RNS Arithmetic Approach in Lattice-based Cryptography

Accelerating the "Rounding-off" Core Procedure

Jean-Claude Bajard[†‡], Julien Eynard[†‡*], Nabil Merkiche[§†‡], Thomas Plantard[¶]
[†] Sorbonne Universités, UPMC Univ. Paris 6, UMR 7606, LIP6, F-75005, Paris, France
[‡] CNRS, UMR 7606, LIP6, F-75005, Paris, France
[§]DGA/MI, Rennes, France
[¶]University of Wollongong, CCISR, Wollongong, Australia
{jean-claude.bajard,julien.eynard}@lip6.fr, nabil.merkiche@intradef.gouv.fr, thomaspl@uow.edu.au

***Abstract*—Residue Number Systems (RNS) are naturally considered as an interesting candidate to provide efficient arithmetic for implementations of cryptosystems such as RSA, ECC (Elliptic Curve Cryptography), pairings, etc. More recently, RNS have been used to accelerate fully homomorphic encryption as lattice-based cryptogaphy. In this paper, we present an RNS algorithm resolving the Closest Vector Problem (CVP). This algorithm is particularly efficient for a certain class of lattice basis. It provides a full RNS Babai round-off procedure without any costly conversion into alternative positional number system such as Mixed Radix System (MRS). An optimized Cox-Rower architecture adapted to the proposed algorithm is also presented. The main modifications reside in the Rower unit whose feature is to use only one multiplier. This allows to free two out of three multipliers from the Rower unit by reusing the same one with an overhead of 3 more cycles per inner reduction. An analysis of feasibility of implementation within FPGA is also given.**

***Index Terms*—Residue Number System, Lattices, CVP, Round-off, Hardware Implementation, FPGA**

INTRODUCTION

On the one hand, Residue Number Systems (RNS) are supplied with efficient arithmetic. Computations over big data are distributed across small units concurrently and independantly. On the other hand, cryptography area deals with lots of computations over huge numbers. Thus, RNS is naturally interesting to provide efficient implementations of some cryptosystems. There have already been conducted works about acceleration of RSA [1], ECC [2], and pairings [3], [4]. Advantages of RNS were exploited so as to optimize modular multiplication, which is a core operation in such cryptographic functions.

Besides these considerations, modern cryptography makes more and more room for lattice-based cryptography which appears to remain secure in a post-quantum paradigm. In this area, protocols are based on some problems related to Euclidean lattices, such as the Closest Vector Problem (CVP) [5]. For instance, this has been used to create GGH [6] and NTRU [7] cryptosystems. Moreover, interest of lattice-based cryptography has significantly increased since discovery of fully homomorphic encryption scheme by Gentry [8]. Expected for more than 30 years, this scheme is based on ideal lattices used within GGH-like cryptosystems. Despite all those powerful results, huge computational cost of underlying operations over high dimensional lattices limits feasibility of practical implementations. A recent result [9] dealing with this drawback contributes to arithmetical enhancement of lattice-based cryptography by suggesting a specific RNS-MRS algorithm which computes Babai's round-off procedure. However, this approach has the inconvenience to require a conversion into a positional system, like MRS [10], in order to obtain an exact RNS modular reduction.

The present work aims at constructing a full RNS round-off procedure, which will avoid an inconvenient conversion into MRS. In particular, the problem of possible inacurracy of RNS modular reduction operation is addressed by using some of geometrical properties of the involved lattice basis. Finally, a full RNS CVP algorithm is analysed and some considerations about hardware implementation are discussed.

Preliminary definitions on RNS and lattice-based cryptography are given in Section I, along with RNS modular reduction inaccuracy problem. Section II presents a solution to this problem along with an optimised algorithm solving the CVP. Section III details a dedicated architecture, and practical considerations are provided in section V before conclusion.

I. BACKGROUND OVERVIEW

In further discussions, matrices (resp. vectors) will be denoted by uppercase (resp. lowercase) boldfaced letters.

A. RNS and modular reduction

1) Recalls and notations: RNS are based on the Chinese Remainder Theorem (CRT) [10]. This theorem states that given an RNS base $\mathcal{B} = \{m_1, \ldots, m_n\}$ (*i.e.* a set of n pairwise coprime numbers called moduli) there exists a ring isomorphism $\mathbb{Z}/M\mathbb{Z} \xrightarrow{\sim} \mathbb{Z}/m_1\mathbb{Z} \times \ldots \times \mathbb{Z}/m_n\mathbb{Z}$, where $M = \prod_{i=1}^{n} m_i$. Each integer x belonging to the so-called dynamic range $[\![0, M[\![$ of $\mathcal{B}$ is bijectively associated to a tuple of residues $(|x|_{m_i} = x \bmod m_i)_{1 \leqslant i \leqslant n}$. Moreover, additions, subtractions and multiplications can be performed concurrently and independantly on the residues. Exact divisions by

This work has been supported in part by the European Unions H2020 Programme under grant agreement number ICT-644209.

*Author granted by Direction Générale de l'Armement (DGA).

an integer z are also possible as long as $\gcd(z, M) = 1$. In this case, it boils down to the multiplication by $\left|z^{-1}\right|_M$.

The main drawback of RNS is that they are non positional numeral systems. Thus, comparisons require conversion within a positional system, such as Mixed Radix System (MRS) [10]. The modular reduction is also quite a difficult task. State of the art algorithms [11], [12] are based on an adaptation of classical Montgomery's modular reduction [13]. Such techniques use some conversion operations between two RNS bases.

2) Base conversions: A base conversion procedure, denoted $\mathrm{Bex}(\mathcal{B}, \mathcal{B}', x)$, aims at computing the residues in a base $\mathcal{B}'$ of an integer $x \in [\![0, M[\![$ given in a base $\mathcal{B} = \{m_1, \ldots, m_n\}$. To achieve it, the constructive proof of CRT leads to:

$$x = \left(\sum_{i=1}^{n} \left| x_i M_i^{-1} \right|_{m_i} M_i \right) \bmod M. \tag{1}$$

The efficiency of a conversion based on (1) depends on how reduction modulo M is performed, or in other words how the integer $\alpha_x = \left\lfloor \frac{1}{M} \sum_{i=1}^{n} \left| x_i M_i^{-1} \right|_{m_i} M_i \right\rfloor$ is computed. It can be done by adding a redundant modulus to $\mathcal{B}$ [14], or by computing an approximation [15]. A particularly efficient but not accurate conversion consists in not computing α_x at all [12]. That way, a conversion is a single matrix multiplication. But one only obtains residues of $\sum_{i=1}^{n} \left| x_i M_i^{-1} \right|_{m_i} M_i$ in $\mathcal{B}'$.

Another approach is to reconstruct x into a positional mixed radix system [10] and to reduce it into $\mathcal{B}'$. The problem is that computation of MRS coefficients is not absolutely parallel. Therefore, it breaks the efficiency brought by RNS.

3) RNS modular reduction: Implementation of efficient RNS modular reduction is a critical question because of non positional character of RNS. Optimized algorithms are adapted from Montgomery's reduction [13] and use base conversions. The principle is the following. Let $\mathcal{B}$ and $\mathcal{B}'$ be two coprime RNS bases and x be an integer expressed in those two bases and which has to be reduced modulo p. Then by denoting $q = \left| -x/p \right|_M$ we compute $s = \frac{x+pq}{M}$. By construction, $s \equiv xM^{-1} \bmod p$. q is easily computed from residues of x and $\left| -1/p \right|_M$ directly into $\mathcal{B}$. However, the division by M cannot be performed in $\mathcal{B}$ as is. Hence, q is converted into an auxiliary base $\mathcal{B}'$ by using a base conversion procedure. Thus, the exact division by M is performed in $\mathcal{B}'$. However, an issue emphasized by Remark 1.1 rises.

Remark 1.1: As for binary Montgomery reduction, RNS modular reduction gives $\left| xM^{-1} \right|_p + e \times p$ where e is integer. The value of e can depend on the sign of x and on the chosen base conversion technique used to convert $q \in [\![0, M[\![$. If $|x| < Mp$ and if $\mathrm{Bex}(\mathcal{B}, \mathcal{B}', q)$ gives residues of $q + \delta M$ with $\delta \geqslant 0$, then the result satisfies $-p < \frac{x+pq+\delta Mp}{M} < (\delta + 2)\, p$. Thus, $e \in [\![-1, \delta + 1]\!]$. Even if $\delta = 0$, e may still be equal to ± 1. This error can be detected by comparing the result to p and 0. But such a comparison requires a costly RNS-to-MRS conversion.

