Fault Detection in RNS Montgomery Modular Multiplication

Jean-Claude Bajard, Julien Eynard*
LIP6 CNRS - Univ. Pierre et Marie Curie
Paris, France
{*jean-claude.bajard,julien.eynard*}*@lip6.fr*

Filippo Gandino
Politecnico di Torino
Torino, Italy
filippo.gandino@polito.it

***Abstract*—Recent studies have demonstrated the importance of protecting the hardware implementations of cryptographic functions against side channel and fault attacks. In last years, very efficient implementations of modular arithmetic have been done in RNS (RSA, ECC, pairings) as well on FPGA as on GPU. Thus the protection of RNS Montgomery modular multiplication is a crucial issue. For that purpose, some techniques have been proposed to protect this RNS operation against side channel analysis. Nevertheless, there are still no effective and generic approaches for the detection of fault injection, which would be additionnally compatible with a leak resistant arithmetic. This paper proposes a new RNS Montgomery multiplication algorithm with fault detection capability. A mathematical analysis demonstrates the validity of the proposed approach. Moreover, an architecture that implements the proposed algorithm is presented. A comparative analysis shows that the introduction of the proposed fault detection technique requires only a limited increase in area.**

***Keywords*-Residue Number System; Fault Detection; Montgomery Reduction; Base Conversions**

I. Introduction

Residue Number Systems (RNS) [1], [2] enable to make computations on large numbers through an arithmetic based on additions, subtractions and multiplications run on small units (i.e. the elements of a chosen RNS base) in a parallel and carry-free way. However, because they are non-positional numeral systems, comparison, division and modular reduction are more difficult to perform. For instance, Bajard et al. [3] propose an algorithm for modular multiplication in RNS (RNSMM) adaptating the classical Montgomery reduction [4]. By combining this algorithm with efficient base conversion [5], [6] fast execution of modular multiplications/exponentiations on large integers, which is of special interest for cryptographic applications, become possible in RNS. For that matter Guillermin et al. [7]–[9] show the real effectiveness of RNS for hardware implementation of cryptosystems as RSA, ECC and pairings.

In cryptographic context, RNS also allows to use a leak resistant arithmetic providing a natural protection against side channel analysis such as timing, power and electromagnetic attacks [10] through randomization of bases between operations. Another kind of dangerous threat is fault attacks, which are a well-known way to extract informations of cryptosystems. It was initially used by Boneh et al. [11] to break RSA-CRT. Because data are scattered across the units, RNS supplies a natural robustness against faults. Indeed it suffices to add a redundant unit to detect any error affecting a single small value. However, standard redundant RNS techniques for fault detection [12]–[15] are feasible as long as no base extension operations and/or overflow appear. In first case, a fault could be propagated to all units of the new base. In second case, the redundant value is not redundancy of RNS data anymore. In the end, the protection of RNS modular multiplication against fault injection appears necessary so as to consolidate interest of RNS arithmetic in cryptography. However, RNSMM algorithms [3], [16] contain two consecutive base conversions which are *a priori* not compliant with standard redundant detection techniques. The goal of the paper is to adapt such standard countermeasures using redundancy to RNS modular multiplication.

In this paper a new algorithm, inserting redundancy into RNS Montgomery multiplication in order to provide efficient and cheap protection against fault attacks in the context of modular exponentiation/elliptic curve point addition, and compliant with a leak resistant arithmetic, is presented and analyzed. A formal analysis proves the detection of any single fault. Beyond theoretical purposes, an architecture resistant to single fault attacks and compliant with previous protections for RNS against side channel attacks is suggested. The architectural design is based on the work presented in [17] by Nozaki et al. An analysis based on equivalent gates shows that the area increase required for single fault detection is lower than 3%, and that the total computational time increases by about $1/(2\log_2(exponent))$ of the normal delay if the output of RNSMM is required by an RNS implementation that cannot execute the fault detection.

The paper is organized as follows. In section II residue number systems, base extensions and redundancy for error detection are introduced. Section III presents and analyzes an algorithm providing detection of single faults injected during the execution of a RNSMM. Section IV proposes an architecture implementing the present method. A comparison to the only other technique (as far as the authors know) aiming to detect same kind of faults and proposed in [8] is discussed. Finally, section V draws conclusions.

*supported by Direction Générale de l'Armement (DGA - French Ministry of Defence)

II. About RNS and fault detection

A. Residue Number Systems - Notations

A RNS base $\mathcal{B}$ is constituted by n moduli $m_1, \ldots, m_n$ and its dynamic range is $[\![0, M[\![:= [0, M) \cap \mathbb{Z}$ with $M := \prod_{i=1}^{n} m_i$. For $i \in [\![1, n]\!]$, M_i denotes $\frac{M}{m_i}$, and for any integer x, $x_i = |x|_{m_i} = x \ mod \ m_i$. The Chinese Remainder Theorem [18] applied to base $\mathcal{B}$ states that there exists a ring isomorphism $\varphi_{\mathcal{B}} : (\mathbb{Z}/(M), +, \times) \xrightarrow{\sim} (\mathbb{Z}/(m_1) \times \ldots \times \mathbb{Z}/(m_n), +_{prod}, \times_{prod})$ if, and only if, the m_i's are pairwise coprime. So the moduli of a RNS base will be implicitly chosen pairwise coprime. Two main consequences of this theorem may be emphasized for our purpose. Firstly, for any $x \in \mathbb{Z}$ there is a one-to-one correspondance between $|x|_M$ and $\left(|x|_{m_1}, \ldots, |x|_{m_n}\right)_{\mathcal{B}}$. $\boldsymbol{x}_{\boldsymbol{\mathcal{B}}}$ can denote as well the n-tuple of residues of x in $\mathcal{B}$ as x or $|x|_M$. More generally, tuples of residues and integers will be later merged in some expressions, in order to emphasize some properties. However, in this case the residues will be seen as the unique integer they represent in the dynamic range. Secondly, for every $(x, y) \in [\![0, M[\![^2$, $\varphi_{\mathcal{B}}(|x \intercal y|_M) = \left(|x_1 \intercal y_1|_{m_1}, \ldots, |x_n \intercal y_n|_{m_n}\right)_{\mathcal{B}}$ with $\intercal \in \{+, -, \times\}$. And if $x \wedge M := \gcd(x, M) = 1$ then $\varphi_{\mathcal{B}}\left(|x^{-1}|_M\right) = \left(|x_1^{-1}|_{m_1}, \ldots, |x_n^{-1}|_{m_n}\right)_{\mathcal{B}}$.

B. Base conversions/extensions

Given residues $\boldsymbol{x}_{\mathcal{B}}$ of $x \in [\![0, M[\![$, a base extension aims to compute residues of x for new moduli. There are two main types of extension which are summarily described hereafter.

