.0950	0.0676	0.0169	0.0507
	40	0.1292	0.0261	0.0255	0.0006	0.0844	0.0676	0.0171	0.0505
	48	0.1137	0.0273	0.0262	0.0011	0.2031	0.0683	0.0177	0.0506
	56	0.1168	0.0282	0.0270	0.0012	0.0920	0.0688	0.0182	0.0506
	64	0.1412	0.0324	0.0319	0.0005	0.0978	0.0712	0.0206	0.0506
AM	24	0.1220	0.0245	0.0171	0.0074	0.1436	0.0257	0.0250	0.0007
	32	0.0679	0.0174	0.0172	0.0002	0.1363	0.0259	0.0251	0.0008
	40	0.1027	0.0174	0.0172	0.0002	0.0979	0.0259	0.0250	0.0009
	48	0.1103	0.0174	0.0172	0.0002	0.1273	0.0260	0.0250	0.0010
	56	0.0980	0.0177	0.0173	0.0004	0.1488	0.0263	0.0251	0.0012
	64	0.0990	0.0175	0.0173	0.0002	0.1615	0.0268	0.0251	0.0017
DF	24	0.1767	0.0229	0.0172	0.0057	0.1777	0.0262	0.0251	0.0011
	32	0.0775	0.0173	0.0173	0.0000	0.1746	0.0259	0.0251	0.0008
	40	0.1014	0.0174	0.0172	0.0002	0.1795	0.0261	0.0251	0.0010
	48	0.0984	0.0176	0.0172	0.0004	0.1124	0.0264	0.0251	0.0013
	56	0.0680	0.0179	0.0173	0.0006	0.1250	0.0260	0.0252	0.0008
	64	0.1006	0.0179	0.0174	0.0005	0.1397	0.0268	0.0252	0.0016
CF	24	0.7379	0.0379	0.0334	0.0045	0.1949	0.0470	0.0439	0.0031
	32	0.1392	0.0366	0.0349	0.0017	0.2062	0.0486	0.0466	0.0020
	40	0.1810	0.0390	0.0370	0.0020	0.4973	0.0569	0.0544	0.0025
	48	0.1391	0.0349	0.0342	0.0007	0.2373	0.0546	0.0527	0.0019
	56	0.1351	0.0350	0.0340	0.0010	0.2282	0.0543	0.0521	0.0022
	64	0.1467	0.0393	0.0376	0.0017	0.2049	0.0577	0.0548	0.0029
MDF	24	0.1079	0.0206	0.0194	0.0012	0.1450	0.0265	0.0257	0.0008
	32	0.1048	0.0180	0.0178	0.0002	0.1303	0.0271	0.0258	0.0013
	40	0.1246	0.0181	0.0177	0.0004	0.1711	0.0267	0.0257	0.0010
	48	0.0723	0.0180	0.0177	0.0003	0.1280	0.0267	0.0257	0.0010
	56	0.0964	0.0180	0.0177	0.0003	0.1211	0.0266	0.0256	0.0010
	64	0.1150	0.0183	0.0178	0.0005	0.1311	0.0266	0.0258	0.0008
SDM	24	0.1709	0.0399	0.0286	0.0113	0.2151	0.0463	0.0444	0.0019
	32	0.1845	0.0439	0.0401	0.0038	0.1881	0.0489	0.0468	0.0021
	40	0.1227	0.0287	0.0279	0.0008	0.2313	0.0525	0.0505	0.0020
	48	0.1145	0.0298	0.0286	0.0012	0.2429	0.0549	0.0519	0.0030
	56	0.1317	0.0294	0.0287	0.0007	0.2404	0.0540	0.0519	0.0021
	64	0.1143	0.0336	0.0314	0.0022	0.2102	0.0570	0.0536	0.0034
DSN-EN	24	0.0259	0.0056	0.0034	0.0022	0.0428	0.0060	0.0059	0.0001
	32	0.0274	0.0039	0.0038	0.0001	0.0596	0.0067	0.0066	0.0001
	40	0.0392	0.0043	0.0042	0.0001	0.0443	0.0075	0.0074	0.0001
	48	0.0342	0.0044	0.0038	0.0006	0.0675	0.0066	0.0065	0.0001
	56	0.0392	0.0045	0.0037	0.0008	0.0400	0.0067	0.0065	0.0002
	64	0.0480	0.0044	0.0034	0.0010	0.0218	0.0060	0.0058	0.0002

Figures A1–A8 present the maximum, minimum, and average time in seconds (s) of sign detection and comparison.

