2009 19th IEEE International Symposium on Computer Arithmetic

# Selected RNS Bases for Modular Multiplication

J.C. Bajard,
LIRMM, CNRS-Univ. Montpellier2,
France.
bajard@lirmm.fr

M. Kaihara,
EPFL Lausanne,
Switzerland.
marcelo.kaihara@epfl.ch

T. Plantard,
Univ. Wollongong,
Australia.
thomaspl@uow.edu.au

## Abstract

*The selection of the elements of the bases in an RNS modular multiplication method is crucial and has a great impact in the overall performance. This work proposes specific sets of optimal RNS moduli with elements of Hamming weight three whose inverses used in the MRS reconstruction have very small Hamming weight. This property is exploited in RNS bases conversions, to completely remove and replace the products by few additions/subtractions and shifts, reducing the time complexity of modular multiplication. These bases are specially crafted to computation with operands of sizes* $256$ *or more and are suitable for cryptographic applications such as the ECC protocols.*

## 1 Introduction

Since the introduction of Residue Number Systems (RNS) [23, 24] in computer science, a lot of works have been done. One of the first applications concerns Digital Signal Processing (DSP) for computing convolution products in filtering [22, 8]. The motivation originated in the need of a certain accuracy during the evaluation. In this context, RNS was found attractive for some part of the calculation, and finding efficient conversion methods from binary to RNS and from RNS to binary became a challenge. As the precision needed was not so huge, the RNS bases were restricted to few moduli [20, 18]. It has always been a topic of research with some interesting evolutions [7, 12]. A good study can be found in [10].

In public key cryptography as in Diffie-Hellman [11], RSA [21], Elliptic Curves Cryptography (ECC) [15, 14] and pairings, the problem settings are different. The numbers involved are huge and some operations as modular multiplication are intensively used. One of the most known algorithm for the modular multiplication which does not require any division in presented in [16]. This algorithm can be implemented in RNS using an auxiliary base [19], and was found efficient only for small integers [1]. Moreover, it is possible to realize full RNS cryptosystems [2].

The main interests of RNS are due to two facts: They are well adapted to parallel implementations on different platforms, such as in GPU [25], and they provide interesting low power architectures [6]. In parallel architectures, for low power devices, with applications to cryptography, the challenge is to find RNS bases where all the elements have the same number of bits and that their inverses, used in bases extensions, have advantageous properties. The works presented in the literature, as the one in [5], do not fulfill all these requirements. The selection of the bases presented in the current work is clearly devoted to architecture where multiplication is costly.

This paper is focused on cryptographic applications where the operands are huge numbers. We present RNS bases specially crafted to computation with operands of $256$ bits or more, suitable for applications such as the ECC protocols. We consider RNS representations where all the elements of the bases have the same number of digits; specifically, we focus on bases with $64$-bit elements with very small Hamming weight, allowing good implementation in software as well as in hardware. We show that it is possible to construct bases with elements with Hamming weight three and that these elements can be selected so that a conversion from one RNS basis to another one, via a Mixed Radix System, can be performed without any multiplication, thanks to the property of small Hamming weights of their inverses. The costs are given by the number of $k$-bits additions (e.g., $k = 64$), thus the cost of a multiplication by a constant is equivalent to the Hamming weight of this constant $k$-bits additions.

Our results are based on an exhaustive search of such bases, from which we withdraw some theoretical results about the criteria for selecting RNS bases. The organization of the paper is the following. Firstly, we introduce the context of this work giving some elements on RNS. Secondly, we give some elements for calculating the complexity based on Hamming weight of the operands that we used for the exhaustive search. Then, we present our experimental results of the exhaustive search of adapted bases. Finally, we ex-

1063-6889/09 $25.00 © 2009 IEEE
DOI 10.1109/ARITH.2009.20

plain some theoretical results and discussions inferred from our experiments.

## 2 Basic RNS operations

### 2.1 Introduction to RNS

We consider an RNS base $\mathcal{B}_n = \{m_1, ..., m_n\}$ (set of relatively prime numbers) with $M = \prod_{i=1}^{n} m_i$. An integer $X$, $X \in [0, M[$, is represented by its residues $(x_1, x_2, \ldots, x_n)$ (i.e., $x_i = X \bmod m_i$). The vector $(x_1, x_2, \ldots, x_n)$ is called the RNS representation of $X$ in the RNS base $\mathcal{B}_n$.

The CRT (Chinese Remainder Theorem) construction from the RNS representation to a classical representation is given by the formula:

$$X = \left| \sum_{i=1}^{n} \left| x_i \left| M_i \right|_{m_i}^{-1} \right|_{m_i} M_i \right|_M \quad (1)$$

where $M_i = \frac{M}{m_i}$, and $|M_i|_{m_i}^{-1}$ is the inverse of $M_i$ modulo $m_i$. The notations $|.|_M$ or $|.|_{m_i}$ mean respectively modulo $M$ and modulo $m_i$.

### 2.2 Conversions RNS $\leftrightarrow$ RNS or RNS $\leftrightarrow$ Binary

There are mainly two approaches for converting from RNS to a classical binary representation. One is based on explicitly applying the CRT, and the other uses as an intermediate representation the Mixed Radix System (MRS). The use of CRT for converting from RNS to a classical representation is not convenient due to the large sizes of the involved operands and the intermediate results. An RNS to RNS conversion via the formula (1) requires $n^2 + n$ modular word products (i.e., a multiplication modulo one element of the RNS basis). A priori, with the CRT construction, it seems difficult to obtain values for $|M_i|_{m_i}^{-1}$ and $M_i$ with particular forms that might improve the complexity of the computation. An idea using divide and conquer approach was proposed in [26]. However, intermediate values remain comparable in size to the case of MRS, and only the asymptotic complexity is improved with a specific architecture.

In contrast, conversion via the MRS provides better performance, as constant values involved in the computation are simple and can conveniently be represented to reduce a multiplication into few additions and shifts.

