Efficient Algorithm for RNS Implementation of RSA

I.R. Fadulilahi
Department of Computer Science
University for Development studies,
Box 1350, Tamale, Ghana

E.K. Bankas
Department of Computer Science
University for Development Studies,
Box 1350, Tamale, Ghana

J.B.A.K. Ansuura
School of Computer Science and Engineering
University of Electronic Science and Technology of China,
611731, Chengdu, China

ABSTRACT
In this paper, we present an algorithm for Residue Number System (RNS) implementation of RSA cryptography based on an existing RNS division algorithm. The proposed algorithm and that of the state of the art were written in C++ programming language to compare their efficiency with respect to execution time. Experimental results show that our algorithm can encrypt and decrypt text without loss of inherent information and faster than the state of the art. It also offers firm resistance to Brute-force and key sensitivity attacks. Considering the moduli-set {2, 3, 5} experimental results shows that, our proposed algorithm is 7.29% and 15.51%, faster than the state of the art algorithm for integer and non-integer quotients respectively. Also, for the moduli-set {7, 9, 11}, our algorithm is as well 11.29% and 10.36% faster than that of the state of the art algorithm for integer and non-integer quotient respectively. We carried out an error analysis of the experimental results at 95 degrees significance level.

Keywords
RSA, RNS, Cryptography, key, algorithm

1. INTRODUCTION
Individuals have secrets that need protection; these secrets appear in areas like medical files, bank statements, paycheck, investment portfolio and credit card bills. Others include social security numbers, credit card numbers, bank account numbers etc. Corporations also have secrets, these includes; strategic reports, sales forecasts, technical product details, research results, personnel files, and so on. [16]. In the past before the advent of computers, security was simply a matter of locking of doors or storing files in locked filling cabinet or safe. Today files are stored in computer databases as well as file cabinets.

Hard-drives and floppy disks hold many of our secrets. In the beginning the best way was to provide security to these data through the Operating System (OS), by locking it using a password.

However, various attacks on passwords have rendered this mode of security vulnerable and attacks bypass the OS. For your secrets to be secured, it may be necessary to add protection not provided by your computer (OS). One of the most important tools for protecting data is cryptography [13].

Cryptography has many advantages, some of which include; adding security to the process of authenticating people identity, Improves privacy, such that, no one can break into files to read your sensitive data. Improved data integrity, which refers to a mechanism that tells us when something has been altered. Also by applying the practice of authentication, you can verify identities.

In recent times, there has been a vigorous and continuous search for improving computer performance [1]. Researchers are coming out with new ideas and technologies to make the computer more efficient. The main task of a computer is computing which deals with numbers all the time. Some examples of number systems are binary number systems, decimal number systems, [1]. Weighted Number System (WNS) and Residue Number System (RNS) Binary and decimal number systems, intrinsically limit the performance of arithmetic units and processors built based on them. Because of this limitation in Weighted Number System (WNS), RNS has many advantages of computing large numbers in computers over WNS. These include carry-free addition and borrow free subtraction, which are the challenges to binary and decimal number system, because in RNS a number is represented by the residues of all moduli, and the arithmetic can be performed on each modulus independently. Therefore RNS offers the properties of parallelism [17].

Even though RNS has many advantages over WNS in terms of encoding large numbers into a set of smaller numbers to speed up computations, the following are time-consuming operations in RNS which affect the wide spread application of RNS in areas like cryptography; overflow detection, sign detection, magnitude comparison and division. Among them, division has modular operations application as can be found in cryptography [17]. Currently, fast hardware implementations of RSA cryptosystem is under study while confidentiality and security requirements are becoming more and more important. In view of this emerging problem of digital security, cryptographers keep increasing the key-length. Recently, it is assumed that a 1024-bit key-length makes a reasonable choice for the cryptography popularly known as RSA, and current analysis predict that 2048-bit or 4096-bit key will become the standard in a near future [16].

The ability to perform fast arithmetic on large integers is still a major issue for the implementation of public key cryptography and digital signature, particularly from the hardware design point of view [16].