The computational cost of RNS modular reduction mainly depends on chosen base conversion technique. Because reduction can be inaccurate whatever base conversion is (cf. Rem. 1.1), a new approach is proposed in [12] to accelerate the reduction by not computing reduction modulo M in (1) at all. In this case, the final result belongs to $]\!] - p, (n+1)p[\![$. It can be sufficient for certain situations, as it will be for ours. The general scheme of this reduction is summarized in Alg. 1.

Algorithm 1 $\mathrm{RNSModRed}(x, p, \mathcal{B}, \mathcal{B}')$

Require: x in $\mathcal{B} \cup \mathcal{B}'$ with $|x| < Mp$. $n \stackrel{def}{=} |\mathcal{B}|$.
Ensure: $s = \left| xM^{-1} \right|_p + ep$ in $\mathcal{B}'$, with $e \in [\![-1, n]\!]$.
1: $q \leftarrow \left(-xp^{-1} \right) \bmod M$ #in parallel in $\mathcal{B}$
2: $q' \leftarrow \sum_{i=1}^{n} \left| q_i M_i^{-1} \right|_{m_i} M_i$
3: $s \leftarrow \frac{x+q'p}{M} \bmod M'$ #in parallel in $\mathcal{B}'$
4: **return** s

B. Round-off operation in lattice-based cryptography

1) GGH-like cryptosystems: Some classical lattice-based cryptographic protocols rely on the complexity of computation of a vector of a lattice $\mathcal{L}$ close to any given vector $\mathbf{c} \in \mathbb{Z}^\ell$. Here and further, ℓ denotes the dimension of $\mathcal{L}$. If one knows a nearly orthogonal basis $\mathbf{R} \in \mathbb{Z}^{\ell \times \ell}$ of $\mathcal{L}$, an obvious way to exhibit such a close vector is to compute coordinates of $\mathbf{c}$ relatively to $\mathbf{R}$, and to select the closest integer vector by applying a round-off operation. This approach was introduced by Babai [5]. Consequently, computations of the form $\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil$ appear to be core operations in lattice-based cryptography. For instance, this approach forges GGH cryptosystem principles [6]. Given a fixed integer parameter σ, the plaintext space is $\mathcal{P}_\sigma = [\![-\sigma, \sigma]\!]^\ell$. The secret key is a basis $\mathbf{R}$ of $\mathcal{L}$ verifying the following "Babai's condition":

$$\exists \varepsilon \in]0, \frac{1}{2}[, 0 < \sigma \rho_{\mathbf{R}} < \frac{1}{2} - \varepsilon, \tag{2}$$

where $\rho_{\mathbf{R}}$ denotes the maximum L_1-norm of columns of $\mathbf{R}^{-1}$ and is defined as $\rho_{\mathbf{R}} = \max\{\sum_{i=1}^{\ell} |(\mathbf{R}^{-1})_{i,j}| \mid 1 \leqslant j \leqslant \ell\}$.

This condition ensures that $\lfloor \mathbf{p}\mathbf{R}^{-1} \rceil = 0$ for any $\mathbf{p} \in \mathcal{P}_\sigma$. More precisely, for any $\mathbf{p} \in \mathcal{P}_\sigma$ and any $\mathbf{k}\mathbf{R} \in \mathcal{L}$ (for $\mathbf{k} \in \mathbb{Z}^\ell$), then the Babai condition implies the following equalities:

$$\lfloor (\mathbf{p} + \mathbf{k}\mathbf{R})\mathbf{R}^{-1} \rceil = \lfloor \mathbf{p}\mathbf{R}^{-1} \rceil + \mathbf{k} = \mathbf{k}. \tag{3}$$

The encryption function requires a public basis $\mathbf{B} = \mathbf{U}\mathbf{R}$ (with $\mathbf{U}$ unimodular: $\mathbf{U} \in GL_\ell(\mathbb{Z})$). Consequently, an encryption (4) consists in adding a vector of $\mathcal{L}$ to $\mathbf{p}$. This vector is obtained by using $\mathbf{B}$. Decryption (5) performs the round-off procedure.

$$\mathrm{Enc}(\mathbf{p}) = \mathbf{p} + \mathbf{k}\mathbf{B} = \mathbf{c} \quad (\mathbf{k} \stackrel{\mathcal{R}}{\leftarrow} \mathbb{Z}^\ell) \tag{4}$$

$$\mathrm{Dec}(\mathbf{c}) = \mathbf{c} - \lfloor \mathbf{c}\mathbf{R}^{-1} \rceil \mathbf{R} = \mathbf{p} \tag{5}$$

Correctness of decryption (*i.e.* $\mathrm{Dec}(\mathrm{Enc}(\mathbf{p})) = \mathbf{p}$ for $\mathbf{p} \in \mathcal{P}_\sigma$) is due to (3). Indeed, the vector $\mathbf{k}\mathbf{B}\mathbf{R}^{-1} = \mathbf{k}\mathbf{U}$ is integer. So it can be ruled out from the round-off operation.

2) RNS Babai's round-off: In [9] it is suggested to reduce computational cost of lattice-based cryptoprimitives by using RNS as an efficient arithmetic. More precisely, an RNS-MRS algorithm implementing the round-off is given.

As a first step, the rational formula $\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil$ is turned into an integer formula. If d denotes $\det \mathbf{R}$ then $\mathbf{R}' \stackrel{def}{=} d\mathbf{R}^{-1}$ is

an integer matrix. Consequently, one can rewrite $\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil = \lfloor \frac{\mathbf{c}\mathbf{R}'}{d} + \frac{1}{2}\mathbf{v_1} \rfloor$ where $\mathbf{v_1} \stackrel{def}{=} (1, \ldots, 1) \in \mathbb{Z}^\ell$. This leads to (6), where $\mathbf{d}$ denotes the vector $d \times \mathbf{v_1}$.

$$\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil = \frac{2\mathbf{c}\mathbf{R}' + \mathbf{d} - [(2\mathbf{c}\mathbf{R}' + \mathbf{d}) \bmod (2d)]}{2d} \quad (6)$$

The modular reduction is feasible through RNS Montgomery reduction (Alg. 1) with a main base $\mathcal{B}$ whose product of moduli is denoted M. To do so, one has to compute the following "Montgomery representations":

$$\tilde{\mathbf{R}} = \left(2M\mathbf{R}'\right) \bmod (2d), \ \tilde{\mathbf{d}} = (M\mathbf{d}) \bmod (2d). \quad (7)$$

Since our goal is to provide a full RNS implementation of the Dec function (5) and because $\mathbf{p}$ is in $\mathcal{P}_\sigma = [\![-\sigma, \sigma]\!]^\ell$, computation of (6) can be performed within a single modulus RNS base called m_σ. m_σ has to be greater than $2\sigma + 1$. Thus, $\mathbf{p}$ is recovered from its centered remainder mod m_σ. The reduction mod $2d$ requires a large base $\mathcal{B} = \{m_1, \ldots, m_n\}$ in order to use RNS modular reduction (in our case, $\mathcal{B}'$ is just $\{m_\sigma\}$). As mentioned in Remark 1.1, a problem is that we can possibly obtain an uncompletely reduced value.

Remark 1.2: When the modular reduction in (6) is performed by using $\mathrm{RNSModRed}\left(\mathbf{c}\tilde{\mathbf{R}} + \tilde{\mathbf{d}}, 2d, \mathcal{B}, m_\sigma\right)$, one obtains $(2\mathbf{c}\mathbf{R}' + \mathbf{d}) \bmod (2d) + 2d\mathbf{e}$ with an error vector $\mathbf{e} \in [\![-1, n]\!]^\ell$. Thus, (6) computed together with Alg. 1 gives $\left(\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil - \mathbf{e}\right) \bmod m_\sigma$.

In [9] this error is corrected thanks to a comparison. A modified version of Alg. 1 is used. A redundant modulus m_r besides a base $\mathcal{B}'$ allows to perform a second reduction directly into $\mathcal{B}'$. That way, the final error vector $\mathbf{e}$ belongs to $[\![0, 1]\!]^\ell$. However, it does not completely solve the problem. The solution in [9] is to use a large base $\mathcal{B}'$ which verifies $M' > 4d > \|(2\mathbf{c}\mathbf{R}' + \mathbf{d}) \bmod (2d) + 2d\mathbf{e}\|_\infty$. Then, a conversion into MRS is performed. It allows the comparison with $2d$. Finally, this allows to recover and to correct $\mathbf{e}$.