The MRS Representation of $X = (\dot{a}_1, \dot{a}_2, \dot{a}_3, \cdots, \dot{a}_n)$ is given by

$$\begin{aligned}
\dot{a}_1 &= x_1 \\
\dot{a}_2 &= \left| (x_2 - \dot{a}_1) m_{1,2}^{-1} \right|_{m_2} \\
\dot{a}_3 &= \left| (\left| (x_3 - \dot{a}_1) m_{1,3}^{-1} \right|_{m_3} - \dot{a}_2) m_{2,3}^{-1} \right|_{m_3} \\
\dot{a}_4 &= \left| (\left| (\left| (x_4 - \dot{a}_1) m_{1,4}^{-1} \right|_{m_4} - \dot{a}_2) m_{2,4}^{-1} \right|_{m_4} - \dot{a}_3) m_{3,4}^{-1} \right|_{m_4} \\
&\vdots \\
\dot{a}_n &= \left| (. \left| (\left| (x_n - \dot{a}_1) m_{1,n}^{-1} \right|_{m_n} - \dot{a}_2) m_{2,n}^{-1}) \right|_{m_n} - . - \dot{a}_{n-1}) m_{n-1,n}^{-1} \right|_{m_n}
\end{aligned} \quad (2)$$

where $m_{i,j}^{-1}$ is the inverse of $m_i$ modulo $m_j$

Then, the conversion from MRS to classical representation (or another RNS) is obtained from the formula $X = \dot{a}_1 + \dot{a}_2 m_1 + \dot{a}_3 m_1 m_2 + \cdots + \dot{a}_n m_1 \cdots m_{n-1}$ ( for RNS, this evaluation is done modulo each element of the new RNS base), or using the Horner's scheme:

$$X = \dot{a}_1 + m_1 \left[ \dot{a}_2 + m_2 \left[ \dot{a}_3 + \cdots + m_{n-2} \left[ \dot{a}_n \right] \cdots \right] \right] \quad (3)$$

A remarkable advantage of this method is that values of the numbers involved in the computation are bounded by a relatively small value. The conversion to a new RNS basis via the MRS can be obtained by applying reduction modulo each element of the new RNS basis.

### 2.3 Modular multiplication in RNS

Montgomery reduction scheme for calculating modular multiplication was introduced in [16]. Here, we consider the calculation of Montgomery modular multiplication in RNS. Let $A$ and $B$ be two large numbers represented in RNS with basis $\mathcal{B}_n = \{m_1, ..., m_n\}$. This means that the values of $A$ and $B$ are less than $M = \prod_{i=1}^{n} m_i$, the product of the elements of the RNS basis. We denote with $(a_1, ..., a_n)$ and $(b_1, ..., b_n)$ the respective RNS representations of $A$ and $B$.

As the result of a product requires twice the precision to be correctly represented, we consider an auxiliary base $\mathcal{B'}_n = (m'_1, ..., m'_n)$ where $M' = \prod_{i=1}^{n} m'_i$ is the product of the values of these elements. Although it is not mandatory, an equal number of elements is assumed for both bases. This provides regularity in the hardware architecture or the software implementation and simplifies the calculation of the complexity.

In the auxiliary basis, $A$ and $B$ are represented as $(a'_1, ..., a'_n)$ and $(b'_1, ..., b'_n)$ accordingly. We denote with $P$ the modulus and its representations in the two bases (the primary and the auxiliary) as $(p_1, ..., p_n)$ and $(p'_1, ..., p'_n)$ respectively. Given $P$, where $P < M < M'$, and $\gcd(P, M) = \gcd(P, M') = \gcd(M, M') = 1$, modular multiplication, that computes $A \times B \times M^{-1} \pmod P$, can be depicted as following:

1. Calculate the product $D$ of $A$ and $B$ in RNS in the two bases $\mathcal{B}_n$ and $\mathcal{B'}_n$: $D = A \times B$ (obtained by mul-

tiplying $d_i = |a_i \times b_i|_{m_i}$ and $d'_i = |a'_i \times b'_i|_{m'_i}$ for $i = 1, ..., n$).

2. Perform Montgomery modular reduction modulo $P$:

   (a) Evaluate $Q = \left|D \times |P|_M^{-1}\right|_M$ which is obtained in RNS with $q_i = \left|d_i \times |P|_{m_i}^{-1}\right|_{m_i}$ for $i = 1, ..., n$. These calculations are made only in $\mathcal{B}_n$ , thus $Q$ is obtained modulo $M$.

   (b) Extend the representation of $Q$ to the auxiliary base $\mathcal{B'}_n$.

   (c) Then, evaluate $R = (D - Q \times P) \times |M|_{M'}^{-1}$ in the base $\mathcal{B'}_n$, and compute $r'_i = \left|(d'_i - q'_i \times p'_i) \times |M|_{m'_i}^{-1}\right|_{m'_i}$. Thus, $R$ is known modulo $M'$.

   (d) Extend the representation of $R$ to the primary base $\mathcal{B}_n$.

We note that, if $A \times B < M \times P$ then $R < 2P$. Hence, it is required that $2P < M < M'$ (i.e.,$4P < M < M'$ if we use this result in other modular multiplications) .

## 3 Hamming weight and complexity

The first part of this section describes the Booth recoding system and some theoretical results which provides an upper bound of the number of trials in an exhaustive search. The second part describes the cost of the conversion from RNS to MRS and from MRS to RNS as a function of the Hamming weight (the number of non zero digits of the representation) of the elements of the bases and their inverses.

It is assumed hereafter that the elements of the RNS bases are represented as: $m_i = 2^k - c_i$ where $0 \leq c_i < 2^{k/2}$. Hence, we consider the Hamming weight of the $c_i$ instead of the one of the $m_i$.

### 3.1 Hamming weight of Booth recoded bases

The Booth recoding (or $NAF_2$) consists on rewriting series of consecutive ones (i.e. $0001111100$) in radix two representation as a power of two minus one (i.e. $0010000\overline{1}00$ where $\overline{1}$ represents minus one). After applying this recoding, each integer has a unique representation which does not contain two successive non-zero digits, and the number of digits is at most the number of digits of the radix two representation plus one.