1) Mixed-Radix Conversion (MRC) [2]: x is transformed from RNS base $\mathcal{B}$ to another one by passing through the mixed-radix system (MRS) associated to $\mathcal{B}$. MRS coefficients of x are also an element of $\mathbb{Z}/(m_1) \times \ldots \mathbb{Z}/(m_n)$.

2) Chinese Remainder Theorem (CRT) [18]: A classical way to compute x from $\boldsymbol{x}_{\boldsymbol{\mathcal{B}}} = (x_1, \ldots, x_n)_{\mathcal{B}}$ is

$$x = \left| \sum_{i=1}^{n} \left|x_i M_i^{-1}\right|_{m_i} M_i \right|_M .$$

$ext_{\mathcal{B}}(x) := \sum_{i=1}^{n} \left|x_i M_i^{-1}\right|_{m_i} M_i$ is called the "crt-sum" of $\boldsymbol{x}_{\boldsymbol{\mathcal{B}}}$, and $k_x := \lfloor \frac{ext_{\mathcal{B}}(x)}{M} \rfloor$ its "crt-correction coefficient". Thus $x = ext_{\mathcal{B}}(x) - k_x M$. Practically, k_x is not recovered through modular reduction of macro quantities like $ext_{\mathcal{B}}(x)$. Two methods are now described to get k_x.

Shenoy and Kumaresan's method (SK) [5]: they consider the fact that $k_x \in [\![0, n[\![$ to propose to use an extra redundant modulus $m_{sk} \geqslant n$ coprime to M. Consequently, if $x_{sk} := |x|_{m_{sk}}$ is known, one has:

$$k_x = |k_x|_{m_{sk}} = \left|\left(ext_{\mathcal{B}}(x) - x_{sk}\right) M^{-1}\right|_{m_{sk}} . \qquad (1)$$

In this method, the base is now $\mathcal{B} \cup \{m_{sk}\}$. Thus, RNS operations handle values in a larger dynamic range $[\![0, m_{sk} M[\![$. However, to keep an exact extension of the integer expressed through the residues over $\mathcal{B}$, x_{sk} must remain pure redundancy, i.e. the true dynamic range is still $[\![0, M[\![$.

Kawamura et al.'s method (KWu, KWc) [6]: Denoting $\xi_i = \left|x_i M_i^{-1}\right|_{m_i}$, the crt-sum is rewritten $ext_{\mathcal{B}}(x) = \sum_{i=1}^{n} \xi_i M_i$. Then, $k_x = \lfloor \sum_{i=1}^{n} \frac{\xi_i}{m_i} \rfloor$. Kawamura et al. use this equality to compute an approximation of the correction term k_x. They underapproximate the quantities $\frac{\xi_i}{m_i}$ by $\frac{trunc(\xi_i)}{2^r}$, where $trunc$ function keeps only q most significant bits and set the others to zero, and r is s.t. $2^{r-1} < m_i < 2^r$. This approximation of k_x is computed by a simple unit called Cox while computations on moduli are performed in cells called Rowers. The problem of approximation is that Cox can return as well k_x as $k_x - 1$. In the last case, the extension returns $|x + M|_m$ for a new residue. More precisely, theorem 2 of [6] proves that this case will occur only for some $x < \Delta_{kw} M$, where $\Delta_{kw} \in [\![0, 1[\![$ is an upper-bound of all possible errors, i.e. $0 \leqslant \sum_{i=1}^{n} \left(\frac{\xi_i}{m_i} - \frac{trunc(\xi_i)}{2^r}\right) < \Delta_{kw}$. Thus if no control is ensured on the size of the input of extension, the result can be not exact. That unexact extension will be denoted KWu. However, theorem 1 of [6] states that if the input is guaranteed smaller than any quantity $(1 - \alpha_{kw}) M$ for any $\alpha_{kw} \in [\![\Delta_{kw}, 1[\![$, then by adding α_{kw} in Cox to correct the computed crt-correction coefficient the extension is exact. This corrected version is denoted KWc.

C. Redundant RNS and fault detection

The way to use redundancy to detect faults in RNS has been studied by several authors (e.g. [12]–[15], [19]) for many years. By adding a redundant modulus m_R to $\mathcal{B}$, the state space is extended to $[\![0, M m_R[\![$, and if m_R is large enough any value x affected by a single fault belongs to $[\![M, M m_R[\![$. Then, receiving $(\boldsymbol{x}_{\boldsymbol{\mathcal{B}}}, x_{m_R})_{\mathcal{B} \cup \{m_R\}}$, the usual detection procedure is the computation of $\left|\boldsymbol{x}_{\boldsymbol{\mathcal{B}}}\right|_{m_R}$ via a base extension, and its comparison to x_{m_R}. This is called a *consistency check*. If Bex denotes the base extension method used for the check (e.g. MRC, SK or KW), then one has to verify if $\left|\text{Bex}\left(\tilde{\boldsymbol{x}}_{\boldsymbol{\mathcal{B}}}\right) - x_{m_R}\right|_{m_R}$ is null or not, where $\tilde{\boldsymbol{x}}_{\boldsymbol{\mathcal{B}}}$ may contain one faulty residue. Thereafter possible values of $err(x) := \left|\text{Bex}\left(\tilde{\boldsymbol{x}}_{\boldsymbol{\mathcal{B}}}\right) - x\right|$ are given to supply necessary and sufficient conditions on m_R for detecting any single fault.

1) MRC: The case where the check is based on MRC extension has been extensively studied. Trivially, the set of all possible values of $err(x)$ is exactly $]\!]0, M[\![\cap (M_i \mathbb{Z})$.

2) CRT: Here, since all residues contribute to the computation k_x, a fault could modify it. So the faulty extended value could not be in the original range $[\![0, M[\![$.

a) SK: $\text{SK}\left(\tilde{\boldsymbol{x}}_{\boldsymbol{\mathcal{B}} \cup \{\boldsymbol{m_{sk}}\}}\right) \in [\![0, m_{sk} M[\![$. More precisely two cases can appear (for brevity's and completeness' sake, constructive examples are given but without details). Either the fault affects the redundant residue x_{sk}. In this case, $err(x) = \left|\text{SK}\left(\tilde{\boldsymbol{x}}_{\boldsymbol{\mathcal{B}} \cup \{\boldsymbol{m_{sk}}\}}\right) - x\right|$ can reach any value in $]\!]0, m_{sk} M[\![\cap (M\mathbb{Z})$. Indeed, from eq. (1), by taking $x = 0$

and a fault $e_{sk} = |-\gamma M|_{sk}$ with γ any value in $]\!]0, m_{sk}[\![$, then $err(x) = \gamma M$. Or the fault is on a residue x_i. Here, $err(x)$ reaches any value in the set $]\!]0, m_{sk} M[\![\cap (m_{sk} M_i \mathbb{Z})$. For instance, to obtain $err(x) = \gamma m_{sk} M_i$ with γ any value in $]\!]0, m_i[\![$, if $\left||-\gamma m_{sk}|_{m_i}\right|_{m_{sk}} \neq 0$ one can take $e_i = |-\gamma m_{sk} M_i|_{m_i}$ and $x = 0$. Else, there exists $0 < \nu < \frac{m_i}{m_{sk}}$ s.t. $|-\gamma m_{sk}|_{m_i} = \nu m_{sk}$, and one can take $e_i = |-\nu m_{sk} M_i|_{m_i}$ and $x = \left|-e_i M_i^{-1}\right|_{m_i} M_i$.