Our purpose is to avoid a costly conversion into any other number system than RNS. So we choose to compute the modular reduction in (6) via Alg. 1 as is. The next section aims at finding another type of approach to correct the error vector $\mathbf{e}$. To achieve it, some properties of $\mathbf{R}$ will be used.

II. Accelerating the round-off

A. Correcting modular reduction in round-off computation

We aim at recovering $\mathbf{e}$ in order to correct it. It will allow to compute $\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil - \mathbf{e}$ thanks to (6) used together with Alg. 1.

1) Establishing a strategy: Following Remark 2.1 will be a guideline for the new strategy.

Remark 2.1: If one can find an integer γ such that $\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil = 0 \bmod \gamma$ for any ciphertext $\mathbf{c}$ and such that $\mathbf{e} \neq 0 \bmod \gamma$ as soon as $\mathbf{e} \neq 0$, then $\mathbf{e}$ can easily be detected.

A priori there is no reason that such a γ exists. Thus, we consider some integer $\gamma > 0$ and we compute $\lfloor \gamma\mathbf{c}\mathbf{R}^{-1} \rceil$. Since this round-off is still computed via (6) and Alg. 1, we obtain $\lfloor \gamma\mathbf{c}\mathbf{R}^{-1} \rceil - \mathbf{e}$, where $\mathbf{e}$ belongs to $[\![-1, n]\!]^\ell$. Moreover, we have $\lfloor \gamma\mathbf{c}\mathbf{R}^{-1} \rceil = \lfloor \gamma\mathbf{p}\mathbf{R}^{-1} \rceil + \gamma\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil$ (cf. (3)). But now, Babai's condition (2) for $\mathbf{R}$ does not guarantee anymore that $\lfloor \gamma\mathbf{p}\mathbf{R}^{-1} \rceil = 0$. Thus, by computing $\lfloor \gamma\mathbf{c}\mathbf{R}^{-1} \rceil$ we introduce a global error which is $\lfloor \gamma\mathbf{p}\mathbf{R}^{-1} \rceil - \mathbf{e}$. Finally, the result obtained at the end of computation is (8).

$$\lfloor \gamma\mathbf{c}\mathbf{R}^{-1} \rceil - \mathbf{e} = \gamma\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil + \lfloor \gamma\mathbf{p}\mathbf{R}^{-1} \rceil - \mathbf{e} \quad (8)$$

$$\Rightarrow \left(\lfloor \gamma\mathbf{c}\mathbf{R}^{-1} \rceil - \mathbf{e}\right) \bmod \gamma = \left(\lfloor \gamma\mathbf{p}\mathbf{R}^{-1} \rceil - \mathbf{e}\right) \bmod \gamma \quad (9)$$

Consequently, if there exists an integer γ large enough so that $\lfloor \gamma\mathbf{p}\mathbf{R}^{-1} \rceil - \mathbf{e}$ is computable from its residue mod γ, we will be able to correct it. Then, the exact round-off result can be obtained. This is the purpose of following remark.

Remark 2.2: Let modc denote the centered remainders mod γ. If for any $i \in [\![1, \ell]\!]$ the bounds $-\lfloor \frac{\gamma-1}{2} \rfloor \leqslant \lfloor \left(\gamma\mathbf{p}\mathbf{R}^{-1}\right)_i \rceil - \mathbf{e}_i \leqslant \lfloor \frac{\gamma}{2} \rfloor$ hold, then one can retrieve $\lfloor \gamma\mathbf{p}\mathbf{R}^{-1} \rceil - \mathbf{e}$ by:

$$\lfloor \gamma\mathbf{p}\mathbf{R}^{-1} \rceil - \mathbf{e} = \left(\lfloor \gamma\mathbf{p}\mathbf{R}^{-1} \rceil - \mathbf{e}\right) \mathrm{modc}\, \gamma. \quad (10)$$

Therefore, using (8), (9) and (10), $\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil$ can be exactly computed via following equality:

$$\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil = \frac{1}{\gamma}\left(\lfloor \gamma\mathbf{c}\mathbf{R}^{-1} \rceil - \mathbf{e} - \left\{\left(\lfloor \gamma\mathbf{c}\mathbf{R}^{-1} \rceil - \mathbf{e}\right) \mathrm{modc}\, \gamma\right\}\right). \quad (11)$$

From now on, the goal is to find a condition so that γ satisfies the bounds emphasized by Remark 2.2. Then, after computation of $\lfloor \gamma\mathbf{c}\mathbf{R}^{-1} \rceil$ through (6), we will be able to deduce the exact round-off $\lfloor \mathbf{c}\mathbf{R}^{-1} \rceil$ by using (11). Furthermore, it may be noticed that since we need to compute $\lfloor \gamma\mathbf{c}\mathbf{R} \rceil \bmod \gamma$ through (6) in order to obtain (9), γ must be coprime to $2d$.

2) Finding an adequate γ: We need to find an acceptable odd γ for which the bounds described in Remark 2.2 are verified for any $\mathbf{p} \in \mathcal{P}_\sigma$ and any $\mathbf{e} \in [\![-1, n]\!]^\ell$. For that purpose, we use Remark 2.3.

Remark 2.3: If $-\lfloor \frac{\gamma-1}{2} \rfloor - \frac{1}{2} < \left(\gamma\mathbf{p}\mathbf{R}^{-1}\right)_i - \mathbf{e}_i < \lfloor \frac{\gamma}{2} \rfloor + \frac{1}{2}$ for all $i \in [\![1, \ell]\!]$, then Remark 2.2 and (10) hold.

From Remark 2.3 we deduce that γ will depend on ε in Babai's condition (2). Theorem 2.4 clarifies this dependancy.

Theorem 2.4: Let $\mathbf{R} \in \mathbb{Z}^{\ell \times \ell}$ be a basis of lattice $\mathcal{L}$ verifying Condition (2) for a certain $\varepsilon \in]0, \frac{1}{2}[$. Let $\mathbf{e} \in [\![-1, n]\!]^\ell$ be an integer vector (with $n \geqslant 1$), and $\mathbf{c} = \mathbf{p} + \mathbf{k}\mathbf{B}$ with $\mathbf{p} \in \mathcal{P}_\sigma$ and $\mathbf{k} \in \mathbb{Z}^\ell$. Then, for any odd integer γ verifying $\gamma\varepsilon \geqslant n$, the following equality holds:

$$\lfloor \gamma\mathbf{p}\mathbf{R}^{-1} \rceil - \mathbf{e} = \left(\lfloor \gamma\mathbf{c}\mathbf{R}^{-1} \rceil - \mathbf{e}\right) \mathrm{modc}\, \gamma.$$

Proof: Due to Babai's condition, $\mathbf{p}\mathbf{R}^{-1} \in]-\frac{1}{2}+\varepsilon, \frac{1}{2}-\varepsilon[^\ell$. Consequently, any i^{th} component of $(\gamma\mathbf{p}\mathbf{R}^{-1} - \mathbf{e})$ verifies:

$$-\frac{\gamma}{2} + \gamma\varepsilon - n < \left(\gamma\mathbf{p}\mathbf{R}^{-1} - \mathbf{e}\right)_i < \frac{\gamma}{2} - \gamma\varepsilon + 1.$$

Hence, by using Remark 2.3, the bounds $-\lfloor \frac{\gamma-1}{2} \rfloor \leqslant \lfloor \left(\gamma\mathbf{p}\mathbf{R}^{-1} - \mathbf{e}\right)_i \rceil \leqslant \lfloor \frac{\gamma}{2} \rfloor$ hold as soon as γ verifies the following set of conditions:

$$\begin{cases} -\lfloor \frac{\gamma-1}{2} \rfloor - \frac{1}{2} \leqslant -\frac{\gamma}{2} + \gamma\varepsilon - n \\ \frac{\gamma}{2} - \gamma\varepsilon + 1 \leqslant \lfloor \frac{\gamma}{2} \rfloor + \frac{1}{2} \end{cases} \Leftrightarrow \begin{cases} \gamma\varepsilon \geqslant n \\ \gamma\varepsilon \geqslant 1 \end{cases} \Leftrightarrow \gamma\varepsilon \geqslant n \quad (12)$$

Then Remark 2.2 and (9) allow to conclude the proof. ∎

Theorem 2.4 provides a lower bound for acceptable γ's for any basis $\mathbf{R}$ verifying $\sigma\rho_{\mathbf{R}} < \frac{1}{2} - \varepsilon$. This enables to directly