The following two theorems provide the number of Booth recoded values with a Hamming weight lower than a certain bound $t$.

**Theorem 1** *The number of $k$-digit Booth recoded integers with Hamming weight less or equal to $t$, is equal to* $T_t = \sum_{i=1}^{t} 2^{i-1} \frac{(k-i-1)!}{(i-1)!(k-2i)!} + 2\sum_{i=1}^{t-1} 2^{i-1} \frac{(k-i-2)!}{(i-1)!(k-2i-1)!}$

Proof: Consider $k$ digits representations of even integers, and the alphabet $a = 0$, $b = 10$ and $c = \overline{1}0$.

**Lemma 1** *The number of even integers coded with $d$ digits, in the Booth recoded representation with an Hamming weight lower or equal to $t$ is equal to* $T_t = \sum_{i=1}^{t} 2^{i-1} \frac{(k-i-1)!}{(i-1)!(k-2i)!}$.

If the Booth recoded value contains $t$ non-zero digits (i.e. letters $b$ or $c$) then the number of letters $a$ is equal to $k - 2t$ and the total number of letters is $k - t$. As we consider $k$ binary digits representations of even integer, the most significant digit is fixed to $b$. Thus, the number of such integers is equal to $N_t = 2^{t-1} \frac{(k-t-1)!}{(t-1)!(k-2t)!}$.

Hence, the number of integers coded with $k$ digits, in the Booth coded representation with an Hamming weight lower or equal to $t$ is equal to : $T_t = \sum_{i=1}^{t} 2^{i-1} \frac{(k-i-1)!}{(i-1)!(k-2i)!}$

**Lemma 2** *Consider odd integers coded with $d$ digits, in Booth coded representation with an Hamming weight lower or equal to $t$. The number of such values is equal to* $T_t = 2\sum_{i=1}^{t-1} 2^{i-1} \frac{(k-i-2)!}{(i-1)!(k-2i-1)!}$.

If we consider odd integers, the least significant digit is fixed to 1 or $\overline{1}$. Thus, the number of values is equal to that of the even integers on $k - 1$ digits with $t - 1$ non-zero digits. □

Note that the Lemma 2 gives a bound for an exhaustive search of RNS bases giving a minimal Hamming weight. In fact, the elements of the RNS bases are co-prime numbers, therefore, only one element can be a power of two; the others are odd values.

In the sequel, we denote with $\omega(a)$ the Hamming weight of the Booth recoded representation of $a$.

### 3.2 Complexity of the conversion from RNS

The main operation in the conversion from RNS to MRS, given by equation (2), is the computation of $\left|(y - a_i) \times m_{i,j}^{-1}\right|_{m_j}$ with $0 \leq y < m_j$, $0 \leq a_i < m_i$ and $0 < m_{i,j}^{-1} < 0$.

We note that the condition $y - a_i < 0$ might occur; we then evaluate $\alpha = y - a_i + m_j$ and $\beta = y - a_i + 2m_j$. If $\beta \geq 2^k$ then $|y - a_i|_{m_j} = \alpha$, otherwise $|y - a_i|_{m_j} = \beta$ . We also note that, $|y - a_i|_{m_j} < 2^k$, and reduction modulo $m_j$ of the value $V = |y - a_i|_{m_j} \times m_{i,j}^{-1} < 2^{2k}$ is required.

This reduction is made in three steps:

1. The value $V$ is split into an upper part $V_h$ and a lower part $V_l$ such that $V = V_h 2^k + V_l$ and we compute $V' = V_h \times c_j < 2^{3k/2}$; it is reminded that $|2^k|_{m_j} = c_j$.
2. The same reduction is applied to $V' = V'_h 2^k + V'_l$, with $V'_h < 2^{k/2}$ to obtain $V'' = V'_h \times c_j < 2^k$.
3. Then, $W = V" + V'_l + V_l \equiv V \pmod{m_j}$ is obtained where $W = W_h 2^k + W_l$ is equal to 0, 1 or 2. We compute $A = W_h c_j + V''_l$ and $B = (W_h + 1)c_j + V''_l$, and represent $B = B_h 2^k + B_l$; if $B_h = 1$ then we retrieve $B_l$ otherwise we retrieve $A$.

We note that the third step of this reduction does not need a multiplication by $c_j$ due to the very small value of $W_h$. The reduction modulo $m_j$ of a value smaller than $2^{2k}$ requires $2\omega(c_j) + 2$ additions of $k$-bit words.

**Theorem 2** *The cost of a reduction modulo $m = 2^k - c_i$ with $0 \leq c_i < 2^{k/2}$, of a value smaller than $2^{2k}$ can be done with $2\omega(c_i) + 2$ additions of $k$-bit words.*

Hence, the cost of computation of $\left|(y - a_i) \times m_{i,j}^{-1}\right|_{m_j}$ is equivalent to $\omega(m_{i,j}^{-1}) + 2\omega(c_j) + 4$ additions of $k$-bit words. Each product in the series of equations in (2) is replaced by few additions/subtractions and shifts as a function of the Hamming weight of the inverses. A multiplication by an integer whose Hamming weight is $\omega$ can be performed in $\omega - 1$ additions.

Thus, the total cost of the conversion from RNS to MRS, given by equation (2), is equivalent, in terms of $k$-bit word additions, to:

$$Cost_{RNS-MRS} = \sum_{j=2}^{n} \sum_{i=1}^{j-1} \left(\omega(m_{i,j}^{-1}) + 2\omega(c_j) + 4\right) \quad (4)$$

The conversion from MRS to RNS is obtained using equation (3). The main operation in this conversion is the computation of $|a_i + m_i \times y|_{m'_j}$, with $0 \leq a_i \leq m_i - 1$ and $0 \leq y \leq m'_j - 1$.

Thus, the reduction modulo $m'_j$ of $V = a_i + (c'_j - c_i) \times y < 2^k$ is made following the previously described procedure (Theorem 2) with $2\omega(c'_j) + 2$ additions of $k$-bit words.