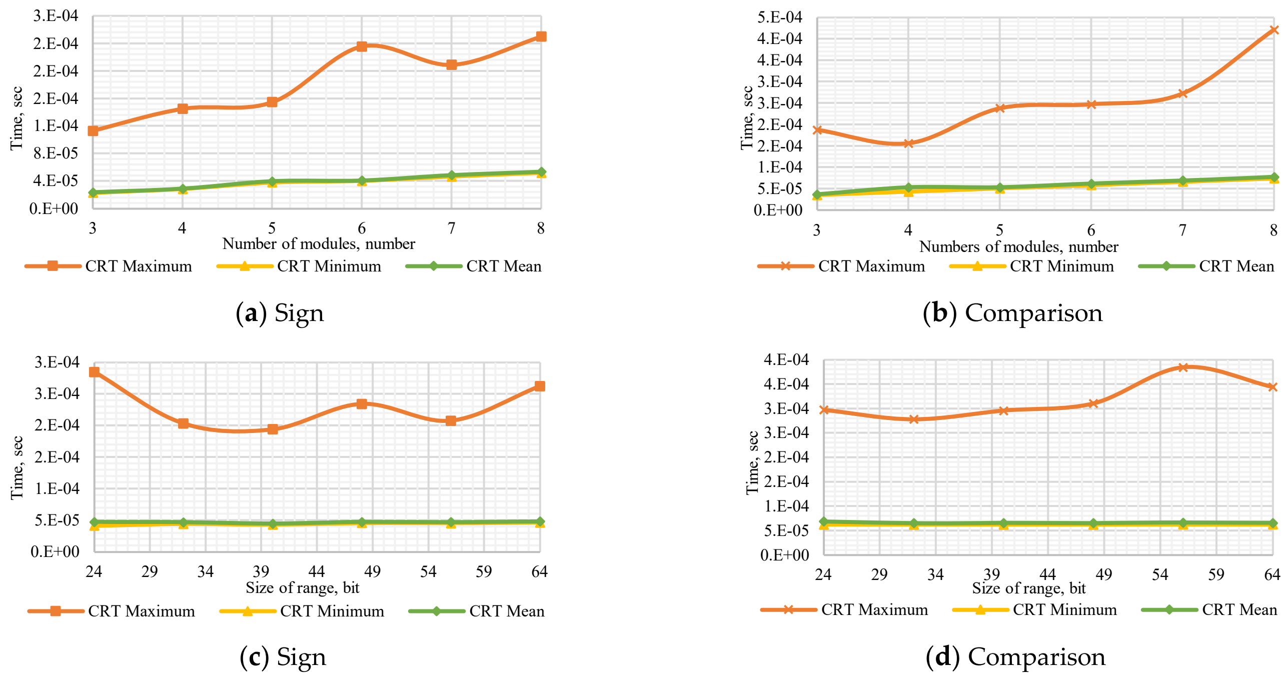


Figure A1. Performance of CRT method to (**a**) determine the number's sign with a dynamic number of modules p, (**b**) compare dynamic numbers of modules p, (**c**) determine the number's sign with a dynamic bit width, and (**d**) compare numbers with dynamic bit width.