In traditional cryptography, encryption and decryption operations are performed with the same key, that is, symmetric key cryptography. This means that the party encrypting the data and the party decrypting it need to share the same decryption key. If two parties already share a secret key, they could easily distribute new keys to each other by encrypting them with prior keys. From symmetric key encryption, researchers continue to build knowledge towards unsymmetrical key encryption. [4] Suggested that, encryption and decryption could be done with a pair of different keys. The decryption key would be kept secret, and the encryption key could be made public. This concept was called public-key cryptography. Every computer can use that encryption key to

protect data sent to the site. However, only the site has the corresponding decryption key that can decrypt the data and the concept of digital signatures were also introduced. [4],

Their key agreement is confronted with the problem of discrete logarithms and integer factorization. In 1977, a public key cryptosystem was invented by [4] called the RSA public key cryptography. However, only integers are encrypted in RSA. Fast RNS division algorithms for fixed divisors with application to RSA encryption, was first written by [1], however their algorithm was restricted to only division with integer quotient. Due to the fact that it was iterative, it was time consuming. In [16], a full RNS implementation which was based on the Chinese Remainder Theorem (CRT) and Montgomery Multiplication (MM), and base extension are presented. The main drawback of CRT emerges from the required modulo-M operation, which is time consuming and rather expensive in terms of area and energy consumption for large M [7]. In 2013, a division algorithm was presented in [4] without using CRT/MRC or MM. They used the parity checking technique and highest powers of two to perform division in RNS, however it is also iterative. Meanwhile, a non-iterative and pure RNS division algorithm has been presented in [10] and this solved the looping problem and restriction to integer quotient. In this paper, we propose an efficient RNS implementation of RSA cryptography based on based on a non-iterative and pure RNS division algorithm by Mansoureh and Mohammed (2012). In fact, this algorithm avoids overall loop and supports all numbers in the range as denominator. The remaining part of this paper is as follows: In Section 2, RNS division algorithm is presented with emphasis on the one proposed by Mansoureh and Mohammad (2012), sections 3, 4, 5, and 6 presents RSA Cryptography, the proposed algorithm and performance analysis of the proposed algorithm and the conclusions respectively.

2. DIVISION IN RNS

Division is one of the main obstacles that discourage the use of RNS. In RNS representation, division is not a simple operation. The analogy between division in conventional representation and RNS representation does not hold.

In conventional representation, we represent division as follows:

$$\frac{x}{y} = q \quad \text{..... (1)}$$

This can be written as

$$y * q = x \quad \text{..... (2)}$$

Where q, is the quotient

In RNS, the congruence:

$$y * q = x mod(m) \quad \text{..... .(3)}$$

Multiplying both sides by the multiplicative inverse of y, we can write:

$$q = x * y^{-1} mod(m) \quad \text{..... (5)}$$

The Equation $\frac{x}{y} = q$ is equivalent to

$$q = x \times y^{-1} \bmod m$$

Only if it has an integer value, otherwise, multiplying by the multiplicative inverse in RNS representation will not be equivalent to division in conventional representation

Example 2.1
Consider an RNS with m=7, we want to compute the following quotient:

$$a)\ \frac{6}{2}, \qquad b)\ \frac{6}{4}$$

In the first case (a)

$$\frac{6}{2} = q = 6 \times 2^{-1}(mod7) = 3$$

This is equivalent to division in conventional representation.

We notice in part (b), that division in RNS is not equivalent to that in conventional representation when the quotient is a non-integer value. Due to this fact, division in RNS is usually done by converting the residues to conventional representation, performing the division, and then converting back to RNS representation. Tedious and complex conversion steps result in undesired overhead. This is one of the main drawbacks of RNS representation.

$$\frac{6}{4} = q = 6 \times 4^{-1}(mod7) = 5$$

However, in cryptography this could serves as an advantage to adding security to cryptosystem.

Moreover, many other algorithms for division in RNS are presented. Some of these iterative algorithms work by subtracting denominator from numerator in a major loop, until numerator gets less than denominator. Quotient is equal to the number of iterations of this major loop. Some of them use Newton iteration to compute reciprocal and then compute quotient [12], [13].