b) KWc: Here it is assumed that $x < (1 - \alpha_{kw}) M$ and $\tilde{x} := \varphi_{\mathcal{B}}^{-1}(\tilde{\boldsymbol{x}}_{\boldsymbol{\mathcal{B}}}) = x + eM_i \in [\![0, M[\![$ where $e \in]\!] - m_i, m_i[\![$. It follows that KWc $(\tilde{\boldsymbol{x}}_{\boldsymbol{\mathcal{B}}}) = x + eM_i - \mu M$ where $\mu \in \{0, 1\}$. Moreover $\mu = 1$ only if $(1 - \alpha_{kw}) M \leqslant \tilde{x} < M$, i.e. only if $e > 0$. Then $err(x) = |\text{KWc}(\tilde{\boldsymbol{x}}_{\boldsymbol{\mathcal{B}}}) - x|$ reaches at most any value of $]\!]0, M[\![\cap (M_i \mathbb{Z})$.

3) Required redundancy to detect single faults: Here is given a sufficient (and necessary) condition on size of m_R to detect any single fault through a consistency check based on a MRC, SK or KWc extension. The MRC case has been treated in [12]–[15] for instance. Condition for SK case is, as far as the authors know, supplied for the first time.

Theorem 2.1: Let $\mathcal{B} \cup \{m_R\}$ be a redundant RNS, where a modulus m_{sk} is included in $\mathcal{B}$ if SK is used for the consistency check. A necessary and sufficient (resp. sufficient) condition to detect any single fault injected on any tuple of residues $(\tilde{\boldsymbol{x}}_{\boldsymbol{\mathcal{B}}}, x_{m_R})$ expressed in $\mathcal{B} \cup \{m_R\}$ through a consistency check based on MRC or SK (resp. KWc) extension is: $\forall m \in \mathcal{B},\ m_R \geqslant m \times \left(m_R \wedge \frac{M}{m}\right)$.

Proof: Previously it has been shown that for MRC, SK and KWc extensions, the faults to be detected have the form $g\frac{M}{m}$ where $|g|$ is in $]\!]0, m[\![$. So if such a fault is not detected, it means that there exists an integer γ verifying $g\frac{M}{m} = \gamma m_R$. Then Euclide's lemma implies that $\frac{m_R}{m_R \wedge \frac{M}{m}}$ divides g, and so $m_R < m \times \left(m_R \wedge \frac{M}{m}\right)$. This proves the sufficiency for all cases. The necessity is proved by contraposition for MRC case, by exhibiting an undetectable error. The sketch of the proof is similar for the SK extension, by using given constructive examples. If $m_R < m_i \times (m_R \wedge M_i)$ for any m_i in $\mathcal{B}$, the error $e = \left|\frac{m_R}{m_R \wedge M_i} M_i M_i^{-1}\right|_{m_i} \times M_i$ injected on $x = 0$ will not be detected. Indeed, $e = \frac{m_R}{m_R \wedge M_i} M_i = m_R \frac{M_i}{m_R \wedge M_i}$ where the first equality uses the hypothesis $m_R < m_i \times (m_R \wedge M_i)$. ■

Rem. 1: Due to the approximations in KWc extension, m_R could be chosen smaller than some m_i's. However, only the sufficiency is useful for the purpose of section IV.

III. Fault detection in RNS modular multiplication

Efficient RNS modular multiplication [3], [16] adapts Montgomery reduction to RNS. Because of division by M in classical Montgomery reduction, an auxiliary base $\mathcal{B}'$ coprime to the main base $\mathcal{B}$ is used. Algorithm 1 summarizes the technique. Hypothesis $\mathcal{H}_{mrc}$ [3], $\mathcal{H}_{sk}$ [16] or $\mathcal{H}_{kw}$ [6] in Fig.1 (where p is supposed given) reflect three versions of RNSMM using different types of base extension for Bex_1 and Bex_2. Hypothesis on M and M' guarantee that $t := xy + \hat{q}p$, which is a multiple of M and the largest value in the algo., stays in the full dynamic range $[\![0, MM'[\![$. For instance for KW version, theorem 2 of [6] states that $\hat{q} < (1 + \Delta_{kw}) M$. Given $\mathcal{H}_{kw}$, $t < (1 - \Delta_{kw}) M + (1 + \Delta_{kw}) M = 2Mp < (1 - \alpha_{kw}) M'M$. So $s = \frac{t}{M} < (1 - \alpha_{kw}) M'$ and KWc is used for Bex_2.

hyp.	$\mathcal{H}_{mrc}$	$\mathcal{H}_{sk}$	$\mathcal{H}_{kw}$
Bex_1	MRC	crt-sum $ext_{\mathcal{B}}$	KWu
Bex_2	MRC	SK	KWc
bases	$MM' \wedge p = 1$	$m_{sk} MM' \wedge p = 1$ $m_{sk} \in \mathcal{B}'$	$MM' \wedge p = 1$
$m_R >$	$\max(m_i, m_i')$	$\max(m_{sk} m_i, m_i')$	$\max(m_i, m_i')$
$xy <$	Mp	Mp	$4p^2$
$M >$	$2p$	$(n+1)^2 p$	$4p/(1 - \Delta_{kw})$
$M' >$	M	$(n+1) p$	$2p/(1 - \alpha_{kw})$
output	$s < 2p$	$s < (n+1) p$	$s < 2p$