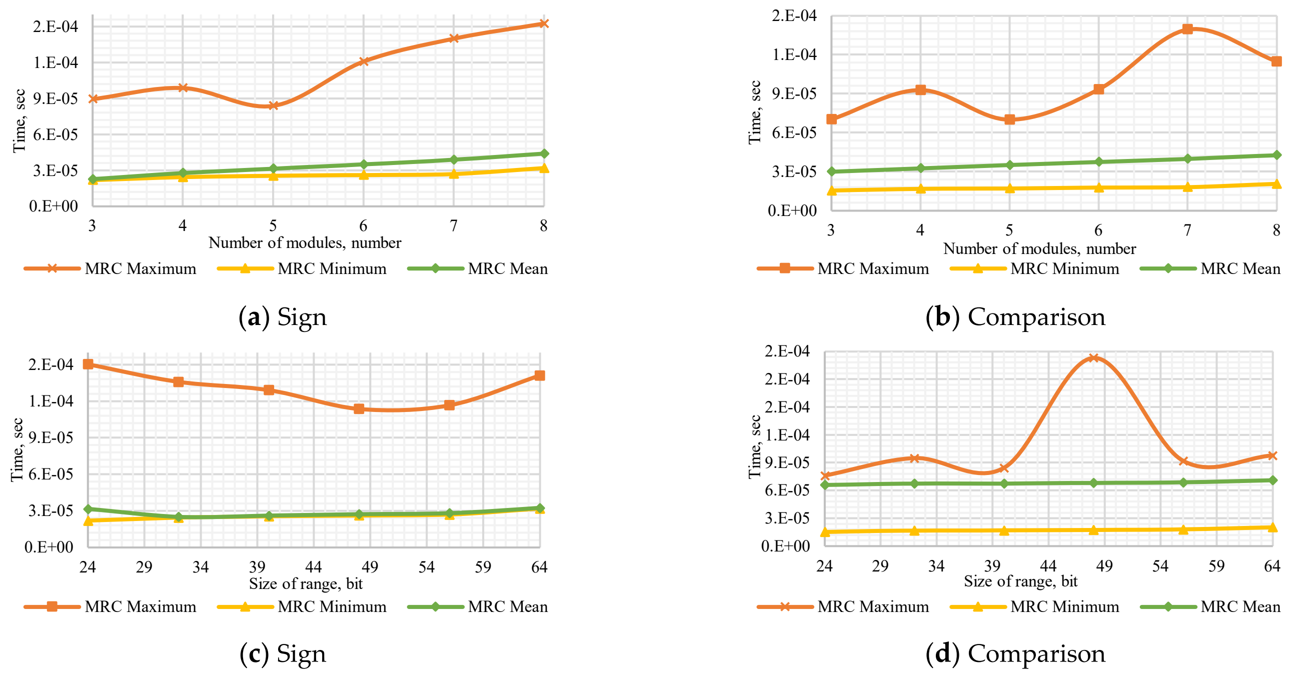


Figure A2. Performance of MRC method to (**a**) determine the number's sign with a dynamic number of modules p, (**b**) compare dynamic number of modules p, (**c**) determine the number's sign with a dynamic bit width, and (**d**) compare numbers with dynamic bit width.

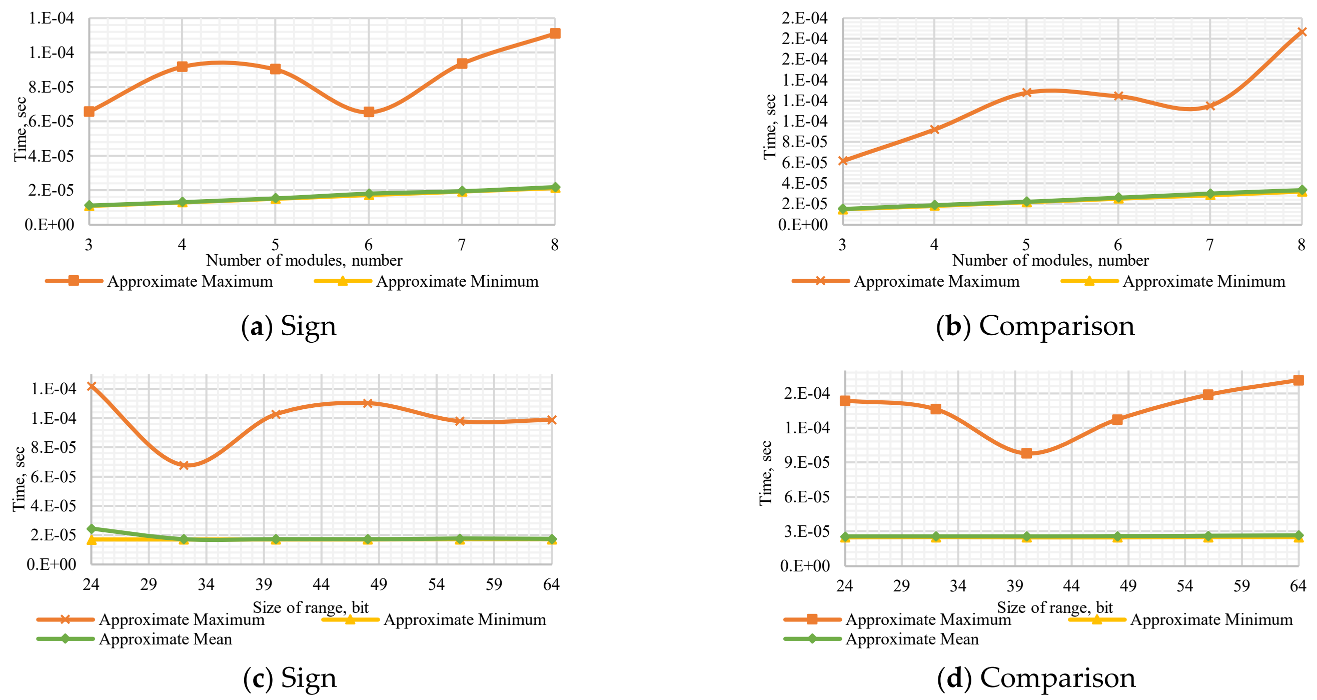


Figure A3. Performance of the Approximate Method to (**a**) determine the number's sign with a dynamic number of modules p, (**b**) compare the dynamic number of modules p, (**c**) determine the number's sign with a dynamic bit width, and (**d**) compare numbers with dynamic bit width.