Another common way for division is using the definition of division. In this algorithm, first the position of the most significant non-zero bit in the divisor and dividend is determined, then, according to the difference between these two positions, divisor is shifted to the left and is subtracted from dividend. These actions are repeated in a major loop until the result is smaller than divisor.

In some other methods for dividing *X* by Y, first the proper *2k* is detected such that $Y.2k \leq X \leq Y.2k+1$. In the next iterations, these two margins varied until quotient obtained. [9].

There is another method in which, instead of dividing two proposed numbers, X and Y, two different numbers which have the same ratio and are less than X and Y, are chosen. For doing this, some new moduli are introduced, and at last, in a major loop, the division result is calculated [14]. [5].

From the above, it is clear that these algorithms have three major deficiencies:

1. All of them have an overall loop which increases the complexity and delay of the algorithm.
2. Some of these methods exclude some numbers in the range of acceptable inputs as a denominator in division operation.
3. They have some operations in the binary or mixed radix system or uses a lookup Table to perform an RNS division.

In order to solve problems one and two, a non-iterative division algorithm was proposed by [11].

2.1 Pure RNS Division Algorithm by Mansoureh and Mohammad (2012)

Input: $(x_1, x_2 \ldots\ldots\ldots x_L)$, $Y = (y_1, y_2 \ldots \ldots \ldots y_L)$
Input $m_1, m_2 \ldots \ldots m_L$ the modulo set
Output: K=X/Y and R=(X mod Y) in RNS, Condition: all moduli $m_1, m_2 \ldots \ldots m_L$ are relatively prime numbers but 2.

1) Calculate Y^{-1}= multiplicative inverse of Y
2) Calculate R = X mod Y
3) Calculate K= (X – R). Y-1
4) If any, $y_i = 0$, then set

$k_i = k_n mod m_i$ In which $k_n \neq 0$

End.

3. THE RSA CRYPTOSYSTEM

This cryptosystem uses computations in $\mathbb{Z}_n$ where n is the product of two distinct odd primes p and q. for such an integer n, note that

$\emptyset_{(n)} = (p-1)(q-1)$…… (6)

Let

$n = pq$ Where p and q are primes

Let
$\mathcal{P} = \mathbb{Z}_n$ And define
$\mathcal{K} = \{(n, p, q, a, b): ab \equiv 1(mod)\Phi_{(n)}\}$…… (7)
For

$$\mathcal{K} = (n, p, q, a \; and \; b)$$

$e_k(x) = x^b mod(n)$…….. (8)

$d_k(y) = y^a mod(n)$…… .(9)

$x, y \in \mathbb{Z}_n$
The values n and b compose the public key, and the values (p, q and a) form the private key.

4. THE PROPOSED ALGORITHM

We propose an algorithm based on Mansoureh and Mohammed (2012) pure RNS division algorithm. Two different moduli sets were considered that is,$\{2^n - 2, 2^n - 1, 2^n + 1\}$ for n =2 [11], we had (2, 3, 5) and $\{2^n - 1, 2^n + 1, 2^n + 3\}$ for n = 3 we had (7, 9,11). The dynamic ranges are 29 and 692 respectively. We then did our analysis on integer and non-integer quotients for the range of values for X and Y within 29 and also for 692. In both cases we run several examples for non-integer and integer quotients regarding improper fraction. From Equation (8)

Let
$M = \left(\frac{X}{Y}\right)$ Where M is the quotient, X is the dividend and Y is the divisor. Thus modified as:

$e_k\left[\frac{X}{Y}\right] = \left[\frac{X}{Y}\right]^b mod(n)$…… (10)

From equation (9)
Let $Q = \left(\frac{D}{C}\right)$ where Q is the cipher text, D is the dividend and C is the divisor

Thus becomes

$d_k\left(\frac{D}{C}\right) = \left[\frac{D}{C}\right]^a mod(n)$ ..... (11).

This algorithm maintained the number of keys, both public keys and private keys.

Thus: (n, b) composed the public keys and (p, q, a) forms the secret keys.