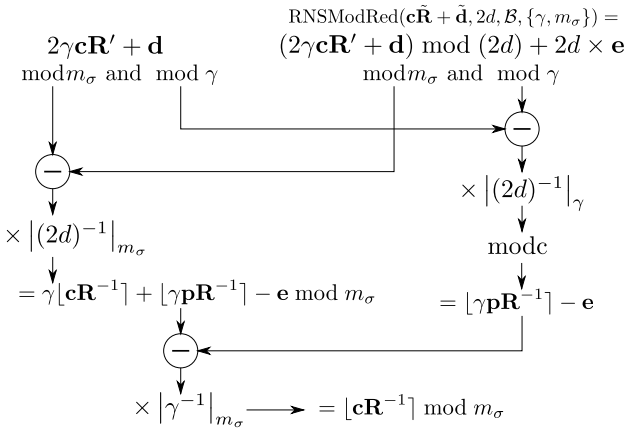

Fig. 1: Computing $\lfloor \mathbf{cR}^{-1} \rceil \bmod m_\sigma$ by formula (6), together with strategy described in Corollary 2.5

deduce the following corollary, which expresses this bound no more in terms of ε but by directly using the value of $\rho_{\mathbf{R}}$. It also details how to use previous theorem to perform an exact round-off through formula (6).

Corollary 2.5: Let $\mathbf{R}$ be a basis of a lattice $\mathcal{L}$ such that $\sigma\rho_{\mathbf{R}} < \frac{1}{2}$, $n \geqslant 1$ an integer (*e.g.* for our purpose, n is the size of RNS base $\mathcal{B}$ involved in the modular reduction, Alg. 1), and m_σ a modulus coprime to $2d$. Let's denote $\gamma_{\mathbf{R},n} \stackrel{def}{=} \lceil \frac{2n}{1-2\sigma\rho_{\mathbf{R}}} \rceil$ and $\mathbf{r} = (2\gamma\mathbf{cR}' + \mathbf{d}) \bmod (2d) + 2d \times \mathbf{e}$ with $\mathbf{e} \in [\![-1, n]\!]^\ell$. For any γ coprime to $2dm_\sigma$ and s.t. $\gamma \geqslant \gamma_{\mathbf{R},n}$, we have:

$$\lfloor \mathbf{cR}^{-1} \rceil \bmod m_\sigma = \left| \left(\frac{2\gamma\mathbf{cR}' + \mathbf{d} - \mathbf{r}}{2d} - \left\{ \frac{2\gamma\mathbf{cR}' + \mathbf{d} - \mathbf{r}}{2d} \operatorname{modc} \gamma \right\} \right) \gamma^{-1} \right|_{m_\sigma} \quad (13)$$

Proof: We fix $\varepsilon = \frac{1}{2} - \sigma\rho_{\mathbf{R}}$. It follows that $\gamma_{\mathbf{R},n} = \lceil \frac{n}{\varepsilon} \rceil = \lceil \frac{2n}{1-2\rho_{\mathbf{R}}} \rceil$ is a lower bound for acceptable odd γ's with respect to theorem 2.4. Moreover, by using (6) and (8), we can write:

$$\frac{2\gamma\mathbf{cR}' + \mathbf{d} - \mathbf{r}}{2d} = \lfloor \gamma\mathbf{cR}^{-1} \rceil - \mathbf{e} = \gamma\lfloor \mathbf{cR}^{-1} \rceil + \lfloor \gamma\mathbf{pR}^{-1} \rceil - \mathbf{e}.$$

Furthermore, previous theorem enables to write $\lfloor \gamma\mathbf{pR}^{-1} \rceil - \mathbf{e} = \frac{2\gamma\mathbf{cR}'+\mathbf{d}-\mathbf{r}}{2d} \operatorname{modc} \gamma$. Hence (13) immediately follows. ■

Given adequate γ with respect to Corollary 2.5, the strategy is to compute the whole formula (6) mod m_σ and mod γ with $\mathrm{RNSModRed}\left(\mathbf{c}\tilde{\mathbf{R}} + \tilde{\mathbf{d}}, 2d, \mathcal{B}, \{\gamma, m_\sigma\}\right)$ where Montgomery representations are now the following ones:

$$\tilde{\mathbf{R}} \stackrel{def}{=} \left(2\gamma M\mathbf{R}'\right) \bmod (2d), \ \tilde{\mathbf{d}} \stackrel{def}{=} (M\mathbf{d}) \bmod (2d). \quad (14)$$

Then, residues modulo γ enable to correct residues modulo m_σ of rounded value $\lfloor \mathbf{cR}^{-1} \rceil$. Fig. 1 illustrates this approach.

B. About the size of γ

GGH-like protocols with parameter $\sigma \in \mathbb{N}$ are based on the matrices $\mathbf{R} \in \mathbb{Z}^{\ell\times\ell}$ verifying $\sigma\rho_{\mathbf{R}} < \frac{1}{2}$. By definition the lower bound $\gamma_{\mathbf{R},n}$ exists for any such matrix $\mathbf{R}$ and any n. It is clear that closer to $\frac{1}{2}$ the $\sigma\rho_{\mathbf{R}}$ is, the bigger $\gamma_{\mathbf{R},n}$ will be.

For implementing an efficient RNS CVP round-off procedure which uses new error correction technique, $\gamma_{\mathbf{R},n}$ should be "reasonably" small, meaning comparatively to word size β of other moduli (*i.e.* $m < \beta$ for all $m \in \mathcal{B}$). Indeed, when $\gamma_{\mathbf{R},n}$ is not larger than β, γ can be a single modulus. Then, the base $\mathcal{B}'$ in Alg. 1 is just the set of two moduli $\{\gamma, m_\sigma\}$. In this case, the needed comparison to recover a centered remainder modulo γ is directly performed on the residue mod γ. Hence, the present technique is particularly efficient for matrices $\mathbf{R}$ such that $0 < \sigma\rho_{\mathbf{R}} < \frac{1}{2} - \frac{n}{\beta}$ (*i.e.* $\gamma_{\mathbf{R},n} < \beta$).

ℓ	128	256	384	512	640	768
σ	3	3	3	3	3	2
max	2003	4775	3637	4059	683	334
min	10	6	6	6	5	4
median	66.5	37.5	31	28.5	21	9
mean diff.	193.7	260.9	162.6	193.6	41.6	13.2

TABLE I: Statistics about $\gamma_{\mathbf{R},1}$ for 100 random LLL-reduced matrices $\mathbf{R}$ for each dim. ℓ

In practice, the constraint $\gamma_{\mathbf{R},n} < \beta$ does not seem to be so restrictive for usual values of $\beta = 2^r$ (*i.e.* $r \geqslant 16$). By analysing basis $\mathbf{R}$ obtained through LLL-reduction of matrices randomly picked up in $[\![-\ell, \ell]\!]^{\ell\times\ell}$ (as suggested in [16]) for some dimensions ℓ from 128 to 768 with respect to $\sigma = 2$ or 3, it appears (cf. Tab. I) that $\gamma_{\mathbf{R},1}$ is most of the time quite "small" (*i.e.* comparatively to β). Then, choosing a lattice basis $\mathbf{R}$ with $\gamma_{\mathbf{R},n} < \beta$ should not weaken the protocol.

C. Full RNS Round-off CVP algorithm

The next part presents a detailed algorithm computing $\mathbf{c} - \lfloor \mathbf{cR}^{-1} \rceil\mathbf{R}$. It is based on Alg. 1 together with (6), and it performs the acceleration technique. To provide an optimized algorithm, all possible precomputations for Alg. 2 are provided in a first part. A radix $\beta = 2^r$ as word size is fixed from now on. All moduli used are constrained to own only one β-digit. So any involved matrix $\mathbf{R}$ is assumed to verify $\gamma_{\mathbf{R},n} < \beta$ with $n = |\mathcal{B}|$. Next, the proposed algorithm is detailed. Finally, a discussion about computational and space costs is provided.