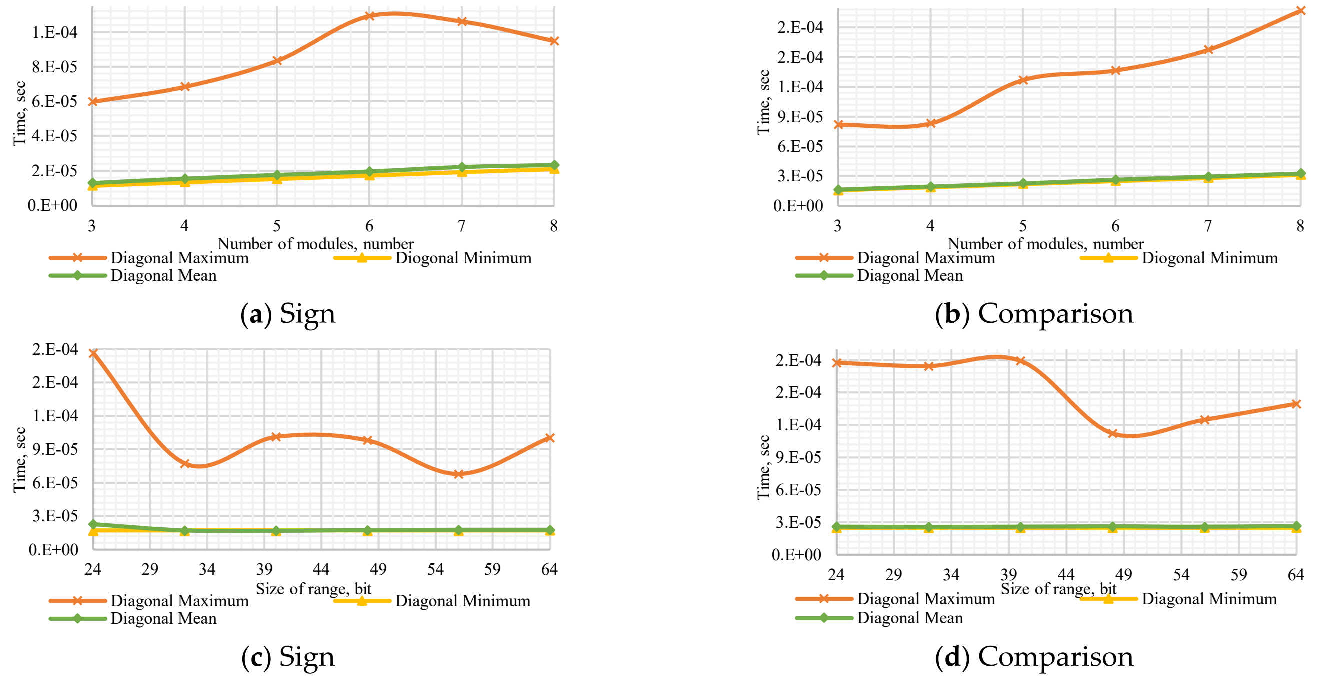


Figure A4. Performance of the Diagonal Method to (**a**) determine the number's sign with a dynamic number of modules p, (**b**) compare dynamic numbers of modules p, (**c**) determine the number's sign with a dynamic bit width, and (**d**) compare numbers with dynamic bit width.