For the Moduli sets

$\{\mathbf{2^n - 2, 2^n - 1, 2^n + 1}\}$,

For n=2; we have (2, 3, 5) and given

$\{\mathbf{2^n - 1, 2^n + 1, 2^n + 3}\}$,
For n=3; we have (7, 9, 11)

We consider the following assumptions:

4.1 Assumptions

The following are assumptions to our propose algorithm

- $X > Y > 0$
- No denominators or divisors should be a multiple of the chosen moduli set. Else the algorithm will break down due to a zero multiplicative inverse in one of the modulus
- After going through the division phase of the algorithm, when the CRT display the decimal equivalent of the RNS say (k) and its greater than (n),then the expected plain text would be (k)n. that is $(\mathrm{k})\mathrm{n}.^b mod(n) = (\mathrm{k})^b mod(n)$

4.2 The Encryption

$e_k(M) = M^b mod(n)$ In the usual RSA encryption

Let $M = Q = \left(\frac{X}{Y}\right)$ where Q is the quotient, X is the dividend and Y is the divisor

1) Input $\{m_1, m_2 m_3\}$ the moduli set,
2) Input $X \; and \; Y \; \; express \; in \; RNS$
3) Input (n and b) the public keys
4) $e_k\left[\frac{X}{Y}\right] = \left[\frac{X}{Y}\right]^b mod(n)$
5) Calculate Y^{-1}= multiplicative inverse of Y
6) Calculate R = X mod Y
7) Calculate Q= (X – R). Y-1
8) If any, $y_i = 0$, then set $Q_i = Q_n mod m_i$, $Q_n \neq 0$.

Note: The Transformation$\left(\frac{X}{Y}\right) = (j, k, l)_{RNS} = Q$. Is based on Mansoureh and Mohammed (2012), and Q output will appears in RNS representation.

9) Using CRT convert $(j, k, l)_{RNS}$ to Q i.e. the decimal value
10) Hence $e_k(Q) = Q^b mod(n) = G$
11) $ELSE; \;\; e_k(j, k, l) = (j, k, l)^b mod(n)$
12) End the program.

4.2.1 The Decryption

$d_k(G); G^a \ mod(n) = Q = M = \frac{X}{Y} \ the\ plaintext$ In the usual RSA decryption

1) Input $\{m_1, m_2 m_3\}$ the moduli set,
2) Input (t, f, h)$_{RNS}$
3) Input (p, q and a) the secret keys
4) Using the CRT convert (t, f, h) to G the decimal value.
5) $d_k[G]^a mod(n) = Q.$ ELSE$d_k[t, f, h]^a = (j, k, l) = M = \frac{X}{Y}$
6) End the program.

The following example illustrates our algorithm.

$$\left[\frac{258}{17}\right]$$

For

$\mathcal{K} = (n, p, q, a, and\ b\ ) = (187,11,17,7, and\ 23\ )$

4.2.2 The Encryption Algorithm

Let $X = 578, Y = 17$

1) Input $\{7,9,11\}$ the moduli set
2) Input $X = 578 \xrightarrow{RNS} (4,2,6)\ and\ Y = 17 \xrightarrow{RNS} (3,8,6)$
3) Input (n and b) the public keys $n = 187, b = 7$7
4) $e_k \left[\frac{578}{17}\right] = \left[\frac{578}{17}\right]^7 mod(187)$
5) Calculate Y^{-1}= multiplicative inverse of Y Else If any, $y_i = 0$, then set $Q_i = Q_n mod m_i\ in\ which Q_n \neq 0$ i.e. 17^{-1} w.r.t (7, 9, 11) = (5, 8, 2)
6) Calculate R = X mod $_Y$=$(578)_{17}$=$0 \xrightarrow{RNS} (0,0,0)$
7) Calculate Q= (X – R). Y^{-1}=$[(4,2,6) - (0,0,0)].[(5,8,2)] = (6,7,1)$
8) Using CRT convert $(6,7,1)_{RNS}$ to (34) i.e. the decimal value
9) Hence $e_k(34) = 34^7 mod(187) = 34$
10) $ELSE;\ \ e_k(6,7,1) = (6,7,1)^{17} mod(187)$
11) Hence $34 \xrightarrow{RNS} (6,7,1,)$ is send as the encrypted massage where X and Y or Q is the plain text.
12) End the program.