Hence, $|a_i + m_i \times y|_{m'_j}$ is evaluated with $\omega(c_i) + 2\omega(c'_j) + 2$ additions of $k$-bit words. The total cost required in the evaluation of the equation (3) is, in term of $k$-bits words additions:

$$Cost_{MRS-RNS} = \sum_{j=1}^{n} \sum_{i=1}^{n-1} \left(\omega(c_i) + 2\omega(c'_j) + 2\right) \quad (5)$$

The total cost of the RNS bases extension is (as number of $k$-bit word additions):

$$\begin{aligned} Cost_{RNS-RNS} = & \sum_{j=2}^{n} \sum_{i=1}^{j-1} \left(\omega(m_{i,j}^{-1}) + 2\omega(c_j) + 4\right) \\ & + \sum_{j=1}^{n} \sum_{i=1}^{n-1} \left(\omega(c_i) + 2\omega(c'_j) + 2\right) \end{aligned} \quad (6)$$

Now, it is also possible to count the complexity in time and space for a parallel implementation. Assuming $n$ $k$-bits word arithmetic units are available, the time complexity in this case is:

$$\begin{aligned} PTC_{RNS-RNS} = & \sum_{i=1}^{n-1} \max_{j=2,n;i<j} \left(\omega(m_{i,j}^{-1}) + 2\omega(c_j) + 4\right) \\ & + \sum_{i=1}^{n-1} \max_{j=1,n} \left(\omega(c_i) + 2\omega(c'_j) + 2\right) \end{aligned} \quad (7)$$

### 3.3 Complexity of the modular multiplication

The RNS Montgomery multiplication algorithm presented in section 2.3, requires two conversions, one RNS product on the two bases, one RNS product on the first basis, two RNS products and one addition on the second basis.

The operations $d_i = |a_i \times b_i|_{m_i}$ and $d'_i = |a'_i \times b'_i|_{m'_i}$ for $i = 1, ..., n$, in the RNS multiplication requires a total of $2n$ products of $k$-bit words and $2n$ reductions of values lower than $2^{2k}$ which represent $\sum_{i=1}^{n}(2\omega(c_i) + 2) + \sum_{j=1}^{n}(2\omega(c'_j) + 2)$ additions of $k$-bits words (Theorem 2).

For the calculation of $q_i = \left|c_i \times |P|_{m_i}^{-1}\right|_{m_i}$ for $i = 1, ..., n$, $n$ products of $k$-bits words and $n$ reductions are required which contribute with a cost of $\sum_{i=1}^{n}(2\omega(c_i) + 2)$ $k$-bits words additions.

The calculation of $r'_i = \left|(c'_i - q'_i \times p'_i) \times |M|_{m'_i}^{-1}\right|_{m'_i}$ for $i = 1, ..., n$, is obtained with $2n$ products of $k$-bits words and $2n$ reductions which represent $2\sum_{j=1}^{n}(2\omega(c'_j) + 2)$ additions of $k$-bits words (Theorem 2).

Thus, the total cost of the presented modular multiplication is:

$$Cost_{MMRNS} = 5nP_k + \left( 2\sum_{i=1}^{n}(2\omega(c_i)+2) + 3\sum_{j=1}^{n}(2\omega(c'_j)+2) + 2Cost_{RNS-RNS} \right) A_k \quad (8)$$

where $P_k$ represent the cost of one $k$-bit word product and $A_k$ the cost of one $k$-bit word addition.

For a parallel implementation, on $n$ $k$-bit word arithmetic units, the time complexity simplifies to:

$$PTC_{MMRNS} = 5P_k + \left( 4\max_{i=1,n}(\omega(c_i)) + 6\max_{j=1,n}(\omega(c'_j)) + 2PTC_{RNS-RNS} \right) A_k \quad (9)$$

## 4 Optimal RNS bases

In this section, pairs of RNS bases which provides reduced RNS-RNS conversions costs are provided. In his exhaustive search, no upper bound are fixed for the $m_i$ or their inverses, only $k$ is fixed to 64, and the only criterion used is the optimization of the cost (parallel or serial).

These bases are made of four, five and six 64-bit integer elements and support a large dynamic range (suitable, for instance, for ECC applications). The optimal RNS bases which provide the minimum RNS-RNS conversion costs have been obtained via an exhaustive search. The evaluation of the total conversion costs has been performed using the previously derived formulae (6) and (7). In order to try all the possible candidates for finding good bases, we have considered elements of the type $m_i = 2^k - c_i$ where $c_i < 2^{k/2}$, so that the reduction procedure is simplified as described in section 3.2. The bases are then formed with this type of moduli that are relatively prime to each other. All possible permutations in the relative positions of the elements have been considered, as this affect the overall performance. We note that the Hamming weights of $c_i$ or $c'_j$ contribute quadratically in the equations. Thus, it is not surprising that the elements of the RNS bases which provide the best complexity are of the form $m_i = 2^k - 2^{t_i} \pm 1$ and $m'_j = 2^k - 2^{t'_j} \pm 1$ (obviously $2^k$ and $2^k - 1$ appear in these sets).

In a parallel mode, where $n$ $k$-bit word arithmetic units are available, the tasks $a_j \leftarrow \left|(a_j - a_i)m_{i,j}^{-1}\right|_{m_j}$, used in the MRS reconstruction, valid for $j > i$ are executed in parallel for all the considered subindeces $j$ ; that is, for each $i$, all the tasks for $j \in [i+1, n]$ are performed in parallel. As a multiplication by the inverse $m_{i,j}^{-1}$ is replaced by additions or subtractions, the computation time of all the parallel tasks for each $i$, depends on the maximum of the Hamming weights of these inverses. By imposing an upper bound for the Hamming weight of the inverses, we were able to obtain bases that are optimal (in a sense that they provide the minimum conversion costs and the minimum Hamming weight for the inverses).

In a sequential mode, where the above mentioned tasks are performed successively, the latter restriction is relaxed. This enabled us to find bases with elements with a few number of inverses with high Hamming weight (which contribute in a small number of times in the reconstruction of the MRS coefficients), but having the remaining inverses with small Hamming weight (which contribute a higher number of times); thus, reducing the overall conversion cost.