1) Precomputations: Precomputable data (22) are found by expanding involved expressions. We aim at computing $\mathbf{c} - \lfloor \mathbf{cR}^{-1} \rceil\mathbf{R} \bmod m_\sigma$. $\mathbf{c}$ is the only input variable. Due to our previous discussions, the strategy exploits following quantities:

$$\mathbf{q}' = \sum_{i=1}^{n} \mathbf{q}_{m_i} M_i, \ \mathbf{q}_{m_i} = |-(\mathbf{c}\tilde{\mathbf{R}} + \tilde{\mathbf{d}})(2dM_i)^{-1}|_{m_i} \quad (15)$$

$$\mathbf{r} = |2\gamma\mathbf{cR}' + \mathbf{d}|_{2d} + 2d\mathbf{e} = \frac{\mathbf{c}\tilde{\mathbf{R}} + \tilde{\mathbf{d}} + 2d\mathbf{q}'}{M} \quad (16)$$

$$\mathbf{s} = \lfloor \gamma\mathbf{cR}^{-1} \rceil - \mathbf{e} = \frac{1}{2d}\left(2\gamma\mathbf{cR}' + \mathbf{d} - \mathbf{r}\right) \quad (17)$$

First, (15) is obtained in RNS base $\mathcal{B}$. Second, (16) is computed in $\{\gamma, m_\sigma\}$. Then, (17) is obtained modulo γ and m_σ. Finally centered remainder of $\mathbf{s}$ mod γ enables to correct residues in RNS channel m_σ so that $\lfloor \mathbf{cR}^{-1} \rceil \bmod m_\sigma$ can be obtained (cf. (13) in Corollary 2.5).

In $\mathcal{B}$: By using $\tilde{\mathbf{R}}_{m_i}$ and $\tilde{\mathbf{d}}_{m_i}$ in (22) we can re-write (15) with $\mathbf{q}_{m_i} = \left|\mathbf{c}\tilde{\mathbf{R}}_{m_i} + \tilde{\mathbf{d}}_{m_i}\right|_{m_i}$ for all $m_i \in \mathcal{B}$. For base conversion, it will be useful to write the set of n vectors $\mathbf{q}_{m_i}$ as a matrix $\mathbf{Q}$ whose i^{th} row is the vector $\mathbf{q}_{m_i}$.

In $\{\gamma\}$: If we develop right member of (17) modulo γ by using (15) and (16) we obtain:

$$|\mathbf{s}|_\gamma = \left| \mathbf{c}\frac{-\tilde{\mathbf{R}}}{2dM} + \frac{\mathbf{d}}{2d} - \frac{\tilde{\mathbf{d}}}{2dM} - \sum_{i=1}^{n} \frac{\mathbf{q}_{m_i}}{m_i} \right|_\gamma . \quad (18)$$

Due to precomputations (22), the required result modulo γ is:

$$\mathbf{s}_\gamma \stackrel{def}{=} \left(\mathbf{c}_\gamma\tilde{\mathbf{R}}_\gamma + \tilde{\mathbf{d}}_\gamma + \mathbf{h}_\gamma\mathbf{Q}\right) \bmod c\, \gamma = \lfloor \gamma \mathbf{p} \mathbf{R}^{-1} \rceil - \mathbf{e}. \quad (19)$$

In $\{m_\sigma\}$: The one modulus-base $\{m_\sigma\}$ is dedicated to the full computation of $\mathbf{c} - \lfloor \mathbf{c}\mathbf{R}^{-1} \rceil \mathbf{R}$. First, by using (13) from corollary and (17) and (19), it comes that:

$$\left|\mathbf{c} - \lfloor \mathbf{c}\mathbf{R}^{-1} \rceil \mathbf{R}\right|_{m_\sigma} = \left| \mathbf{c} - \frac{1}{\gamma}\left(\mathbf{s} - \mathbf{s}_\gamma\right)\mathbf{R} \right|_{m_\sigma} \quad (20)$$

where $|-\mathbf{s}|_{m_\sigma} = \left| \mathbf{c}\frac{\tilde{\mathbf{R}} - 2\gamma M \mathbf{R}'}{2dM} + \frac{\tilde{\mathbf{d}} - M\mathbf{d}}{2dM} + \sum_{i=1}^{n} \frac{\mathbf{q}_{m_i}}{m_i} \right|_{m_\sigma}$. Thus, precomputed data (22) allow to write:

$$\left|\mathbf{c} - \lfloor \mathbf{c}\mathbf{R}^{-1} \rceil \mathbf{R}\right|_{m_\sigma} = \left| \mathbf{c}\tilde{\mathbf{R}}_\sigma + \tilde{\mathbf{d}}_\sigma + \left(\mathbf{h}_\sigma\mathbf{Q} + \mathbf{s}_\gamma\right)\mathbf{R}_\sigma \right|_{m_\sigma} . \quad (21)$$

$$\begin{aligned}
&\forall m_i \in \mathcal{B},\ \tilde{\mathbf{R}}_{m_i} = \left|\frac{-\tilde{\mathbf{R}}}{2dM_i}\right|_{m_i} ; \tilde{\mathbf{d}}_{m_i} = \left|\frac{-\tilde{\mathbf{d}}}{2dM_i}\right|_{m_i} \\
&\tilde{\mathbf{R}}_\gamma = \left|\frac{-\tilde{\mathbf{R}}}{2dM}\right|_\gamma ; \tilde{\mathbf{d}}_\gamma = \left|\frac{M\mathbf{d} - \tilde{\mathbf{d}}}{2dM}\right|_\gamma ; \mathbf{h}_\gamma = \left(\left|\frac{-1}{m_i}\right|_\gamma\right)_{1\leqslant i\leqslant n} \\
&\tilde{\mathbf{R}}_\sigma = \left|\mathbf{I}_\ell + \frac{\tilde{\mathbf{R}} - 2\gamma M\mathbf{R}'}{2d\gamma M}\mathbf{R}\right|_{m_\sigma} ; \mathbf{R}_\sigma = \left|\frac{\mathbf{R}}{\gamma}\right|_{m_\sigma} \\
&\tilde{\mathbf{d}}_\sigma = \left|\frac{\tilde{\mathbf{d}} - M\mathbf{d}}{2d\gamma M}\mathbf{R}\right|_{m_\sigma} ; \mathbf{h}_\sigma = \left(\left|\frac{1}{m_i}\right|_{m_\sigma}\right)_{1\leqslant i\leqslant n}
\end{aligned} \quad (22)$$

2) Main Algorithm 2: RNS base $\mathcal{B}$ used for the modular reduction (16) verifies $2dM > \|\mathbf{c}\tilde{\mathbf{R}} + \tilde{\mathbf{d}}\|_\infty$ for any input $\mathbf{c}$. Thus we assume that the set of all possible $\|\mathbf{c}\|_\infty$ is bounded by some integer c_∞. $\mathbf{c}$ can own negative coefficients. Finally, $\mathcal{B}$ is assumed to satisfy $M > \ell c_\infty + 1$.

Complexity analysis: Complexity of RNS algorithms is generally expressed in terms of elementary modular multiplications with respect to some moduli m such that $m < \beta = 2^r$. To simplify complexity analysis, $\texttt{EMM}_\beta$ denotes such modular multiplication. The $n+1$ moduli of $\mathcal{B} \cup \{\gamma\}$ are assumed to own only one β-digit. Since bit-size of m_σ and β can differ by orders of magnitude, it is useful to denote β_σ the quantity $2^{\lceil \log_2(m_\sigma) \rceil}$. Binary-to-RNS conversions are based on decomposition of coefficients of $\mathbf{c}$ in base β. Residues of powers of β can be precomputed. That way, the conversion requires $(n+1)\ell \times \lceil \log_\beta(c_\infty) \rceil \texttt{EMM}_\beta + \ell \times \lceil \log_{\beta_\sigma}(c_\infty) \rceil \texttt{EMM}_{\beta_\sigma}$. When this conversion is done, the cost of steps 2 to 11 of RNS CVP algorithm 2 is $\left((n+1)\ell^2 + n\ell\right)\texttt{EMM}_\beta + \left(2\ell^2 + n\ell\right)\texttt{EMM}_{\beta_\sigma}$.