4.2.3 The Decryption Algorithm

Let $G = \left(\frac{D}{C}\right)$ where G is the cipher text, D is the dividend and C is the divisor.

1) Input {7,9,11} the moduli set,
2) Input $(G)_{RNS}$=(6,9,1)
3) Using the CRT convert $(G)_{RNS}$=(6,7,1) to decimal 34
4) Input (p, q and a) = 23 the secret key
5) **$d_k[6,7,1]^{23} mod(187) = d_k[34]^{23} mod(187) = (6,7,1)$**
6) Using (CRT)transform $(6,7,1)_{RNS}$ to 34
7) Hence $Q = 34$ is decrypted
8) End the program.

5. PERFORMANCE ANALYSIS OF THE PROPOSED ALGORITHM

Security analysis was performed to test the effectiveness of the proposed algorithm to known RSA attacks (key space, and key sensitivity analysis). Experimental results showed that the proposed algorithm is highly secured against such attacks.

5.1 Key Space Analysis

The brute force attack is computationally infeasible for cryptosystems with sufficiently large key space. The proposed algorithm is a public key cryptosystem which has $\mathcal{K} = (n, p, q, a, and\ b\ )$

The proposed algorithm achieves an efficient coding process when the key space K is large, note that, n depends on p and q. however, we adapt 1024bits as our key space as with the DES. This gives us the combination of choices. $k = 2^n$.

$2^{1024} = 1.797693134862315907729305190789e + 308.$

Diffie and Hellman(1976) outlined a "brute force" attack on DES by "brute force" is meant that you try as many of the 2^{56} possible keys as you have to before decrypting the cipher-text into a sensible plaintext message. Our key space is$2^{1024} > 2^{56}$

5.2 Sensitivity Analysis

The greatest sensitivity analysis in our algorithm is that explained in example 2.3 (a, and b) respectively. In (a) division in RNS is equivalent to division in conventional representation.

Where as in part (b), we notice that division in RNS is not equivalent to that in conventional representation when the quotient is a non-integer value. This can enhance the sensitivity of the key. A good cryptosystem should be sensitive to secret keys. A slight change in the key value should lead to a significant change in either a plain text or a cipher text. The addition of the used of the moduli set also help in security of the algorithm. There must be moduli set to be agreed on by both parties. The parameters (p, q, n, a, and b) in our examples resulted in significant difference with the actual answers during our experiment. Table 1 below illustrates these changes between conventional division and RNS division.

5.3 Executing Time Analysis for the Moduli set (7, 9, 11)

This moduli set used larger dynamic range, which gives room for larger domain for values of X and Y. the average time for integer quotient in this scheme is **16822.3** microseconds approximately 17seconds and that of non-integer is **15228.1** microseconds approximately 15sec. Table 2 did the comparison between the average times for the two different moduli sets, and it was observed that they both have almost the same average time. We compared our algorithm to some state of the art algorithms, and it reveal that our algorithm has the best average time for both integer and non-integer quotient with respect the two different moduli sets used. Table 3 explained these.

Observed that when the moduli set is chosen such that it has a small dynamic range, the algorithm would be limited to just a few numbers qualified for encryption and thus, attackers can

easily break the algorithm. The encryption time for these moduli set is higher than the decryption time. The average time to do a complete encryption and decryption with respect to an integer quotient is 16822.3microseconds which is approximately 17seconds, and that for a non-integer quotient is 15228.1microseconds approximately15 seconds. Tables 2 illustrate these

5.3.1 Executing Time Analysis for the Moduli-set (2, 3, 5)

The freedom of number representation in RNS is limited by the Dynamic range which is dependent on the moduli set. It is observed that when the moduli set is chosen such that it has a small dynamic range, the algorithm would be limited to just a few numbers qualified for encryption and thus, attackers can easily break the algorithm. The encryption time for these moduli set is higher than the decryption time.