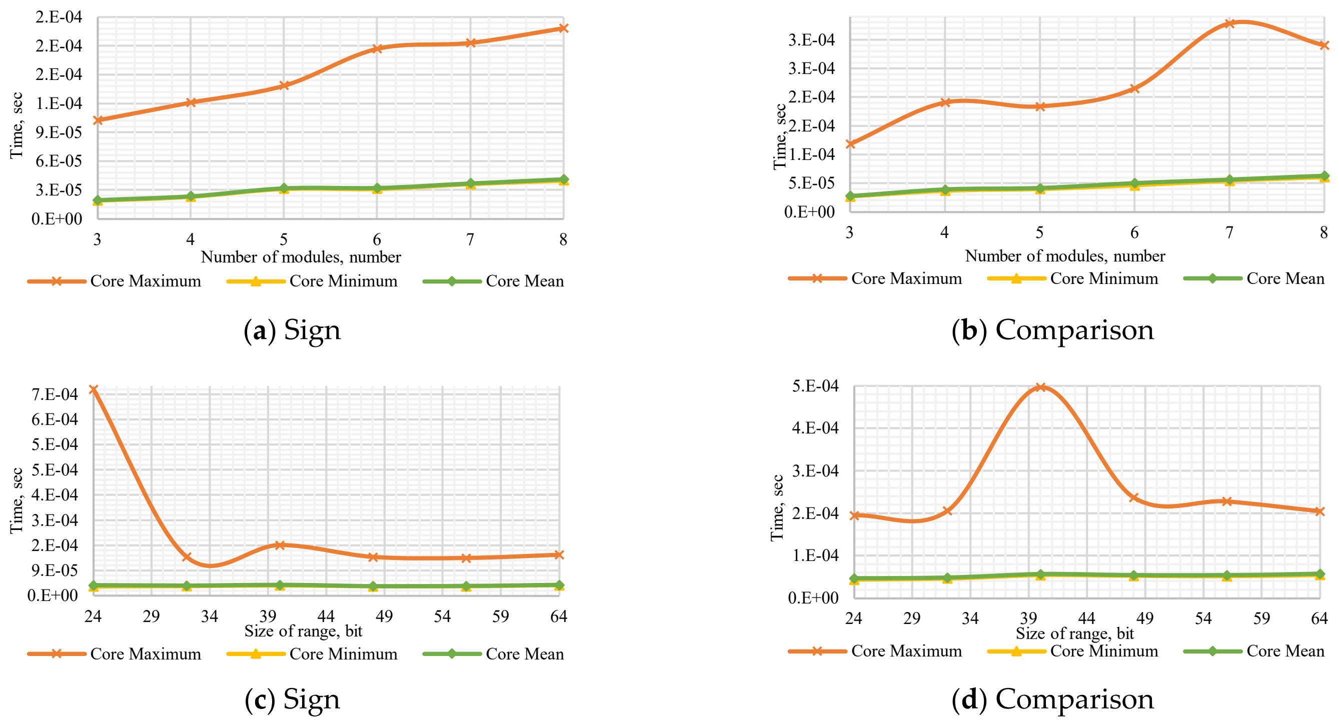


Figure A5. Performance of the Core Method to (**a**) determine the number's sign with a dynamic number of modules p, (**b**) compare dynamic numbers of modules p, (**c**) determine the number's sign with a dynamic bit width, and (**d**) compare numbers with dynamic bit width.

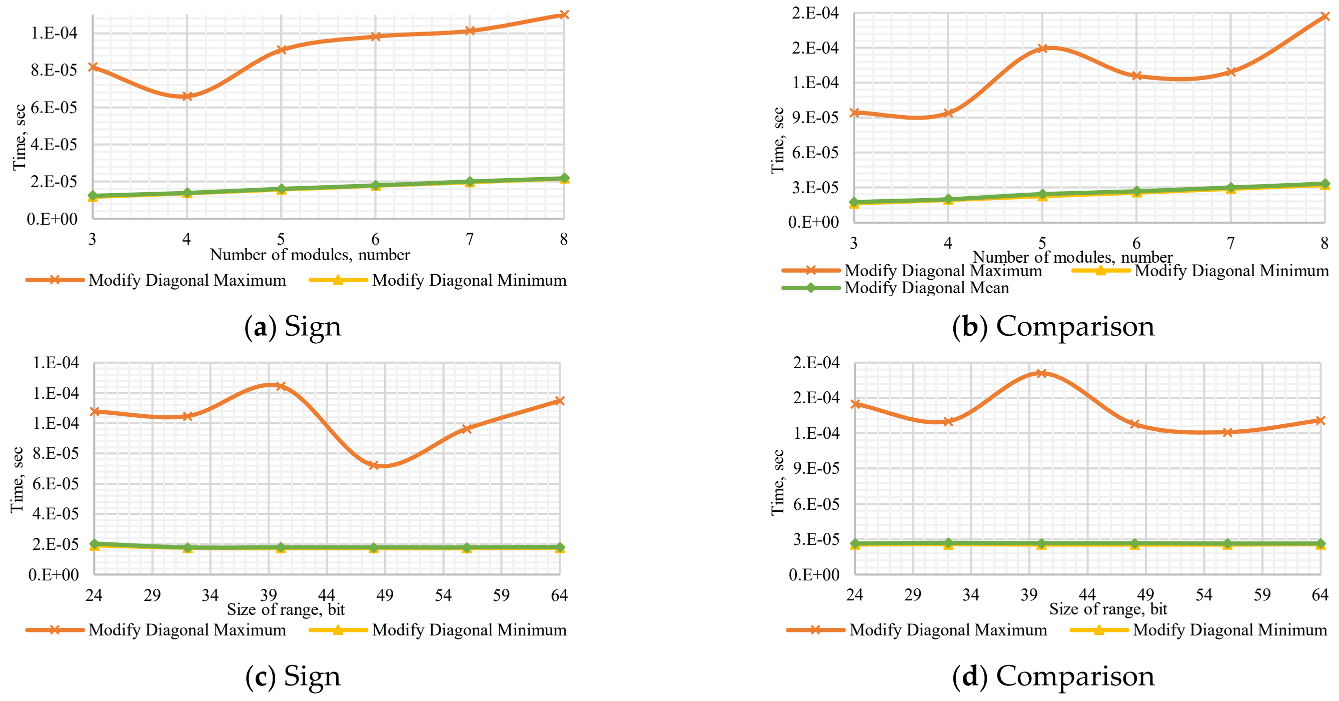


Figure A6. Performance of the Modify Diagonal Method to (**a**) determine the number's sign with a dynamic number of modules p, (**b**) compare dynamic numbers of modules p, (**c**) determine the number's sign with a dynamic bit width, and (**d**) compare numbers with dynamic bit width.