Space efficiency: Precomputed powers of β for binary-to-RNS conversion cost no more than $(n+1)(\lceil \log_\beta(c_\infty) \rceil - 1)$ β-words and $(\lceil \log_{\beta_\sigma}(c_\infty) \rceil - 1)$ β_σ-words. Precomputations (22) represent $(n+1)(\ell^2+\ell) + n$ β-words and $2\ell^2 + \ell + n$ β_σ-words. To compare with a binary approach, each coefficient of $\mathbf{R}^{-1}$ should be computed with a $\lceil \log_2\left(c_\infty(\frac{1}{2} - \sigma\rho_{\mathbf{R}})^{-1}\right) \rceil$ bit precision to ensure no decryption error [6]. We recall that $\gamma_{\mathbf{R},n} \stackrel{def}{=} \lceil n \times (\frac{1}{2} - \sigma\rho_{\mathbf{R}})^{-1} \rceil$ and that our approach is optimal when $\gamma_{\mathbf{R},n} < \beta$. Thus, the binary method requires at most $\lceil \log_2(\frac{c_\infty \beta}{n}) \rceil \times \ell^2$ bits of storage. Tab. II shows some memory ratio between binary storage of $\mathbf{R}^{-1}$ and RNS precomputations where we assume that $c_\infty = d$ and d is set to Hadamard's bound $\ell^{\frac{3}{2}\ell}$ (*e.g.* $\mathbf{R}$ results in a LLL-reduction applied to some random matrix in $[\![-\ell, \ell]\!]^{\ell^2}$, cf. [16]).

3) Comparison with hybrid RNS-MRS approach [9]: In [9], the final computation is also performed modulo m_σ. The same main RNS base $\mathcal{B}$ than ours (*i.e.* $M > \ell c_\infty + 1$) is used for computing modular reduction. The main difference in this other approach is that an auxilairy base $\mathcal{B}'$ with $M' > 4d$ is required. Indeed, the result of reduction is in $[\![0, 4d[\![$, and it must be compared with $2d$ through MRS. The first step of reduction is a fast conversion (like in Alg. 1) from $\mathcal{B}$ to $\mathcal{B}' \cup \{m_r, m_\sigma\}$. Then, a second reduction is based on a conversion from $\{m_r\}$ to $\mathcal{B}' \cup \{m_\sigma\}$ (simple "duplication" of residues mod m_r) in order to reduce the result in $[\![0, 4d[\![$. Montgomery representations differ from (14) but their size is still the same. They are $\tilde{\mathbf{R}} = |2m_r M\mathbf{R}'|_{2d}$ and $\tilde{\mathbf{d}} = |m_r M\mathbf{d}|_{2d}$.

By denoting $k = |\mathcal{B}'|$, *i.e.* $k = \lceil \log_\beta(4d) \rceil$, the extra cost (mem. and comp.) of hybrid approach are due to computations in $\mathcal{B}'$ and to the MRS conversion. In terms of precomputations dedicated to modular reduction, the overhead is $k(\ell^2 + \ell) + (k+1)n$ β-words for adequate matrices and vectors $\tilde{\mathbf{R}}_{m'_j}$, $\tilde{\mathbf{d}}_{m'_j}$, $\mathbf{h}_{m'_j}$ and extra specific values $|m_r^{-1}|_{m'_j}$ for each $m'_j \in \mathcal{B}'$. Size of precomputations modulo m_σ is almost identical.

Algorithm 2 Full RNS Round-off CVP algorithm

Require: input $\mathbf{c}$; data: RNS base $\mathcal{B}$ with $M > (\ell c_\infty + 1)$ and $n = |\mathcal{B}|$, $\gamma \geqslant \gamma_{\mathbf{R},n}$, $m_\sigma \geqslant 2\sigma + 1$, M, γ, m_σ and $2d$ all pairwise coprime, and precomputations (22).
Ensure: $\left(\mathbf{c} - \lfloor \mathbf{c}\mathbf{R}^{-1} \rceil \times \mathbf{R}\right) \bmod c\, m_\sigma$.

```
(c_m)_{m∈B∪{γ,m_σ}} ← Bin_to_RNS(c)        #1st parallel step
for i = 1 to n do                           #2nd parallel step
    Q_{i,*} ← |c_{m_i} R̃_{m_i} + d̃_{m_i}|_{m_i}        #q_{m_i}, (15)
end for
s_γ ← |c_γ R̃_γ + d̃_γ|_γ                    #2nd parallel step; part of (19)
p_{m_σ} ← |c_{m_σ} R̃_σ + d̃_σ|_{m_σ}        #2nd parallel step; part of (21)
(q'_γ ; q'_σ) ← (h_γ ; h_σ) Q ∈ Z^{2×ℓ}      #3rd parallel step
s_γ ← (s_γ + q'_γ) mod c γ                  #4th parallel step; (19)
t_{m_σ} ← |q'_σ + s_γ|_{m_σ}                #5th parallel step; part of (21)
p_{m_σ} ← |p_{m_σ} + t_{m_σ} R_σ|_{m_σ}     #6th parallel step; (21)
return p_{m_σ} mod c m_σ
```

ℓ	64	128	256	512	1024
$r = 24$	+6.9%	+3.2%	+1.5%	+0.7%	+0.3%
$r = 32$	+8.0%	+3.7%	+1.7%	+0.8%	+0.4%
$r = 64$	+12.4%	+5.7%	+2.6%	+1.2%	+0.6%

TABLE II: Memory overhead of RNS precomputations comparatively to binary storage of $\mathbf{R}^{-1}$ for several word-sizes $\beta = 2^r$ and dimensions ℓ.

ℓ	64	128	256	512	1024
$r = 24$	+97.0%	+98.8%	+99.5%	+99.8%	+99.9%
$r = 32$	+95.6%	+98.1%	+99.2%	+99.7%	+99.9%
$r = 64$	+91.2%	+95.8%	+98.1%	+99.2%	+99.6%

TABLE III: Memory overhead of [9] comparatively to present full RNS approach for several word-sizes $\beta = 2^r$ and dimensions ℓ.

ℓ	64	128	256	512	1024
24	+133.5%	+141.9%	+149.1%	+155.9%	+162.3%
32	+123.1%	+130.5%	+136.4%	+141.7%	+146.6%
64	+105.1%	+112.1%	+116.7%	+120.2%	+123.0%

TABLE IV: Number of $\texttt{EMM}_\beta$ overhead of [9] comparatively to present full RNS approach for several word-sizes $\beta = 2^r$ and dimensions ℓ.

The only extra cost is due to $|m_r^{-1}|_{m_\sigma}$ for second conversion. Next, MRS conversion requires to precompute $\frac{k(k-1)}{2}$ residues $|(m_i')^{-1}|_{m_j'}$ for $1 \leqslant i < j \leqslant k$. A similar analysis leads to an extra cost in terms of $\texttt{EMM}_\beta$ given by $(k\ell^2 + k\ell(n+1) + \frac{k(k-1)}{2})\texttt{EMM}_\beta$. Tab. III and IV show these overheads with bounds $c_\infty = d \leqslant \ell^{\frac{3}{2}\ell}$. In this case, $n \sim k$. Thus the RNS-MRS approach requires around twice more moduli. Ratio in Tab. IV are clearly greater than 2 because of MRS conversions.

III. Towards hardware implementation

In order to use concurrency properties of RNS, a feasibility study about efficient and dedicated architecture for hardware implementations on FPGA is led. The chosen architecture is based on Cox-Rower architecture proposed in [15], and in [17] with several modifications. We also explain how the main Algorithm 2 fits with such type of architecture.

A. Cox-Rower Architecture

In [15], the Cox-Rower architecture was proposed along with a new approach for base conversion. It has been improved several times [2]–[4], [17]–[19]. Its structure consists in a Sequencer, a Cox and a set of Rower units. Rowers deal with the core of RNS computations. They compute modular sum-of-products $\sum_{i=1}^{n} a_i b_i \mod m_i$. A Rower is divided into 2 stages. The first one consists in multiplication of 2 elements, and the second one is in charge of reducing and accumulating elements from first stage as shown in Fig. 2. Efficient algorithms for reduction inside the Rower are proposed in [2], [3], [17], [18]. They take advantage either by the pseudo-Mersenne form of the moduli, which is $m_i = 2^r - \mu_i$ with $\mu_i < 2^{r/2}$, allowing a direct reduction, or by using a second level of Montgomery reduction as proposed in [17] with $\gcd(m_i, 2^r) = 1$.

B. Proposed Architecture

Our architecture is based on the one proposed in [2], [17], [18], with several modifications. The Cox unit is unnecessary for our purpose, as Alg. 2 does not reduce the base conversion. So as to decrease the area of a Rower, 2 multipliers out of 3 are removed from the Rower for a cost of 3 extra cycles to execute the reduction inside it. Indeed, one of the most time-consuming operation in RNS is the base conversion procedure which is a computation of the form $\sum_{i=1}^{n} a_i b_i \mod m_i$. Moreover, the most time-consuming operations in Alg. 2 are the matrix-vector products. By using a second level of Montgomery reduction coupled with lazy-reduction [20], it is possible to reduce a sum of products only once. Hence, the second stage in charge of reduction can be deleted by reusing first stage. Fig. 3 shows the proposed implementation for Rower Unit.