Given an upper bound for the Hamming weight of the inverses, as soon as the inverse produced a Hamming weight higher than this bound, the entire set is discarded. Hence, the overall number of evaluations is actually much smaller than the theoretical upper bound of $n\frac{n-1}{2}\left(\prod_{i=0}^{n} k - i\right)$ evaluations of inverses. We have also considered a second set where one of the elements is $m_i = 2^k$. This gives a maximum number of $n\frac{n-1}{2}\left(\prod_{i=0}^{n-1} k - i\right)$ evaluations of inverses. Once the sets are obtained, the costs of the conversions back and forth between the sets are evaluated between sets with relatively prime elements using formulae (6) and (7).

As an example, for bases of four elements, the number of elements of the type $m_i = 2^k - 2^{t_i} \pm 1$ where $t < \frac{k}{2}$ is, for $k = 64$, equal to 61. We have then obtained, 2960 sets of relatively prime elements without the element $m_i = 2^{64}$ and 1007 sets where one of the elements is $m_i = 2^{64}$. The maximum Hamming weight for the recoded inverses is four. Thus, $2960 \times 1007$ evaluations of conversion costs have been performed. In addition, $2960 \times 2960$ conversions costs have been evaluated to verify that the no pairs of sets with smaller costs exists when the element $m_i = 2^{64}$ is excluded.

Table 1 summarizes the pairs of sets for optimal RNS bases with 64-bit elements. Table 2 shows the conversion costs. In the table, (P) means a parallel mode of operation, whereas (S) means a sequential mode of operation. We just give the total cost in number of 64-bits additions. We do not give the time complexity of a parallel implementation which should be $n$ times faster than the serial one.

## 5 Some theoretical results

We note that the inverses obtained from the experimental search have regular forms. In fact, it is not surprising, as the bases obtained have specific forms and the inverses $m_{i,j}^{-1}$ can easily be inferred. We formalize these remarks in two

**Table 1. Set of optimal RNS bases**

| | Basis $\mathcal{B}_n$ | $\omega_{m_i}$ | Basis $\mathcal{B}'_n$ | $\omega_{m'_i}$ |
|---|---|---|---|---|
| RNS bases 4 moduli (P and S) | $2^{64}-2^{10}-1$ | 3 | $2^{64}-2^{22}-1$ | 3 |
| | $2^{64}-2^{19}-1$ | 3 | $2^{64}-2^{23}-1$ | 3 |
| | $2^{64}-1$ | 2 | $2^{64}-2^{22}+1$ | 3 |
| | $2^{64}-2^{28}-1$ | 3 | $2^{64}$ | 1 |
| RNS bases 5 moduli (P) | $2^{64}-2^{8}-1$ | 3 | $2^{64}-2^{10}+1$ | 3 |
| | $2^{64}-2^{16}-1$ | 3 | $2^{64}-2^{9}-1$ | 3 |
| | $2^{64}-2^{22}-1$ | 3 | $2^{64}-2^{2}+1$ | 3 |
| | $2^{64}-2^{28}-1$ | 3 | $2^{64}-1$ | 2 |
| | $2^{64}$ | 1 | $2^{64}-2^{10}-1$ | 3 |
| RNS bases 5 moduli (S) | $2^{64}-2^{10}-1$ | 3 | $2^{64}-2^{8}-1$ | 3 |
| | $2^{64}-2^{19}-1$ | 3 | $2^{64}-2^{15}-1$ | 3 |
| | $2^{64}-2^{28}-1$ | 3 | $2^{64}-2^{16}-1$ | 3 |
| | $2^{64}-2^{20}-1$ | 3 | $2^{64}-1$ | 2 |
| | $2^{64}$ | 1 | $2^{64}-2^{22}-1$ | 3 |
| RNS bases 6 moduli (P) | $2^{64}-2^{10}-1$ | 3 | $2^{64}-2^{26}-1$ | 3 |
| | $2^{64}-2^{16}-1$ | 3 | $2^{64}-2^{18}-1$ | 3 |
| | $2^{64}-2^{19}-1$ | 3 | $2^{64}-2^{25}-1$ | 3 |
| | $2^{64}-2^{28}-1$ | 3 | $2^{64}-2^{17}-1$ | 3 |
| | $2^{64}-2^{20}-1$ | 3 | $2^{64}-1$ | 2 |
| | $2^{64}$ | 1 | $2^{64}-2^{5}-1$ | 3 |
| RNS bases 6 moduli (S) | $2^{64}-2^{10}-1$ | 3 | $2^{64}-2^{22}-1$ | 3 |
| | $2^{64}-2^{16}-1$ | 3 | $2^{64}-2^{13}-1$ | 3 |
| | $2^{64}-2^{19}-1$ | 3 | $2^{64}-2^{29}-1$ | 3 |
| | $2^{64}-2^{28}-1$ | 3 | $2^{64}-2^{30}-1$ | 3 |
| | $2^{64}-2^{20}-1$ | 3 | $2^{64}-1$ | 2 |
| | $2^{64}-2^{31}-1$ | 3 | $2^{64}$ | 1 |

**Table 2. Cost of the base extensions**

| | min./max. $\omega_{inv.}$ | $\mathcal{B}_n \rightarrow \mathcal{B}'_n$ MRS | $\mathcal{B}_n \rightarrow \mathcal{B}'_n$ ext. | $\mathcal{B}'_n \rightarrow \mathcal{B}_n$ MRS | $\mathcal{B}'_n \rightarrow \mathcal{B}_n$ ext. | Total cost |
|---|---|---|---|---|---|---|
| 4 mod. (P-S) | 2/5 | 45 | 71 | 35 | 79 | 230 |
| 5 mod. (P) | 2/11 | 90 | 131 | 96 | 122 | 439 |
| 5 mod. (S) | 2/17 | 97 | 128 | 86 | 119 | 430 |
| 6 mod. (P) | 2/17 | 159 | 193 | 198 | 182 | 732 |
| 6 mod. (S) | 2/20 | 219 | 174 | 131 | 202 | 726 |

theorems. In the first one we present the cases where the elements have the form $2^k$, $2^k-1$ and $2^k-2^t\pm1$. The second theorem is dedicated to pair of elements of the form $2^k-2^t\pm1$ and $2^k-2^{t+1}\pm1$.