Figure 1: Hypothesis for Algorithm 1

Algorithm 1: Redundant RNS modular multiplication
Input: coprime bases $\mathcal{B}$ and $\mathcal{B}'$, m_R coprime to $\mathcal{B}$ and $\mathcal{B}'$, integers p, x and y (expressed in $\mathcal{B} \cup \mathcal{B}' \cup \{m_R\}$) and base extensions Bex_1 and Bex_2 verifying hypothesis $\mathcal{H}_{mrc}$, $\mathcal{H}_{sk}$ or $\mathcal{H}_{kw}$
Output: s expressed in $\mathcal{B} \cup \mathcal{B}' \cup \{m_R\}$ s.t. $s \equiv xyM^{-1}\ (mod\ p)$
1: $q \leftarrow (-x \times_{rns} y) \times_{rns} p^{-1}$ in $\mathcal{B}$
2: $\hat{q} \leftarrow \text{Bex}_1(q)$: extension from $\mathcal{B}$ to $\mathcal{B}' \cup \{m_R\}$
3: $t \leftarrow x \times_{rns} y +_{rns} \hat{q} \times_{rns} p$ in $\mathcal{B}' \cup \{m_R\}$
4: $s \leftarrow t \times_{rns} M^{-1}$ in $\mathcal{B}' \cup \{m_R\}$
5: $\hat{s} \leftarrow \text{Bex}_2(s)$: extension from $\mathcal{B}'$ to $\mathcal{B} \cup \{m_R\}$
6: **if** $|\hat{s}|_{m_R} \neq |s|_{m_R}$ **then**
7: Error detected

A. Adequation of single fault model for RNSMM algo.

The purpose of this discussion is the pertinence of the single fault model in the presence of base extensions, as it is the case in RNSMM algo. If the fault is injected during an extension on a quantity computed in a quotient ring $\mathbb{Z}/(m)$ where m is a modulus of one of the bases concerned by the extension, then the effect is strictly equivalent to a single fault injected either before or after the extension. This consideration is really pertinent because in all base extension techniques previously seen, computations are ran only in such rings (except for Cox; but cf. part III-B4). So a fault injected during a base extension will be considered as a single fault appearing either before or after the extension.

Rem. 2: From hardware point of view, this analysis really depends on chosen architecture and capacities of the attacker, because for instance an attack could be launched during the distribution of the ξ_i's coefficient in new residues. Thus some of them could receive the correct value, and some others a faulty value. Such possibilities must be prevented.

B. Introduction of redundancy - Fault location

Rem. 3: (guideline) As consistency checks require a base extension which is a costly operation, the detection tech-

nique should only use the own extensions of RNSMM algo.

Given part III-A about faults during base extension, only five types of error are to be considered in context of RNSMM algo: on a residue in base $\mathcal{B}$ before Bex_1 called category 1; on a residue in the second base $\mathcal{B}'$ called category 2; on a residue in base $\mathcal{B}$ after Bex_2 called category 3; and category 4 for faults on extra material for extensions such as m_{sk} channel or Cox. Category 5 concerns faults on redundant residues. Moreover, a fundamental hypothesis is that, given m_R, the redundant residues of x, y, and p are part of input of RNSMM algo. This requires that redundant version of the algo. must output s expressed in $\mathcal{B} \cup \mathcal{B}' \cup \{m_R\}$. Due to independency between residues during parallel RNS operations, as in lines 1, 3 and 4 in algo.1, cat.1 is reduced to faults injected on q before Bex_1 and cat.2 to faults on s before Bex_2. Cat.3 is obviously the set of single faults injected on $\hat{s} = \text{Bex}_2(s)$.

1) Cat.1: *A priori* a fault on q must be detected with a consistency check using Bex_1, otherwise the fault would infect all residues of $\hat{q}$ and the single fault model would become unsuitable. The problem is that, by construction of $q = \left|-xyp^{-1}\right|_M$, it is impossible to compute $|q|_{m_R}$ from x, y and p by using only parallel operations. Thus Bex_1 is only used to compute q on $\mathcal{B}' \cup \{m_R\}$.

2) Cat.2: The aim is to obtain $|s|_{m_R}$ which is equal to $\left|\left|tM^{-1}\right|_{M'}\right|_{m_R}$ by definition of s. From $|\hat{q}|_{m_R}$ given by Bex_1, one now easily has $|t|_{m_R} = |xy + \hat{q}p|_{m_R}$. Then t is expressed over the extended base $\mathcal{B} \cup \mathcal{B}' \cup \{m_R\}$ and $t < MM'$. But by construction t is also a multiple of M, i.e. $t = (\mathbf{0}_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'}$. So $sM = \left|tM^{-1}\right|_{M'} M = (\mathbf{0}_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'}$, and $|s|_{m_R} = \left|\left|tM^{-1}\right|_{M'}\right|_{m_R} = \left|tM^{-1}\right|_{m_R}$. So by theorem 2.1, a consistency check using Bex_2 will detect faults of category 2.

3) Cat.3: The fault becomes of category 1 as soon as the output is reused as input of algo.1. Otherwise it is possible to use only one more base extension to verify integrity of $\hat{s}$, because its redundancy is known (cf. III-C).

4) Cat.4: For KWc, a fault injected directly in the Cox unit is not envisaged, because the essential interest of KW extensions is more practical than theoretical. In practice, either the hardware Cox unit is protected thanks to a standard hardware redundancy (cf. [8]), or one can bind one Cox per Rower (cf. [17]). In the last case, a fault on a Cox unit is just a fault on the residue computed by the Rower linked to the faulty Cox. However, it is possible to keep only one Cox. In this case, a fault on Cox could add $\pm M$ for Bex_1 or $\pm M'$ for Bex_2 to the extended value. It would require to use larger bases to avoid overflow, and to deliberately add M to $\text{Bex}_1(q)$ so as to counteract the possible appearance of $\text{Bex}_1(q) = q - M$, which is a problem for detection of faults of category 1 (cf. proof of th. 3.1). About SK extension, a fault in extra-channel m_{sk} is easily detected by the consistency check (cf. th. 2.1).

5) Cat.5: Trivially detected.

Figure 2: Categories of fault in RNSMM

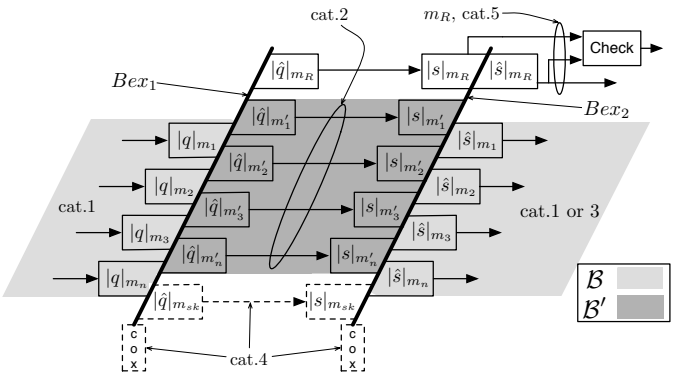


Fig.2 illustrates the redundant version of algo.1 and the categories of fault. By keeping redundancy structure of algo.1 and Fig.2, theorem 3.1 states that faults of category 1 are also detected via a consistency check based on Bex_2.

Theorem 3.1: Given any version of redundant RNSMM defined by a set of hypothesis in Fig.1, it is assumed that at most one single fault of category 1, 2, 4 or 5 is injected during the execution of algorithm 1. Then the fault is injected if, and only if, the consistency check based on Bex_2 and applied to $(\boldsymbol{s}_{\mathcal{B}'}, s_{m_R})_{\mathcal{B}'\cup\{m_R\}}$ fails.