The average time to do a complete encryption and decryption with respect to an integer quotient is 16822.3 microseconds which is approximately 17 seconds, and that for a non-integer quotient is 15228.1 microseconds approximately 15 seconds.

5.4 Error Analysis of Our Proposed Algorithm

We did the error analysis on our proposed algorithm in terms of the average executing time. This was performed on the different moduli set for {2, 3, 5} and {7, 9, 11}. Experimental results revealed that the mean, standard deviation and standard error are very minimal. Table 4 shows these statistics. We assume a significant level of 0.05.

The confidence interval of the two different moduli sets are shown in table 5. In table 6, we illustrated the percentage difference between our proposed algorithm and that of the state of the art we did the comparison with. The results showed that, our proposed algorithm is 7.29%, 15.51%, 11.29% and10.36% better than that of the state of the art we did the comparison with in terms of the moduli sets and the type of quotient involved.

Table 1 Sensitivity Analysis of Integer and Non-Integer Quotient

Example	Type of quotient	Plaintext in conventional division	Plaintext in RNS	Modul i Set	Cipher text
6/4	Non-integrin	1.5	**19** $\xrightarrow{RNS}$ (**5**, **1**, **8**)	(7,9,11)	24 $\xrightarrow{RNS}$ (3,6,2)
6/2	Integer	3	**3** $\xrightarrow{RNS}$ (**3**, **3**, **3**)	(7,9,11)	33 $\xrightarrow{RNS}$ (5,6,0)
258/33	Non integrin	7.8181	**11** $\xrightarrow{RNS}$ (**4**, **2**, **0**)	(7,9,11)	16 $\xrightarrow{RNS}$ (2,7,5)

Table 2 Average Time complexity for moduli-sets (2, 3, 5) and (7, 9, 11) in microseconds

Type of quotient	Average time (7, 9, 11)	Average time (2, 3, 5)
Non integer	19500.6	15228.1
Integer	18933.0	16822.3

Table 3 Comparison of our average times to some state of the art Algorithm in microseconds

Type of algorithm	Average Time With Respect to Moduli Set	
Our algorithm	(2,3,5)	(7,9,11)
Non-integer quotient	15228.1	19500.6
Integer quotient	16822.3	18933.0
Chin, Y. H. & Behrooz,P. (1994)	(2,3,5)	(7,9,11)
Non-integer quotient	20816.9	24008.1
Integer quotient	19468.6	23753.9

Table 4 Error Analysis of our Proposed Algorithm

	N	Minimum	Maximum	Sum	Mean		Std. Deviation
	Statistic	Statistic	Statistic	Statistic	Statistic	Std. Error	Statistic
Delay for encryption {2,3,5}	10	2308	21011	147556	14755.60	1583.749	5008.255
Delay for decryption {2,3,5}	10	1395	22865	141438	14143.80	1662.706	5257.938
Delay for encryption {7,9,11}	10	13442	22902	183509	18350.90	1029.528	3255.655
Delay for decryption {7,9,11}	10	13391	29409	200827	20082.70	1346.445	4257.832
Valid N (listwise)	10						

Table 5 Confidence Interval for the Error in our proposed Algorithm

Moduli Set	Confidence Interval	
(2,3,5)	***Lower bound***	***Upper bound***
Encryption	14677.60	14833.60
Decryption	14063.88	14223.72
(7,9,11)	***Lower bound***	***Upper bound***
Encryption	18288.01	18413.79
Decryption	20010.78	20154.62

Table 6 Percentage Difference between the proposed Algorithm and Parhami

Quotient	Parhami		Proposed Algorithm		Percentage difference	
	(2,3,5)	(7,9,11)	**(2,3,5)**	(7,9,11)	**(2,3,5)**	(7,9,11)
Delay for Integer	**19468.6**	23753.9	**16822.3**	18933.0	**7.29%**	11.29%
Delay for Non-integer	**20816.9**	24008.1	**15228.1**	19500.6	**15.51%**	10.36%