**Theorem 3**

1. *Let $k\in\mathbb{N}^*$, then*

$$\left.\begin{array}{l} m_1=2^k,\\ m_2=2^k-1 \end{array}\right\}\Rightarrow\left\{\begin{array}{l} gcd(m_1,m_2)=1,\\ m_{1,2}^{-1}\equiv 1,\\ m_{2,1}^{-1}\equiv -1. \end{array}\right.$$

2. *Let $k,t\in\mathbb{N}^*$, then*

$$\left.\begin{array}{l} m_1=2^k,\\ m_3=2^k-2^t-1 \end{array}\right\}\Rightarrow\left\{\begin{array}{l} gcd(m_1,m_3)=1,\\ m_{3,1}^{-1}\equiv -\displaystyle\sum_{i=0}^{\lceil\frac{k}{t}-1\rceil}\left(-2^t\right)^i. \end{array}\right.$$

3. *Let $k,t\in\mathbb{N}^*$, then*

$$\left.\begin{array}{l} m_2=2^k-1,\\ m_3=2^k-2^t-1 \end{array}\right\}\Rightarrow\left\{\begin{array}{l} gcd(m_2,m_3)=1,\\ m_{2,3}^{-1}\equiv 2^{k-t}-1,\\ m_{3,2}^{-1}\equiv -2^{k-t}. \end{array}\right.$$

4. *Let $k,t_1,t_2\in\mathbb{N}$, then*

$$\left.\begin{array}{l} m_{3a}=2^k-2^{t_1}-1,\\ m_{3b}=2^k-2^{t_2}-1\\ \frac{k-t_1}{t_1-t_2}\in\mathbb{N} \end{array}\right\}\Rightarrow\left\{\begin{array}{l} gcd(m_{3a},m_{3b})=1,\\ m_{3a,3b}^{-1}\equiv -\displaystyle\sum_{i=0}^{\frac{k-t_1}{t_1-t_2}}(2^{t_1-t_2})^i,\\ m_{3b,3a}^{-1}\equiv \displaystyle\sum_{i=1}^{\frac{k-t_1}{t_1-t_2}}(2^{t_1-t_2})^i. \end{array}\right.$$

5. *Let $k,t\in\mathbb{N}^*$, then*

$$\left.\begin{array}{l} m_1=2^k\\ m_4=2^k-2^t+1 \end{array}\right\}\Rightarrow\left\{\begin{array}{l} gcd(m_1,m_4)=1,\\ m_{4,1}^{-1}\equiv \displaystyle\sum_{i=0}^{\lceil\frac{k}{t}\rceil-1}(2^t)^i. \end{array}\right.$$

6. *Let $k,t\in\mathbb{N}^*$, then*

$$\left.\begin{array}{l} m_2=2^k-1\\ m_4=2^k-2^t+1\\ \frac{k-1}{t-1}\in\mathbb{N} \end{array}\right\}\Rightarrow\left\{\begin{array}{l} gcd(m_2,m_4)=1,\\ m_{4,2}^{-1}\equiv \displaystyle\sum_{i=0}^{\frac{k-1}{t-1}-1}(2^{t-1})^i. \end{array}\right.$$

7. *Let $k,t\in\mathbb{N}^*$, then*

$$\left.\begin{array}{l} m_3=2^k-2^t-1\\ m_4=2^k-2^t+1 \end{array}\right\}\Rightarrow\left\{\begin{array}{l} gcd(m_3,m_4)=1,\\ m_{3,4}^{-1}\equiv 2^{k-1}-2^{t-1},\\ m_{4,3}^{-1}\equiv 2^{k-1}-2^{t-1}. \end{array}\right.$$

8. *Let $k,t_1,t_2\in\mathbb{N}$, then*

$$\left.\begin{array}{l} m_{4a}=2^k-2^{t_1}+1\\ m_{4b}=2^k-2^{t_2}+1\\ \frac{k-t_1}{k-t_2}\in\mathbb{N} \end{array}\right\}\Rightarrow\left\{\begin{array}{l} gcd(m_{4a},m_{4b})=1,\\ m_{4a,4b}^{-1}\equiv \displaystyle\sum_{i=0}^{\frac{k-t_2}{t_1-t_2}}(2^{t_1-t_2})^i,\\ m_{4b,4a}^{-1}\equiv -\displaystyle\sum_{i=1}^{\frac{k-t_1}{t_1-t_2}}(2^{t_1-t_2})^i. \end{array}\right.$$

**proof 1** *The proofs are given in a report [4].*

Table 3 resumes the results presented in Theorem 3, in terms of Hamming weight with $m_1=2^k$, $m_2=2^k-1$, $m_3=2^k-2^{t_1}-1$, $m_4=2^k-2^{t_2}-1$, $m_5=2^k-2^{t_1}+1$, $m_6=2^k-2^{t_2}+1$.