Proof: Detection of faults of cat.2 is ensured by theorem 2.1. It just remains to prove the result for faults of category 1. The consistency check verifies the nullity of $|s_{m_R} - \hat{s}|_{m_R}$, where s_{m_R} is the redundancy of s before Bex_2 and $\hat{s} := \text{Bex}_2(\boldsymbol{s}_{\mathcal{B}'})$. Because M is coprime to m_R, it is the same to verify if $|M(s_{m_R} - \hat{s})|_{m_R} \neq 0$ when a fault is injected. Considering $(0, \ldots, 0, e_i, 0, \ldots, 0)_{\mathcal{B}}$ affecting q, one has $\tilde{q} := \varphi_{\mathcal{B}}^{-1}\left(q_1, \ldots, q_{i-1}, |q_i + e_i|_{m_i}, q_{i+1}, \ldots, q_n\right)_{\mathcal{B}}$. Then even if $t = xy + \hat{\tilde{q}}p$ stays an element of $[\![0, MM'[\![$ it is not a multiple of M anymore. Indeed, $\varphi_{\mathcal{B}}(t) = (0, \ldots, 0, e_i p_i, 0, \ldots, 0)_{\mathcal{B}}$. It follows that

$$s_{m_R} = \left|((0, \ldots, 0, e_i p_i, 0, \ldots, 0)_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'} M^{-1}\right|_{m_R}.$$

But Bex_2 only extends $\boldsymbol{s}_{\mathcal{B}'} = \left|tM^{-1}\right|_{M'} = (\mathbf{0}_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'}/M$. Depending on the chosen set of hypothesis in Fig.1, Bex_2 can be MRC, SK, or KWc. In all cases, $\hat{s} = \boldsymbol{s}_{\mathcal{B}'} + \mu M'$, where possible values of μ are detailed thereafter. Then

$$|\hat{s}|_{m_R} = \left|(\mathbf{0}_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'} M^{-1} + \mu M'\right|_{m_R}.$$

Finally: $|M(s_{m_R} - \hat{s})|_{m_R}$

$$\begin{aligned} &= \left|(\boldsymbol{t}_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'} - (\mathbf{0}_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'} - \mu MM'\right|_{m_R} \\ &= \left|(\boldsymbol{t}_{\mathcal{B}}, \mathbf{0}_{\mathcal{B}'})_{\mathcal{B}} - \delta MM' - \mu MM'\right|_{m_R} = \left|a_i M_i M'\right|_{m_R} \end{aligned}$$

where $a_i = \left|e_i p_i M_i^{-1} M'^{-1}\right|_{m_i} - \delta m_i - \mu m_i$ and $\delta \in \{0, 1\}$ is such that $(\boldsymbol{t}_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'} = (\boldsymbol{t}_{\mathcal{B}}, \mathbf{0}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'} + (\mathbf{0}_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'} - \delta MM'$. Because $m_R \wedge M_i M' = 1$, one just has to verify that $0 < |a_i| < m_R$, i.e. $|a_i|_{m_R} \neq 0$.

1) MRC: Bex_2 exact so $\mu = 0$ and $0 < |a_i| < m_i < m_R$.

2) SK: the context is different of the one of theorem 2.1. Indeed, all residues of $\boldsymbol{s}_{\mathcal{B}'}$ are affected by the fault over q, so SK reconstruction could return $s+\mu M'$ with μ any value in $]\!]-m_{sk}, m_{sk}[\![$. Then, with $m_R > \max(m_{sk} m_i)$, one has $0 < |a_i| < m_{sk} m_i < m_R$.

3) KWc: $\mu \in \{-1, 0\}$. Actually $\delta = 0$ implies $\mu = 0$ and so $0 < |a_i| < m_i$. Indeed $\delta = 0$ means $(\boldsymbol{t}_{\mathcal{B}}, \boldsymbol{0}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'} + (\boldsymbol{0}_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'} = (\boldsymbol{t}_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'} = t$. Moreover, hyp. $\mathcal{H}_{kw}$ still ensure that $t = xy + \tilde{q}p < (1-\alpha_{kw}) MM'$. Then $\boldsymbol{s}_{\mathcal{B}'} = (\boldsymbol{0}_{\mathcal{B}}, \boldsymbol{t}_{\mathcal{B}'})_{\mathcal{B}\cup\mathcal{B}'}/M < (1-\alpha_{kw}) M'$, so $\mu = 0$. ■

Rem. 4: Since randomizations of bases in Leak Resistant Arithmetic technique [10] only use RNSMM to switch between different Montgomery representations of data, it is compliant with the present detection method.

C. Handling faults of category 3

This case may appear if the integrity of $\hat{s}$ expressed in $\mathcal{B}$ must be ensured and if it will not be used again as input of algo.1. By hypothesis there is at most one fault on it, so $\hat{s}$ can be $s + e_i M_i < M$. Moreover $|s|_{m_R}$ is given by the last consistency check.

1) $\mathcal{H}_{mrc}$: a standard consistency check applied to $(\hat{s}, s)_{\mathcal{B}\cup\{m_R\}}$ is possible.

2) $\mathcal{H}_{sk}$: in the same manner, because $|s|_{sk}$ is also known, a SK extension can be used to verify integrity of data, where a redundancy $m_R > \max(m_i)$ suffices.

3) $\mathcal{H}_{kw}$: one easily shows that $\frac{2p}{1-\alpha_{kw}} \leqslant \frac{4p}{1-\Delta_{kw}}$ if, and only if, $\alpha_{kw} \leqslant \frac{1+\Delta_{kw}}{2}$. In practice, it is highly feasible to have $\alpha_{kw} = \frac{1}{2}$. Indeed, by picking up notations of part 5.1 in [6], for given parameters n and r then, if the $trunc$ function keeps at least q most significant bits where $q \geqslant \frac{r}{2} + \log_2(n) - \log_2\left(2^{\frac{r}{2}-1} - n\right)$, then $\Delta_{kw} \leqslant \frac{1}{2}$. For instance, a 4096 bits level of security requires $n = 65$ and $r = 64$ ($rn > 4096$). Thus, $q = 8$ is sufficient. Hence, in practice one can always have condition $2p \leqslant (1-\alpha_{kw}) M$, and a KWc can be used for a consistency check.

IV. Architecture

The goal of this section is to present an architecture implementing RNSMM and compliant both with the fault detection approach proposed here, and with the Leak Resistant Arithmetic (LRA) based on the randomization of the RNS bases [10]. A previous RNS architecture for Montgomery multiplication with Kawamura et al. base extension was proposed in [17]. This architecture is composed by a set of n identical arithmetic cells (rowers) that work in parallel and include a modular adder and multiplier unit, a cox unit (that corrects the base extension) and some memory. An in-depth analysis of the arithmetic cells has been presented in [20], where some other cell architectures have been proposed. In this section, the modifications required to introduce the fault attack protection in the state-of-the-art architectures are analyzed and evaluated.