6. CONCLUSIONS

In this paper, an efficient algorithm based on RNS division algorithm to implement RSA public key cryptography is proposed. The proposed scheme can absorb and control the emerging problem of growing key length in RSA. The proposal algorithm is general and can be applied to RSA cryptography. Fractions, decimals, and integers can all be encrypted and decrypted. The proposed algorithm performed relatively well comparing it to existing state of the art algorithms. Experimental results demonstrates that, the proposed scheme is 7.29% and 15.51%, faster than the state of the art algorithm for integer and non-integer quotients respectively with respect to the moduli set {2, 3 5}. Additionally, with the moduli-set {7, 9, 11}, the proposed algorithm is 11.29% and 10.36% faster than that of the state of the art for integer and non-inter quotients respectively. The proposed algorithm would work efficiently when: - you choose a larger dynamic range for larger domain, Non-integer quotient is highly secured and sensitive relative to integer quotient, and it is having the best time as well. It was observed that the algorithm is having the best average time for both integer and non-integer quotient relative to the state of the art algorithm that it was compared to. The error analysis was performed and we had a negligible error margin to the proposed algorithm.

This paper has made some gains. However, development can be looked at in the future in the following areas:-The hardware implementation of the proposed system, Analyzing the performance of the proposed techniques and other division algorithm technique, The moduli set could also be enhanced to increase the dynamic range relative to the key space, The proposed algorithm can be tested on other public key cryptosystem like the Elgamal, With the idea of the proposed algorithm scaling in RNS could also be exploited for possible application to RSA.

7. REFERENCES

[1] Behrooz. P. (1999). Computer Arithmetic Algorithm and Hardware Design. New York Oxford University Press.

[2] Chin-Chen, C. and Jen-Ho, Y. (2013).A Division Algorithm Using Bisection Method in Residue Number

[3] System. International Journal of Computer, Consumer and Control.2 (1): 59-66.

[4] Chin-Chen, C. and Yeu-Pong, L. (2006). A division algorithm for residue numbers, Applied Mathematics and Computation in Elsevier.

[5] Diffie and Hellman (1976).New Direction in Cryptography, IEEE Transactions on Information Theory archive vol. 22 Issues 6Th November.

[6] Hiasat and Zohdy (1997).Design and Implementation of an RNS Division Algorithm.IEEE.240-249.

[7] Hitz, M. and Kaltofen M. (1995).Integer division in residue number systems. IEEE Transactions on Computers. 44(8):983– 989.

[8] Hussein, E., Hasan, M. A. and Elmasry M. I. (1998).A low power algorithm for division in residue number system. IEEE.

[9] Jean-Claude, B. G. (1991). New Approach to Integer Division in Residue Number Systems, Proceedings of 10th IEEE symposium on Computer Arithmetic, Grenoble, France. 84–91.

[10] Laurent, I. and Jean-Claude, B. (2004). A Full RNS Implementation of RSA Transactions on computers. IEEE.53 (5):1-6.

[11] Mansoureh and Mohammed (2012). Non iterative RNS division Algorithm, IMECS Vol. I. ISSN;2078-0966 online

[12] Mi, L. (2004).Arithmetic and Logic in Computer Systems. John Wiley and Sons, Inc., Hoboken, New Jersey.

[13] Nobuhiro, T. and Teruki, I. (2002).Design of High-Speed RSA Encryption Processor Based on the Residue Table for Redundant Binary Numbers. Systems and Computers in Japan. 33(5):423-432.

[14] Omondi, A. and Premkumar, B., (2007). Residue Number System Theory and Implementation. Imperial College Press..

[15] Rivest, R., Shamir, A. and Adleman L. (1978).A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM .21(2):120–126.

[16] Steven Burnett and Stephen Paine (2004), RSA Security Official Guide to Cryptography, Keller Graduate School of Management of DeVry University Edition, ISBN 0-07-225494-7.

[17] Szabo, N. and Tanaka, R. (1967), Residue Arithmetic and Its Applications to Computer Technology. McGraw Hill, New York.

[18] Yang J. H. Chang C. C and Chen Y. Y (2004), A High Speed Division Algorithm in RNS using the Parity Checking Technique. International Journal of Computer Mathematics. 81(6).