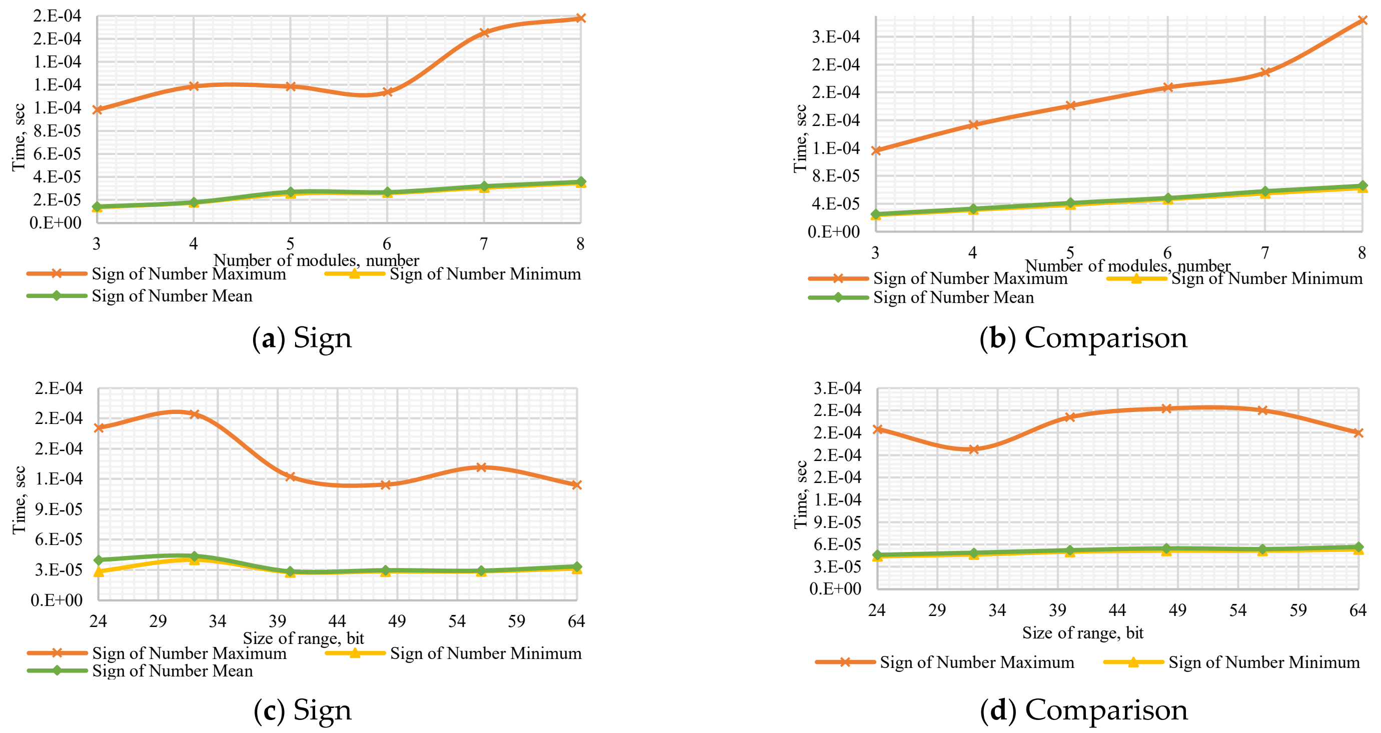


Figure A7. Performance of the Sign of Number Method to (**a**) determine the number's sign with a dynamic number of modules p, (**b**) compare dynamic numbers of modules p, (**c**) determine the number's sign with a dynamic bit width, and (**d**) compare numbers with dynamic bit width.