A. Description of the attacker

The general context of the attack is not the purpose of the paper. The strategy adopted when a fault is detected is at the discretion of the user of the proposed architecture, which simply aims at detecting the fault. A stop of the process in order to avoid any leakage of information is typically a possible choice. The attacker is assumed able to inject one fault. Such type of attack is realistically achieved by a laser shot, that can change a value stored in a register, or temporary change the value of some transistors and lines during a sampling for instance. So the attacker has the ability to hit a specific area in any single cell. Furthermore he could be able to control the moment of the injection.

This paper proposes an architecture resistant against one fault. When an attacker injects several faults, there are two possible scenarios. In first case, they are injected sequentially in different Montgomery multiplications, i.e. at most one fault is injected between two consecutive consistency checks. Then the first fault corresponds to a normal single fault, so this attack is prevented by the proposed approach. In second case, some of them are injected simultaneously between two consecutive consistency checks. The proposed protection approach can be extended in order to protect against this attack, by increasing the number of redundant moduli.

B. Hardware fault model

Contrary to theoretical faults, hardware faults are defined by size of registers (r bits since $2^{r-1} < m_j, m'_j < 2^r$) which output ξ_i or ξ'_i in each rower. The model in [8] for a fault on ξ_i is $\tilde{\xi}_i = \xi_i + e_i \in [\![0, 2^r[\![$. The only problematic case is $m_i \leqslant \tilde{\xi}_i < 2^r$ ($\tilde{\xi}_i < m_i$ is theoretical model).

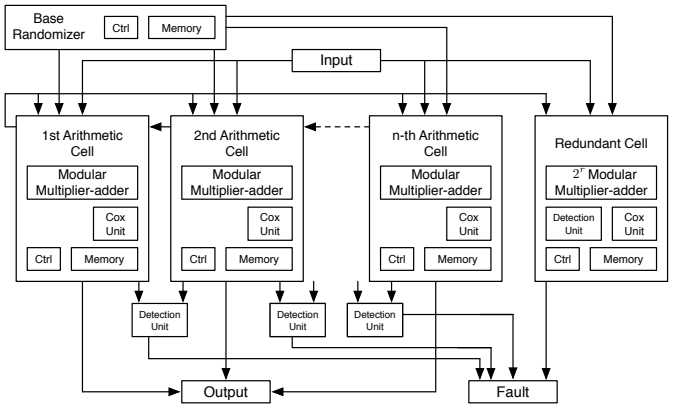
Proposition 4.1: If outputs of rowers are over r bits, then $m_R = 2^r$, $4p \leqslant M$ and $4p \leqslant (1-\alpha_{kw}) M'$ allow to detect single hardware faults.

Proof: It suffices to consider faults of cat. 1 and 2.

Cat.1: $\xi_i + e_i = m_i + f_i$, $f_i \in [\![0, 2^r - m_i[\![\subset [\![0, m_i[\![$. Denoting $\tilde{q} = \varphi_{\mathcal{B}}^{-1}(q_1, \ldots, f_i, \ldots, q_n)$, the crt-sum with $\tilde{\xi}_i$ is crt-sum of $\tilde{q}$ plus $m_i M_i = M$. Because $eval(f_i) \leqslant eval(\tilde{\xi}_i) < eval(f_i) + 1$, Cox of Bex_1 can compute $k_{\tilde{q}} + \delta$, $\delta \in \{-1, 0, 1\}$. $\delta \in \{-1, 0\}$ is usual in KWu case. $\delta = 1$ is not a problem as it just corrects the extra M which appears in crt-sum. Finally, $\hat{\tilde{q}}$ can contain one extra M. It can be shown that if $\Delta_{kw} \leqslant \frac{1}{2}$, this extra M is always corrected if the inequality $\tilde{q} < \Delta_{kw} M$ holds (which is a necessary condition to have $\hat{\tilde{q}} = \tilde{q} + M$ when there is no fault). Thus, one has at least $\hat{\tilde{q}} < 2M$ in all cases, and it suffices to have new conditions $4p \leqslant M$ and $\frac{3p}{1-\alpha_{kw}} \leqslant M'$ to avoid overflow. Otherwise $2M \leqslant \hat{\tilde{q}} < 3M$ could happen. In this case, it is sufficient to have $\frac{4p}{1-\alpha_{kw}} \leqslant M'$. Finally, if $f_i \in \{0, \xi_i\}$ the fault is corrected, else the theoretical model is suitable.

Cat.2: The considered ξ'_j's are those of $\boldsymbol{s}_{\mathcal{B}'}$. If $\tilde{\xi}'_i = \xi'_i + e'_i$ with $e'_i \in [\![m'_i - \xi'_i, 2^r - \xi'_i[\![$, $eval(\xi'_i) \leqslant eval(\tilde{\xi}'_i) < eval(\xi'_i) + 1$ and Cox in Bex_2 could compute $k_s + \delta$ with

Figure 3: General architecture



$\delta \in \{0,1\}$. Finally, $\text{Bex}_2(\tilde{s}) = \sum_{j=1}^{n} \xi'_j M'_j + e'_i M'_i - k_s M' - \delta M' = s + e'_i M'_i - \delta M'$. But $e'_i - \delta m'_i \in [\![-\xi'_i, 2^r - \xi'_i[\![\subset]\!] - m'_i, 2^r[\![\subset]\!] - 2^r, 2^r[\![$. If $e'_i = m'_i$ and $\delta = 1$, the fault is corrected, else it is detected. ■

C. Description of the architecture

The general architecture is presented in Fig.3. The Kawamura et al.'s architecture requires two modifications to be applied with the proposed fault detection procedure. First, an extra redundant cell is required to detect the fault as explained in the previous sections. The cells treat data expressed over r bits. Hence, by prop.4.1 the chosen redundancy is 2^r. Secondly, a new fault detection unit (FDU) (cf. Fig.5) is required to check that when a value is propagated to all the cells they indeed receive the same value (cf. rem.2). Such propagation is executed during base extensions. However, it could be avoided by using a round propagation system [17], where at each step each cell takes the input of the previous cell, and gives its previous input to the subsequent. In this case, a similar check would be required at the end of the rotation, in order to control that each cell receives again the first input, and so that no fault was injected during the rotation. Each FDU compares the input of two cells, so $n-1$ units are required. Furthermore, a FDU is in the redundant cell to apply the consistency check.