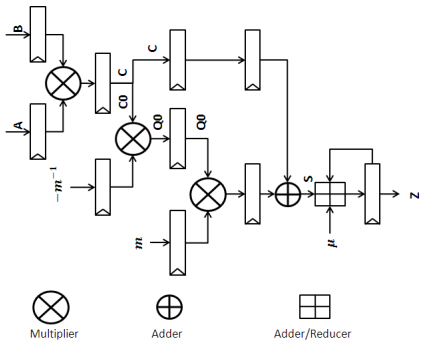

Fig. 2: ALU's Rower

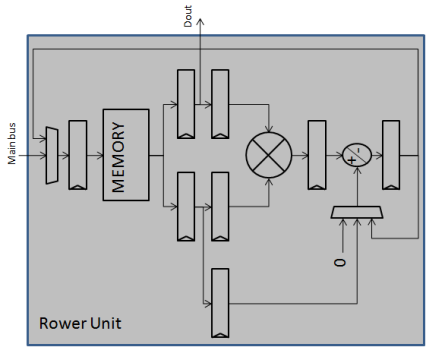

Fig. 3: Proposed Rower architecture

From lines 5 to 11 of Alg. 2, only 2 Rower units are required. To compute the modc function (line 8), two tricks are used to make it easy in hardware without using signed numbers. First, one can see that $\mathbf{s}_{\gamma,i} \bmod^{\mathrm{c}} \gamma = \begin{cases} \mathbf{s}_{\gamma,i} & \text{if } \mathbf{s}_{\gamma,i} \leqslant \lfloor \gamma/2 \rfloor \\ \mathbf{s}_{\gamma,i} - \gamma & \text{otherwise} \end{cases}$. This can be used if the comparison is easy to perform. Thus, our second trick is to choose $\gamma = 2\alpha - 1$ such that $\alpha = 2^\theta$. It makes the comparison easy to be done in hardware (we just need to check the θ^{th} bit). Moreover, these 2 Rower units are used to compute respectively mod m_σ and mod γ. Because m_σ is small ($\leqslant 7$ as $\sigma \in \{2, 3\}$), and because of the trick for γ, we dedicate two Rower units to computations mod m_σ and mod γ. The γ Rower unit remains the same than the one in Fig. 3 but with the test of the θ^{th} bit in order to subtract or not γ.

Alg. 3 details the computation of an inner reduction by using the proposed Rower implementation. From line 1 to 4, we multiply and accumulate the result. Then, the reduction is performed at steps 5 and 6. Because of the Rower unit pipeline, the computation cannot be done directly. Thus, we need to wait 2 depth of the pipeline. However, as usually in cryptography, most of computations can be pipelined. Thus, no

idle cycles are induced. We can show that the computations in Alg. 2 can be done with a 100% occupancy of the pipeline. In order to fully compute Alg. 2 with the proposed architecture, the residues have to be reduced from $2m_i$ to m_i before they go to the main bus (as we need the exact value during the base conversion procedure). This last reduction can be done by a module added in the main bus. It reduces the residues going through the bus.

Algorithm 3 $\mathrm{InnerReduction}(a, b, m)$

Require: A set of $a, b \in (\mathbb{Z}/2m\mathbb{Z})^k$, s parameter for lazy-reduction, $-m^{-1} \mod 2^{r+s}$ and d the pipeline depth
Ensure: $t \equiv \sum_{i=1}^{k} a_i b_i 2^{-(r+s)} \mod m$, $0 \leqslant t < 2m$
1: $t \leftarrow a_1 \times b_1$ #Cycle 1: Multiply
2: **for** $i \leftarrow 2$ to k **do** #Cycle 2 to k
3: $t \leftarrow t + a_i \times b_i$ #Mult. and acc.
4: **end for**
5: $q \leftarrow (t \times -m^{-1}) \mod 2^{r+s}$ #Cycle $k+1+d$: Mult. and get $r+s$ LSB
6: $t \leftarrow \frac{t+q\times m}{2^{r+s}}$ #Cycle $k+1+2d$: Mult., cycle $k+2+2d$: add and get $r+1$ MSB
7: **return** t

C. Full RNS CVP Algorithm Implementation

Alg. 4 gives the procedure to compute lines 2 to 4 of Alg. 2 by using the Rower unit detailed previously. Most of the precomputations have to be done using the inner Montgomery representation in the Rower unit (*i.e.* precomputed values have to be multiplied by 2^{r+s} where r the radix size and s enables the use of lazy reduction). Other steps of Alg. 2 are not detailed but are executed in a similar way. The point is that all computations are modular sums of products, fitting well with Cox-Rower architecture. As emphasized in comments of main Alg. 2, there are 6 main steps which do not overlap.

Features of the Cox-Rower architecture presented here match GPGPU and multi-core CPU architecture purposes. More generally, practical considerations that we make are essentially due to the well flexibility of main algorithm. Since it involves only RNS and fast base conversions, parallelization is easily achieved. In next section, we analyse feasibility of FPGA implementation. The efficiency of such platforms has been proven for RNS implementations.

IV. About feasibility of implementation on FPGA

Estimations about performances and area for some dimensions will be given. Parameters are determined by a radix size $\beta = 2^r$, and by Hadamard's bound for $c_{\mathscr{O}}$ and d.

A. Performance and Memory Analysis

With $n+2$ parallel Rowers, binary-to-RNS conversion requires $\lceil \log_\beta(c_{\mathscr{O}}) \rceil - 1 < n$ cycles when residues of powers of β are precomputed. Table V details the number of cycles needed to execute Alg. 2, as well as the number of precomputed elements required for each of the 6 main steps of Alg. 2. For instance, we can notice that lines 5 and 6 completely overlap with the computation of $\mathbf{Q}$ detailed in Alg. 4.

Algorithm 4 Full RNS CVP Detailed: $\mathbf{Q}$ computation step

Require: Residues of $\mathbf{c}$ in $\mathcal{B}$ and precomputations among (22)
Ensure: $\mathbf{Q} \in \mathbb{Z}^{n\times\ell}$ s.t. $\mathbf{Q}_{i,*} = (|\mathbf{c}\tilde{\mathbf{R}} + \tilde{\mathbf{d}}|_{m_i})$ for $1 \leqslant i \leqslant n$
1: **for** $i \leftarrow 1$ to n **do** #concurrently in n Rowers
2: $\mathbf{Q}_{i,*} \leftarrow (\mathbf{0})^\ell$
3: **for** $y \leftarrow 1$ to ℓ **do** #cols of $\tilde{\mathbf{R}}_{m_i}$
4: **for** $x \leftarrow 1$ to ℓ **do** #rows of $\tilde{\mathbf{R}}_{m_i}$
5: $\mathbf{Q}_{i,y} \leftarrow \mathbf{Q}_{i,y} + \mathbf{c}_{m_i,x} \times \left(\tilde{\mathbf{R}}_{m_i}\right)_{x,y}$
6: **end for**
7: $\mathbf{Q}_{i,y} \leftarrow \mathbf{Q}_{i,y} + \left(\tilde{\mathbf{d}}_{m_i}\right)_y$ #last add.: line 4 of Alg. 2
8: **end for** #this loop: $(\ell+1)\times\ell$ cycles
9: **for** $x \leftarrow 1$ to ℓ **do** #reduc. step, no idle cycle here
10: $q_{m_i,x} \leftarrow \mathbf{Q}_{i,x} \times -m_i^{-1} \mod 2^{r+s}$
11: **end for** #this loop: ℓ cycles
12: **for** $x \leftarrow 1$ to ℓ **do** #no idle cyc. when pipeline depth $< \ell$
13: $\mathbf{Q}_{i,x} \leftarrow \frac{\mathbf{Q}_{i,x} + q_{m_i,x}\times m_i}{2^{r+s}}$
14: **end for** #this loop: 2ℓ cycles
15: **end for** #this loop: $(\ell+4)\times\ell$ cycles
16: **return** $\mathbf{Q}$

lines of Algorithm 2	Cycles	Memory used (nb of words)
1	$n\ell$	$(n+1)(n+2)$
2 to 6	$(\ell+4)\ell$	$(n+2)((\ell+1)\ell+2)$
7	$(n+3)\ell$	$2(n+2)$
8	$\ell+3$	2
9	$\ell+3$	2
10	$(\ell+4)\ell$	$(\ell+1)\ell+2$
Total	$(2\ell+2n+13)\ell$ $+6$	$(n+2)(n+\ell^2+\ell+1)$ $+\ell^2+\ell+2n$