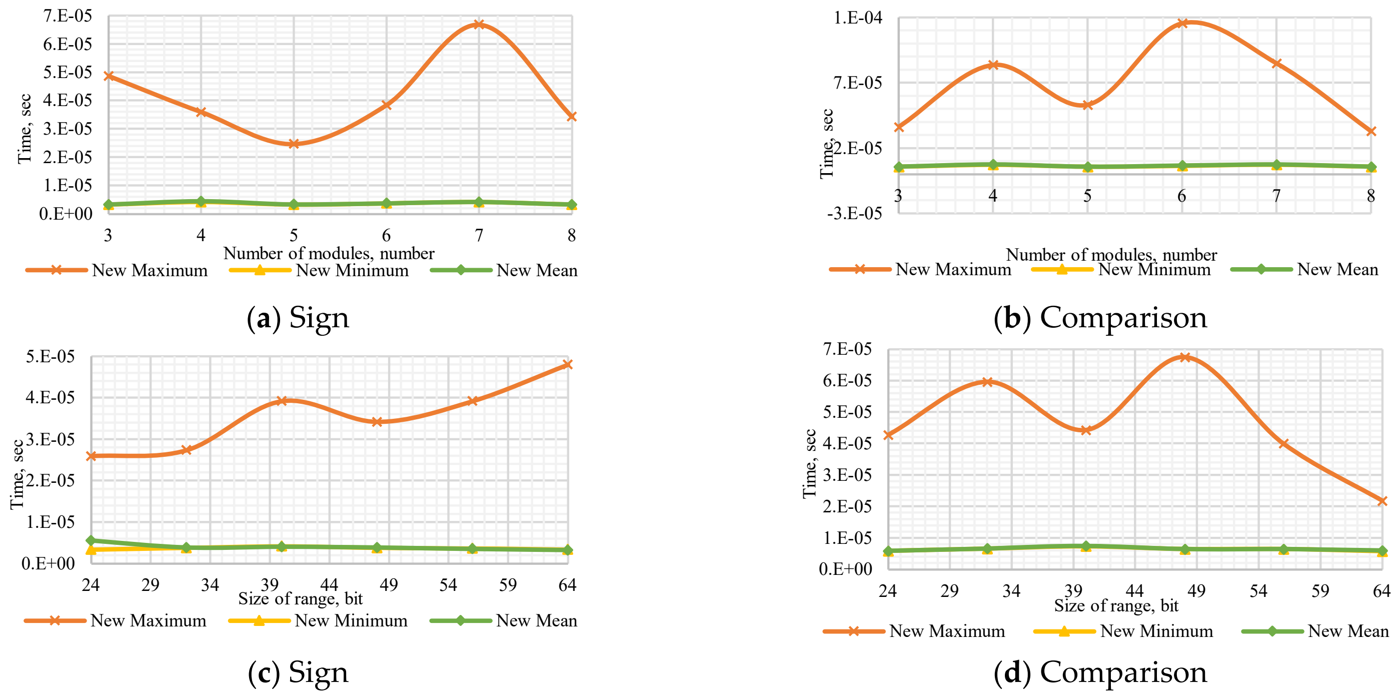


Figure A8. Performance of the DSN-EN method to (**a**) determine the number's sign with a dynamic number of modules p, (**b**) compare dynamic numbers of modules p, (**c**) determine the number's sign with a dynamic bit width, and (**d**) compare numbers with dynamic bit width.