**Theorem 4**

1. *Let $k,t\in\mathbb{N}$, then,*

$$\left.\begin{array}{l} m_{3a}=2^k-2^{t+1}-1\\ m_{3b}=2^k-2^t-1 \end{array}\right\}\Rightarrow\left\{\begin{array}{l} gcd(m_{3a},m_{3b})=1,\\ m_{3a,3b}^{-1}\equiv -2^{k-t}+1,\\ m_{3b,3a}^{-1}\equiv 2^{k-t}-2. \end{array}\right.$$

**Table 3. Hamming weight $w(m_{i,j}^{-1})$ of the inverse of $m_i$ modulo $m_j$, with $m_1 = 2^k$, $m_2 = 2^k - 1$, $m_3 = 2^k - 2^{t_1} - 1$, $m_4 = 2^k - 2^{t_2} - 1$, $m_5 = 2^k - 2^{t_1} + 1$, $m_6 = 2^k - 2^{t_2} + 1$.**

| | $m_j$ | | | | | |
|---|---|---|---|---|---|---|
| $m_i$ | $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | $m_6$ |
| $m_1$ | | 1 | | | | |
| $m_2$ | 1 | | 2 | 2 | | |
| $m_3$ | $\frac{k}{t_1}$ | 1 | | $\frac{k-t_2}{t_1-t_2}$ | 2 | |
| $m_4$ | $\frac{k}{t_2}$ | 1 | $\frac{k-t_1}{t_1-t_2}$ | | | 2 |
| $m_5$ | $\frac{k}{t_1}$ | $\frac{k-1}{t_1-1}$ | 2 | | | $\frac{k-t_1}{t_1-t_2}$ |
| $m_6$ | $\frac{k}{t_2}$ | $\frac{k-1}{t_2-1}$ | | 2 | $\frac{k-t_1}{t_1-t_2}$ | |

*2. Let $k, t \in \mathbb{N}$, then,*

$$\left.\begin{array}{l} m_{4a} = 2^k - 2^{t+1} + 1 \\ m_{4b} = 2^k - 2^t + 1 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} gcd(m_{4a}, m_{4b}) = 1, \\ m_{4a,4b}^{-1} \equiv 2^{k-t} - 1, \\ m_{4b,4a}^{-1} \equiv -2^{k-t} + 2. \end{array}\right.$$

*3. Let $k, t \in \mathbb{N}^*$, then,*

$$\left.\begin{array}{l} m_{3a} = 2^k - 2^{t+1} - 1 \\ m_{4b} = 2^k - 2^t + 1 \\ \frac{k-t}{2(t-1)} \in \mathbb{N} \end{array}\right\} \Rightarrow \left\{\begin{array}{l} gcd(m_{3a}, m_{4b}) = 1, \\ m_{3a,4b}^{-1} \equiv -\sum_{i=1}^{\frac{k-t}{t-1}} (-2^{t-1})^i. \end{array}\right.$$

$$\left.\begin{array}{l} m_{3a} = 2^k - 2^{t+1} - 1 \\ m_{4b} = 2^k - 2^t + 1 \\ \frac{k-t-1}{2(t-1)} \in \mathbb{N} \end{array}\right\} \Rightarrow \left\{\begin{array}{l} gcd(m_{3a}, m_{4b}) = 1, \\ m_{4b,3a}^{-1} \equiv 2 \sum_{i=1}^{\frac{k-t-1}{t-1}} (-2^{t-1})^i. \end{array}\right.$$

*4. Let $k, t \in \mathbb{N}^*$, then,*

$$\left.\begin{array}{l} m_{3b} = 2^k - 2^t - 1 \\ m_{4a} = 2^k - 2^{t+1} + 1 \\ \frac{k-t-1}{t-1} \in \mathbb{N} \end{array}\right\} \Rightarrow \left\{\begin{array}{l} gcd(m_{3b}, m_{4a}) = 1, \\ m_{3b,4a}^{-1} \equiv -2 \sum_{i=1}^{\frac{k-t-1}{t-1}} (2^{t-1})^i. \end{array}\right.$$

$$\left.\begin{array}{l} m_{3b} = 2^k - 2^t - 1 \\ m_{4a} = 2^k - 2^{t+1} + 1 \\ \frac{k-t}{t-1} \in \mathbb{N} \end{array}\right\} \Rightarrow \left\{\begin{array}{l} gcd(m_{3b}, m_{4a}) = 1, \\ m_{4a,3b}^{-1} \equiv -\sum_{i=1}^{\frac{k-t}{t-1}} (2^{t-1})^i. \end{array}\right.$$

**proof 2** *Most of the cases correspond to corollaries of Theorem 3, or can be proved using the same schemes.*

We present in Table 4 the Hamming weight of the different inverses found in Theorem 4.

## 6 Discussion

Modular multiplication with large modulus is heavily used in asymmetric key cryptography. For computationally intensive applications, such as signing servers in e-commerce, acceleration of ECC operations are performed by using dedicated hardware. One of the main drawbacks of this approach is that hardware is usually designed for a fixed prime and scaling it up is usually not supported. In contrast to this, the use of RNS might overcome such a problem since the prime $P$ of the base field can be modified even when the RNS bases are fixed, providing flexibility and gradual security levels. Furthermore, since the bit sizes can be modified, randomization techniques can be applied on the modulus to increase security against differential power attacks [17, 3]. That is, before performing the scalar multiplication on the EC, the modulus is scaled by a random integer. The result is then normalized to $[0, P[$.

**Table 4. Hamming weight $w(m_{i,j}^{-1})$ of the inverse of $m_i$ modulo $m_j$, with $m_1 = 2^k$, $m_2 = 2^k - 1$, $m_3 = 2^k - 2^{t+1} - 1$, $m_4 = 2^k - 2^t - 1$, $m_5 = 2^k - 2^{t+1} + 1$, $m_6 = 2^k - 2^t + 1$.**

| | $m_j$ | | | | | |
|---|---|---|---|---|---|---|
| $m_i$ | $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | $m_6$ |
| $m_1$ | | 1 | | | | |
| $m_2$ | 1 | | 2 | 2 | | |
| $m_3$ | $\left\lceil \frac{k}{t+1} \right\rceil$ | 1 | | 2 | 2 | $\frac{k-t}{t-1}$ |
| $m_4$ | $\left\lceil \frac{k}{t} \right\rceil$ | 1 | 2 | | $\frac{k-t-1}{t-1}$ | 2 |
| $m_5$ | $\left\lceil \frac{k}{t+1} \right\rceil$ | $\frac{k-1}{t}$ | 2 | $\frac{k-t}{t-1}$ | | 2 |
| $m_6$ | $\left\lceil \frac{k}{t} \right\rceil$ | $\frac{k-1}{t-1}$ | $\frac{k-t-1}{t-1}$ | 2 | 2 | |