TABLE V: Performance and Memory Size

(ℓ, n)	Mem./Rower (Elts)	Area/Rower (LUTs/Regs/DSP/BRAM)	Freq. MHz	Tot. Area (estimation)
$(64, 16)$	4162	19 (30/17/2/15)	400	304 (480/272/32/240)
$(128, 38)$	16514	19 (30/17/2/56)	300	722 (1140/646/76/2128)

TABLE VI: Results of P&R for ML510 Eval. board (Virtex5)

(ℓ, n)	Mem./Rower (Elts)	Area/Rower (LUTs/Regs/DSP/BRAM)	Freq. MHz	Tot. Area (estimation)
$(64, 16)$	4162	20 (60/17/2/15)	468	320 (960/272/32/240)
$(128, 38)$	16514	20 (60/17/2/56)	415	760 (2280/646/76/2128)

TABLE VII: Results of P&R for KC705 Eval. board (Kintex7)

B. Implementation

We focus on FPGA SRAM devices. Indeed, FPGA are good candidates for hardware implementation as they embed DSP blocks as well as Block RAM. Moreover, the Rower unit proposed in previous section easily fits in Block RAM and DSP blocks, without adding too many LUTs or registers from the FPGA (except for multiplexers and control). Thus, good performances can be reached in terms of frequency. We implement the Rower unit in Xilinx Virtex-5 FPGA family (Tab. VI) as well as Kintex-7 (Tab. VII) and estimate overall area and performance. We take $r = 24$ as bit-size of radix by cascading 2 DSP blocks in order to have few more bits for lazy reduction (10 bits). The results show good perspective to

implement lattice on FPGA, but a memory bottleneck rapidly appears. Indeed, for $\ell = 128$, there are not enough BRAM in both devices (Virtex-5 and Kintex-7) to store precomputed data.

C. Discussion

In previous part, we have shown that hardware implementation of lattices can be done using RNS and Algorithm 2. Adding a margin error of 25% to the maximal frequency of the design, one can easily deduce that we need around $20\mu s$ to compute a close lattice vector in dimension $\ell = 64$, which can be implemented on FPGA as is. For dimension 128, too much precomputations have to be stored. Thus, it could be interesting to work on specific architectures such as clusters of FPGA, or with a DMA for quick changes of memorized values. Nevertheless, our hardware implementation shows a real good trade-off for all existing RNS architecture in terms of footprint or reachable frequency [2]–[4], [17].

V. Conclusion

A method for implementing the Babai round-off procedure in RNS has been proposed. We provide the first full RNS implementation of an algorithm solving the CVP (*i.e.* without using MRS). The efficiency of this technique depends on the secret lattice basis $\mathbf{R}$ which must verify some condition connected to RNS word size. Nevertheless, by analysing several random matrices verifying Babai's condition, the new requirement does not seem restrictive when considering practical RNS word sizes. An architecture for hardware implemention has been detailed. It is based on a Cox-Rower like architecture which is traditionally used for RNS implementations. Some modifications are suggested in order to fit with specific features of lattice-based algorithms. We have given an analysis of FPGA implementation for dimensions 64 and 128. Despite some memory bottlenecks, the results highlight the viability of our approach in terms of flexibility. Indeed, the principles of our implementation match GPGPU and multi-core CPU features. It reinforces the fact that the RNS is an excellent candidate for practical implementation of lattice-based protocols.

We fully realize that 64 and 128 are not dimensions high enough to allow lattice-based cryptography with secure parameters. Nevertheless, this implementation should be seen as a proof of concept. It opens the door to multiple directions such as approaches by blocks and pipeline, or by using FPGA clusters or eventually multi-core CPU's or GPU's. All those future works will use as their core this new proposition.

References

[1] N. Guillermin. (2011, Jul.) A coprocessor for secure and high speed modular arithmetic. Cryptology ePrint Archive, Report 2011/354. [Online]. Available: http://eprint.iacr.org/2011/354

[2] ——, "A High Speed Coprocessor for Elliptic Curve Scalar Multiplications over GF(p)," in *Cryptographic Hardware and Embedded Systems, CHES 2010*, ser. Lecture Notes in Computer Science, S. Mangard and F.-X. Standaert, Eds. Springer Berlin Heidelberg, 2010, vol. 6225, pp. 48–64. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15031-9_4

[3] R. C. C. Cheung, S. Duquesne, J. Fan, N. Guillermin, I. Verbauwhede, and G. X. Yao, "FPGA Implementation of Pairings Using Residue Number System and Lazy Reduction," in *Proceedings of the 13th International Conference on Cryptographic Hardware and Embedded Systems*, ser. CHES'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 421–441. [Online]. Available: http://dl.acm.org/citation.cfm?id=2044928.2044966

[4] G. X. Yao, J. Fan, R. C. Cheung, and I. Verbauwhede, "Faster Pairing Coprocessor Architecture," in *Proceedings of the 5th International Conference on Pairing-Based Cryptography*, ser. Pairing'12. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 160–176. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36334-4_10

[5] L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1–13, 1986. [Online]. Available: http://dx.doi.org/10.1007/BF02579403

[6] O. Goldreich, S. Goldwasser, and S. Halevi, "Public-key cryptosystems from lattice reduction problems," in *Advances in Cryptology CRYPTO '97*, ser. Lecture Notes in Computer Science, J. Kaliski, BurtonS., Ed. Springer Berlin Heidelberg, 1997, vol. 1294, pp. 112–131. [Online]. Available: http://dx.doi.org/10.1007/BFb0052231

[7] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte, "NTRUSign: Digital Signatures Using the NTRU Lattice," in *Topics in Cryptology CT-RSA 2003*, ser. Lecture Notes in Computer Science, M. Joye, Ed. Springer Berlin Heidelberg, 2003, vol. 2612, pp. 122–140.

[8] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," in *Proc. of the Forty-first Annual ACM Symp. on Theory of Computing*, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 169–178.

[9] J.-C. Bajard, J. Eynard, N. Merkiche, and T. Plantard, "Babai round-off CVP method in RNS: Application to lattice based cryptographic protocols," in *Integrated Circuits (ISIC), 2014 14th International Symposium on*, Dec 2014, pp. 440–443.

[10] N. S. Szabó and R. I. Tanaka, *Residue arithmetic and its applications to computer technology*. New York : McGraw-Hill, 1967, based on the authors' Report on residue (modular) arithmetic survey.

[11] K. C. Posch and R. Posch, "Modulo reduction in residue number systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 6, no. 5, pp. 449–454, May 1995.

[12] J.-C. Bajard, L.-S. Didier, and P. Kornerup, "Modular multiplication and base extensions in residue number systems," in *Computer Arithmetic, 2001. Proceedings. 15th IEEE Symposium on*, 2001, pp. 59–65.

[13] P. L. Montgomery, "Modular Multiplication without Trial Division," *Math. of Computation*, vol. 44, no. 170, pp. 519–521, 1985.

[14] P. P. 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. [Online]. Available: http://dx.doi.org/10.1109/12.16508

[15] S. Kawamura, M. Koike, F. Sano, and A. Shimbo, "Cox-Rower Architecture for Fast Parallel Montgomery Multiplication," in *Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques*, ser. EUROCRYPT'00. Berlin, Heidelberg: Springer-Verlag, 2000, pp. 523–538.

[16] D. Micciancio, "Improving Lattice Based Cryptosystems Using the Hermite Normal Form," in *Cryptography and Lattices*, ser. Lecture Notes in Computer Science, J. H. Silverman, Ed. Springer Berlin Heidelberg, 2001, pp. 126–145.

[17] J.-C. Bajard and N. Merkiche, "Double Level Montgomery Cox-Rower Architecture, New Bounds," in *13th Smart Card Research and Advanced Application Conference, Paris, France*, 2014 Nov. 5-7.

[18] H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura, "Implementation of RSA Algorithm Based on RNS Montgomery Multiplication," in *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, ser. Lecture Notes in Computer Science, vol. 2162. Springer, 2001, pp. 364–376.

[19] K. Bigou and A. Tisserand, "Improving Modular Inversion in RNS Using the Plus-Minus Method," in *CHES*. Springer, 2013, pp. 233–249.

[20] S. Gueron, "Enhanced Montgomery Multiplication," in *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, ser. Lecture Notes in Computer Science, vol. 2523. Springer, 2002, pp. 46–56.