LRA [10] (cf. [8] for FPGA implementation) requires an additional unit, the base randomizer, which randomly matches the arithmetic cells with base elements, and distributes data required to calculate the constants matched to the current two random bases. Moreover, the technique requires some small changes in the cells: they will have a different control unit, able to manage the additional operations for the precomputation of the constants (calculated after each randomization), and a different access to the memory, since some data sent by the randomizer unit to the arithmetical cells must be directly stored (e.g. the remainder of $2^r \bmod m_i$), while other values must be processed through additions and multiplications, and then stored.

Figure 4: Modular adder and multiplier unit in a standard (a) and redundant (b) cell

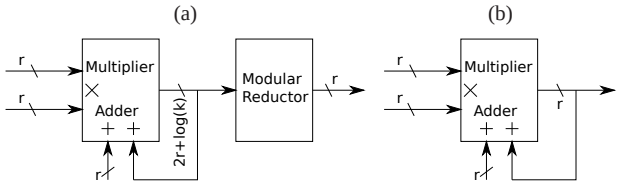
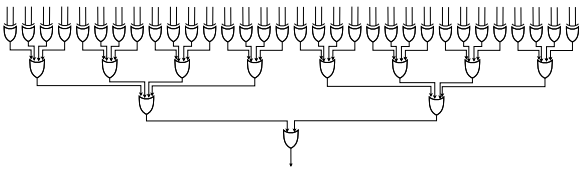


Figure 5: Fault detection unit (FDU) with r=32



The redundant cell has two main differences with respect to the standard cells. The first one is the presence of a FDU. It is used for the consistency check of theorem 3.1. The second one is that the modular adder and multiplier subunit does not work with generic modules, but with a power of 2. Fig.4a and 4b show the modular adder and multiplier units used by the standard (a) and redundant (b) cell. The standard modular adder and multiplier is compliant with the cell proposed in [17], since the accumulation is performed by the same unit that executes additions and multiplications. However, according to another architectural strategy presented in [20], the accumulation could be executed inside the modular reduction unit. The main difference of the adder and multiplier used in the redundant cell is that the modular reduction subunit is not required. Although the multiplier and adder subunit could be identical to the same subunit used in the other cells, a smaller subunit can be used since one additive input for the accumulation is shorter and the computation of the bits larger than 2^r can be avoided.

D. Fault detection

In the previous sections, the mathematical detection of one fault has been described. In order to design an architecture secure against this attack, it is required that the injection of the fault do not have effects not considered in the analysis.

A fault injected in a cell can only modify its output values. If a fault changes a value during its computation or while it is stored, the effects correspond to the single fault injection analyzed previously. Since there is a Cox per cell, even a fault injected in a Cox unit corresponds to a single fault.

During the base extension all the cells receive an input from each cell. The distribution of these values is executed as follows: the first cell gives its result to all the cells, which use it as an input; all the other cells give their result to the previous cell, which stores it for one cycle. Thus the first cell will distribute them, one per cycle. If a fault affects only a part of the propagation line of the output of the first cell (e.g.,

Table I: Basic logic library in CMOS technology [21]

Gate	Area (transistors)	Delay (Inverter)
Inverter	2	1
NAND	4	1.4
NOR	4	1.4
XNOR	12	3.2
NAND3	8	1.8
NAND4	10	2.2
REGISTER	15	4.8

it is injected on a buffer), some arithmetic cells could receive the wrong input. In this case, the fault does not correspond to a single fault. However, it can be easily detected through the FDUs, which check that each cell receives the same input. If during the base extension a fault is injected in a value that must be distributed while it is stored by a cell different from the first, it corresponds to a normal single fault injected before the base extension. Since the same cell works at least for 2 moduli (one per base), the injection of one fault could generate differences on various moduli. In order to avoid this threat, the data of different moduli must be stored in different memories, so the local condition of the fault will avoid the possibility of multiple fault.

A fault injected in a FDU or in the redundant cell is detected. A fault in any part of the redundant cell, but not in its fault detection subunit, is simply a fault of category 5 and so is detected by the consistency check. If the fault is injected in a detection unit, the unit detects the fault.

E. Evaluation

To evaluate the impact of the fault detection, an analytical study based on equivalent gates has been conducted. The adopted model is shown in Tab.I. To reach a fair evaluation, the same parameters used in [17] have been considered: 33 cells, 32 bits per cell, 2 moduli per cell (one per base), $2^r - 2^h < m_i < 2^r,\ \forall i$ and $h = 11$; the architecture of standard arithmetic cells is the one of [17].

A detailed analysis based on equivalent gates on the RNS arithmetic cell proposed in [17] has been presented in [22]. Tab.II shows the delay of the standard and the new units. The time required by one cycle is determined by the slowest pipeline stage. The modular reduction subunit is the slowest in the standard cell, but it is divided in 2 pipeline stages, so the multiplier and adder subunit has the longest delay. The FDUs have a short delay, so they can execute the check in parallel to the normal work flow, after the cycle in which the data have been computed. Therefore, they neither require additional cycles nor increase the delay. The redundant cell executes the same operations of the standard cells with a shorter delay and with less cycles. Hence, it can works in parallel to the standard cell. Therefore, the fault detection technique proposed in this paper do not affect the delay of an exponentiation. However, as explained previously, if a valid result is requested on both the RNS bases, an additional check is required to detect final faults of cat.3. This final check, if it is not executed during the subsequent operations, requires a base extension, so its delay is less than half of a RNSMM. For example, in a RSA implementation, with a N-bit key and a Montgomery ladder for exponentiation, the increase is less than $\frac{1}{2\times N}$ with respect to the total delay.

The fault detection requires additional components, which increase the total area. The area of the standard and the new units is shown in Tab.III. The largest new unit is the 2^r redundant cell. However, also considering that this cell uses a normal multiplier (without avoiding the column over the r-th bit), it is significantly smaller than a standard one, since it does not need a modular reduction unit. Even the total area of all the additional components is lower than one standard cell. Therefore, the total area increase is lower than $\frac{1}{33}$ of the total area, considering that the number of parallel cells is 33. More generally, if the architecture contains n cells, the area increase is about $\frac{1}{n}$ of the total area.