If we compare our approach with the recommendations for ECDSA [13] (suggested by the NIST), taking into account that the cost (conjointly considering area and delay) of a $64\times64$-bit multiplier is equivalent to 63 additions of 64-bit values (which is true in radix 2 with an architecture without multiplier), we obtain for $P_{384} = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$ a cost of 2365 additions for a modular multiplication and 1898 additions for $P_{256} = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. We note that the cost is due to a multiplication of big numbers, and then a modular reduction is with $P_{384}$ than with $P_{256}$ using the fact that they are sparse. With our approach, for any $P$ of 320 up to 384 bits (lower than $M$) with $n = 6$, we need 2850 additions (which represents 20% more than for $P_{384}$) and 2089 additions for any $P$ of 256 up to 320 bits with $n = 5$ (which represents 10% more than for $P_{256}$). In this comparison, we do not take into account the cost of the propagation of 64-bits values used in big number arithmetics. If we compare to the standard Montgomery algorithm [16, 9] or with the previous RNS adaptations [1, 25] which requires at least $2n^2$ multiplications of $k$-bit values (for 384 that represents 4536 additions and 3150 for $k = 320$). If we compare to tha Karatsuba implementation of [9], taking into account the total cost, we obtain similar costs with a straightforward architecture.

## References


[1] J.C. Bajard, L.S. Didier and P. Kornerup, Modular multiplication and base extension in residue number systems, 15th IEEE Symposium on Computer Arithmetic, IEEE Computer Society Press (2001) 59–65

[2] J.C. Bajard, L. Imbert, A full RNS implementation of RSA, IEEE Transactions on Computers **53:6** (2004) 769–774

[3] J.C. Bajard, L. Imbert, P.Y. Liardet and Y. Teglia, Leak resistant arithmetic, CHES 2004, LNCS **3156** 59–65

[4] J.C. Bajard,M. Kaihara and T. Plantard, Report: Selected RNS Bases for Modular Multiplication, http://www.lirmm.fr/ bajard/MesPublis/BKP09Report.pdf

[5] G.C. Cardarilli, M. Re and R. Lojacono, RNS-to-binary conversion for efficient VLSI implementation, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, Volume 45, Issue 6, Jun 1998.

[6] G.C. Cardarilli, A. Nannarelli and M. Re, Residue Number System for Low Power DSP Applications, Proc. of 41st Asilomar Conference on Signals, Systems, and Computers, p. 1412-1416. November 4-7, 2007.

[7] R. Conway and J. Nelson, New CRT-based RNS converter using restricted moduli set, Computers, IEEE Transactions on On page(s): 572- 578, Volume: 52, Issue: 5, May 2003

[8] E.D. Di Claudio, F. Piazza, and G. Orlandi, Fast combinatorial RNS processors for DSP applications, IEEE Transactions on Computers, page(s): 624-633, Volume: 44, Issue: 5, May 1995

[9] J. P. David, K. Kalach, and N. Tittley, Hardware Complexity of Modular Multiplication and Exponentiation IEEE Transactions on Computers, Vol. 56, No. 10, October 2007

[10] L.-S. Didier and P.-Y. Rivaille, A generalization of a Fast RNS conversion for a new 4-modulus base, to appear in EEE transactions on Circuits and Systems II.

[11] W. Diffie and M.E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory 22 (1976), 644-654.

[12] M. Hosseinzadeh, S.J. Jassbi and K. Navi, A New Moduli Set $3^n - 1, 3^n + 1, 3^n + 2, 3^n - 2$ in Residue Number System, 10th ICACT 2008. Volume 3, 17-20 Feb. 2008 Page(s):1601 - 1603

[13] D. Johnson and A. Menezes The Elliptic Curve Digital Signature Algorithm (ECDSA) http://www.cacr.math.uwaterloo.ca/techreports/... ...1999/tech_reports99.html

[14] N. Koblitz Elliptic Curve Cryptosystems Mathematics of Computation, Vol. 48, No. 177. (Jan., 1987), pp. 203-209.

[15] V. S Miller Use of elliptic curves in cryptography LNCS 218 on Advances in cryptology—CRYPTO 85 Santa Barbara, California, US P.: 417 - 426, 1986.

[16] P.L. Montgomery, Modular multiplication without trial division. Mathematics of Computation, 44:170 (1985) 519–521

[17] M. Ciet, M. Neve, E. Peeters and J.J. Quisquater, Parallel FPGA implementation of RSA with residue number systems– can side-channel threats be avoided? 46th IEEE Int. MW Symp. on Circuits and Systems (2003)

[18] S. J. Piestrak, A high-speed realization of a residue to binary number system converter, IEEE Trans. Circuits Syst. II, vol. 42, pp. 661-663, Oct. 1995.

[19] K.C. Posch and R. Posch, Modulo reduction in residue number systems. IEEE Transaction on Parallel and Distributed Systems, 6:5 (1995) p.: 449–454

[20] A. B. Premkumar, An RNS to binary converter in $2^n - 1, 2^n, 2^n + 1$ moduli set, IEEE Trans. Circuits Syst. II, vol. 39, pp. 480-482, July 1992.

[21] R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, Vol. 21 (2), pp.120-126. 1978

[22] M.A. Soderstrand, W.K. Jenkins, G.A. Jullien, and F.J. Taylor, Residue Number System Arithmetic: Modern Applications in Digital Signal Processing, eds. IEEE Press, 1986

[23] A. Svoboda and M. Valach, Operational Circuits. Stroje na Zpracovani Informaci, Sbornik III, Nakl. CSAV, Prague, 1955, pp.247-295.

[24] N.S. Szabo and R.I. Tanaka, Residue Arithmetic and its Applications to Computer Technology. McGraw-Hill (1967)

[25] R. Szerwinski and T. Güneysu, Exploiting the Power of GPUs for Asymmetric Cryptography CHES 2008, LNCS 5154, pp. 79-99, 2008

[26] Y. Wang, New Chinese remainder theorems, Thirty-Second Asilomar Conference on Signals, Systems & Computers, 1998, page(s): 165-171 vol.1