References

1. Pirlo, G.; Impedovo, D. A new class of monotone functions of the residue number system. *Int. J. Math. Models Methods Appl. Sci.* **2013**, *7*, 803–809.
2. Piestrak, S.J. A note on RNS architectures for the implementation of the diagonal function. *Inf. Process. Lett.* **2015**, *115*, 453–457. [CrossRef]
3. Dimauro, G.; Impedovo, S.; Pirlo, G. A new technique for fast number comparison in the residue number system. *IEEE Trans. Comput.* **1993**, *42*, 608–612. [CrossRef]
4. Babenko, M.; Deryabin, M.; Piestrak, S.; Patronik, P.; Chervyakov, N.; Tchernykh, A.; Avetisyan, A. RNS Number Comparator Based on a Modified Diagonal Function. *Electronics* **2020**, *9*, 1784. [CrossRef]
5. Van Vu, T. Efficient implementations of the Chinese remainder theorem for sign detection and residue decoding. *IEEE Trans. Comput.* **1985**, *100*, 646–651. [CrossRef]
6. Babenko, M.G.; Tchernykh, A.N.; Chervyakov, N.I.; Kuchukov, V.A.; Miranda-López, V.; Rivera-Rodriguez, R.; Du, Z. Efficient number comparison in the residue number system based on positional characteristics. *Proc. Inst. Syst. Program. RAS* **2019**, *31*, 187–202. [CrossRef]
7. Babenko, M.; Piestrak, S.J.; Chervyakov, N.; Deryabin, M. The Study of Monotonic Core Functions and Their Use to Build RNS Number Comparators. *Electronics* **2021**, *10*, 1041. [CrossRef]
8. Isupov, K.; Knyazkov, V.; Kuvaev, A. Design and implementation of multiple-precision BLAS Level 1 functions for graphics processing units. *J. Parallel Distrib. Comput.* **2020**, *140*, 25–36. [CrossRef]
9. Isupov, K. Using Floating-Point Intervals for Non-Modular Computations in Residue Number System. *IEEE Access* **2020**, *8*, 58603–58619. [CrossRef]
10. Gentry, C. *A Fully Homomorphic Encryption Scheme*; Stanford University: Stanford, CA, USA, 2009.
11. Pulido-Gaytan, B.; Tchernykh, A.; Cortés-Mendoza, J.M.; Babenko, M.; Radchenko, G.; Avetisyan, A.; Drozdov, A.Y. Privacy-preserving neural networks with Homomorphic encryption: Challenges and opportunities. *Peer-Peer Netw. Appl.* **2021**, *14*, 1666–1691. [CrossRef]
12. Cortés-Mendoza, J.M.; Tchernykh, A.; Babenko, M.; Pulido-Gaytán, L.B.; Radchenko, G.; Leprevost, F.; Wang, X.; Avetisyan, A. Privacy-preserving logistic regression as a cloud service based on residue number system. In *Russian Supercomputing Days*; Springer Cham: Berlin/Heidelberg, Germany, 2020.
13. Kamara, S.; Kristin, L. Cryptographic Cloud Storage. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2010.
14. Kucherov, N.; Kuchukova, E.; Tchernykh, A.; Kuchukov, V.; Babenko, M. Towards Optimizing Cloud Computing Using Residue Number System. *J. Phys. Conf. Ser.* **2021**, *1715*, 012052. [CrossRef]
15. Babenko, M.; Tchernykh, A.; Golimblevskaia, E.; Pulido-Gaytan, L.B.; Avetisyan, A. Homomorphic Comparison Methods: Technologies, Challenges, and Opportunities. In Proceedings of the 2020 International Conference Engineering and Telecommunication (En&T), IEEE, Dolgoprudny, Russia, 25–26 November 2020.
16. Babenko, M.; Tchernykh, A.; Pulido-Gaytan, B.; Golimblevskaia, E.; Cortés-Mendoza, J.M.; Avetisyan, A. Experimental Evaluation of Homomorphic Comparison Methods. In Proceedings of the 2020 Ivannikov Ispras Open Conference (ISPRAS), Moscow, Russia, 10–11 December 2020.
17. Lee, Y.; Lee, J.-W.; Kim, Y.-S.; No, J.-S. Near-optimal polynomial for modulus reduction using l2-norm for approximate homomorphic encryption. *IEEE Access* **2020**, *8*, 144321–144330. [CrossRef]
18. Chase, M.; Chen, H.; Ding, J.; Goldwasser, S.; Gorbunov, S.; Hoffstein, J.; Lauter, K.; Lokam, S.; Moody, D.; Morrison, T.; et al. Security of homomorphic encryption. *HomomorphicEncryption. Org. Redmond WA. Tech. Rep.* **2017**.
19. Shiryaev, E.; Golimblevskaia, E.; Babenko, M.; Tchernykh, A.; Pulido-Gaytan, B. Improvement of the Approximate Method for the Comparison Operation in the RNS. In Proceedings of the 2020 International Conference Engineering and Telecommunication (En&T), IEEE, Dolgoprudny, Russia, 25–26 November 2020.
20. Babenko, M.; Tchernykh, A.; Chervyakov, N.; Kuchukov, V.; Miranda-López, V.; Rivera-Rodriguez, R.; Du, Z.; Talbi, E.-G. Positional Characteristics for Efficient Number Comparison over the Homomorphic Encryption. *Program. Comput. Softw.* **2019**, *45*, 532–543. [CrossRef]
21. Pulido-Gaytan, L.B.; Tchernykh, A.; Cortés-Mendoza, J.M.; Babenko, M.; Radchenko, G. A Survey on Privacy-Preserving Machine Learning with Fully Homomorphic Encryption. In *Communications in Computer and Information Science*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 115–129. [CrossRef]
22. Garner, H.L. The residue number system. Presented at the Western Joint Computer Conference (IRE-AIEE-ACM '59 (Western)), San Francisco, CA, USA, 3–5 March 1959. [CrossRef]
23. Pei, D.; Arto, S.; Cunsheng, D. *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*; World Scientific: Singapore, 1996.
24. Yassine, H.M.; Moore, W.R. Improved mixed-radix conversion for residue number system architectures. *IEE Proc. G Circuits Devices Syst.* **1991**, *138*, 120–124. [CrossRef]
25. Chervyakov, N.I.; Molahosseini, A.S.; Lyakhov, P.A.; Babenko, M.G.; Deryabin, M.A. Residue-to-binary conversion for general moduli sets based on approximate Chinese remainder theorem. *Int. J. Comput. Math.* **2016**, *94*, 1833–1849. [CrossRef]
26. Burgess, N. Scaled and unscaled residue number system to binary conversion techniques using the core function. In Proceedings of the 13th IEEE Sympsoium on Computer Arithmetic, Asilomar, CA, USA, 6–9 July 1997.

27. Miller, D.D.; Altschul, R.E.; King, J.R.; Polky, J.N. Analysis of the Residue Class Core Function of Akushskii, Burcev, and Pak. In *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*; IEEE Publications: Manhattan, NY, USA, 1986; pp. 390–401.
28. Gonnella, J. The application of core functions to residue number systems. *IEEE Trans. Signal Processing* **1991**, *39*, 69–75. [CrossRef]
29. Chervyakov, N.I.; Babenko, M.G.; Deryabin, M.A.; Nazarov, A.S.; Shabalina, M.N. Computation of Positional Characteristics of Numbers in RNS Based on Approximate Method. In Proceedings of the 2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), St. Petersburg, Russia, 2–3 February 2016.