F. Comparison to Guillermin's technique [8]

Guillermin proposes a method to detect the same kind of faults considered in this paper. However, its approach is valid only in the context of Cox-Rower architecture, while the proposed technique is not limited to a specific architecture. Moreover, when associated to a leak resistant arithmetic, the detection of cat.1 faults is not guaranteed. In [8], the detection is achieved by a modified Cox which computes a more precise approximation of $\frac{\xi_i}{m_i}$. For an input xy bounded by νp^2, required conditions on size of bases are: $\nu p \leqslant M$ and $3p2^{r+2} \leqslant M'$. Comparing to $\nu p \leqslant M$ and $\frac{4p}{1-\alpha_{kw}} \leqslant M'$ for the present architecture, the Guillermin's approach requires to increase base $\mathcal{B}'$ by at least one extra standard modulus. Thus extra hardware needed is one standard Rower and a modified Cox for [8], against one redundant Rower only dedicated to 2^r modulus and detection units. So, considering Tab. III, the present technique needs less extra hardware than [8]. Moreover, to add a standard Rower directly impacts computation time of RNSMM. For a 1024 bit RSA-CRT, Guillermin measured that the extra time cost due to his technique is about 5%, for two consecutive 512 bit exponentiations based on Montgomery ladder. So this is two times 1024 executions of RNSMM algorithm. In present technique, the extra delay is less than $\frac{2}{2\times 1024}$ $\sim$0.1%. Finally, the present technique can be generalized for detection of several faults by adding more redundant moduli. The additional delay remains the same.

V. Conclusion

A simple and cheap way to detect single faults injected during a RNS modular multiplication has been presented. The new method has the advantage that no condition is imposed on the choice of bases and base extension techniques, and so is compliant with any RNS Montgomery modular multiplication algorithm. During a base extension, the redundant modulus works in parallel to the cells of the output base. Thus redundant computations and consistency checks

Table II: Delay of Considered Units with $r = 32, k = 33, h = 11$

Unit	Delay (gates)	Delay (inverters)
Standard multiplier	15XNOR+4NAND4+15NAND+4NOT+register	86.6
Standard modular reductor	13XNOR+4NAND4+11NAND+7NOT+register	77.6
2^r multiplier	15XNOR+4NAND4+9NAND+4NOT+register	78.2
Fault detection unit	XNOR+2NAND4+NAND	6.6

Table III: Area Cost for the Considered Units with $r = 32, n = 33, h = 11$

Cell	Multiplier adder unit	Modular reduction unit	Cox	Detection Units	Tot.(kilo transistors)	
Standard	992FA+32HA+140XNOR+69NAND4 +92NAND3+1623NAND+70NOR +1784NOT+70register	562FA+51HA+214XNOR+102NAND4 +136NAND3+1357NAND+107NOR +1390NOT+108register	9FA+9register	0	100	
2^r	992FA+32HA+128XNOR+60NAND4 +80NAND3+1444NAND+64NOR +1704NOT+64register	0	9FA+9register	32XNOR+ 10NAND4+NAND	63	79
Detection	0	0	0	32×(32XNOR+ 10NAND4+NAND)	16	

do not increase the computation time of a RNS modular exponentiation/elliptic curve point addition. Moreover, the integration in an architecture like Cox-Rower's one implies a limited increase of area and a possible extra delay smaller than $\frac{1}{2\times\log_2(exponent)}$ times normal delay.

REFERENCES

[1] H. L. Garner, "The residue number system," in *Papers presented at the the March 3-5, 1959, western joint computer conference*, ser. IRE-AIEE-ACM '59 (Western). New York, NY, USA: ACM, 1959, pp. 146–153.

[2] N. S. Szabo and R. I. Tanaka, *Residue Arithmetic and its application to Computer Technology*. McGraw-Hill, 1967.

[3] J.-C. Bajard, L.-S. Didier, and P. Kornerup, "An RNS Montgomery modular multiplication algorithm," *IEEE Trans. Comput.*, vol. 47, no. 7, pp. 766 –776, jul 1998.

[4] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, no. 170, 1985.

[5] A. Shenoy and R. Kumaresan, "Fast base extension using a redundant modulus in RNS," *IEEE Trans. Comput.*, vol. 38, no. 2, pp. 292 –297, feb. 1989.

[6] S. Kawamura, M. Koike, F. Sano, and A. Shimbo, "Cox-rower architecture for fast parallel Montgomery multiplication." in *EUROCRYPT*, ser. Lecture Notes in Computer Science, B. Preneel, Ed., vol. 1807. Springer, 2000, pp. 523–538.

[7] N. Guillermin, "A high speed coprocessor for elliptic curve scalar multiplications over Fp," in *Proc. of the 12th intern. conf. on Cryptographic hardware and embedded systems*, ser. CHES'10. Springer, 2010.

[8] ——, "A coprocessor for secure and high speed modular arithmetic," Cryptology ePrint Archive, 2011.

[9] S. Duquesne and N. Guillermin, "A FPGA pairing implementation using the residue number system," Cryptology ePrint Archive, 2011.

[10] J. C. Bajard, L. Imbert, P. Y. Liardet, and Y. Teglia, "Leak resistant arithmetic." in *CHES*, ser. Lecture Notes in Computer Science, M. Joye and J.-J. Quisquater, Eds., vol. 3156. Springer, 2004, pp. 62–75.

[11] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Proc. 16th annual intern. conf. on Theory and appli. of crypto. techniques*, ser. EUROCRYPT'97. Springer, 1997.

[12] R. Watson and C. Hastings, "Self-checked computation using residue arithmetic," *Proc. of the IEEE*, vol. 54, no. 12, 1966.

[13] D. Mandelbaum, "Error correction in residue arithmetic," *IEEE Trans. Comput.*, vol. 21, no. 6, pp. 538–545, Jun. 1972.

[14] F. Barsi and P. Maestrini, "Error correcting properties of redundant residue number systems," *IEEE Trans. Comput.*, vol. C-22, no. 3, pp. 307–315, mar 1973.

[15] S. S. S. Yau and Y. C. Liu, "Error correction in redundant residue number systems," *IEEE Trans. Comput.*, vol. C-22, no. 1, pp. 5–11, jan 1973.

[16] J. C. Bajard, L. S. Didier, and P. Kornerup, "Modular multiplication and base extensions in residue number systems," in *Proc. IEEE 15TH Symp. on Comp. Arithmetic*. IEEE, 2001.

[17] H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura, "Implementation of RSA algorithm based on RNS Montgomery multiplication," in *Proc. of the Third Intern. Workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES '01. Springer, 2001.

[18] M. Pohst and H. Zassenhaus, *Algorithmic algebraic number theory*. Cambridge University Press, 1989.

[19] M. H. Etzel and W. K. Jenkins, "Residue number system arithmetic: modern applications in digital signal processing." IEEE Press, 1986.

[20] F. Gandino, F. Lamberti, G. Paravati, J. C. Bajard, and P. Montuschi, "An algorithmic and architectural study on Montgomery exponentiation in RNS," *IEEE Trans. Comput.*, vol. 61, no. 8, pp. 1071 –1083, aug. 2012.

[21] D. Gajski, *Principles of Digital Design*. Prentice-Hall, 1997.

[22] F. Gandino, F. Lamberti, P. Montuschi, and J. C. Bajard, "A general approach for improving RNS Montgomery exponentiation using pre-processing," in *Proc. IEEE 20th Symposium on Computer Arithmetic*, july 2011, pp. 195 –204.