HyPoRes: An Hybrid Representation System for ECC

Paulo Martins, Jérémy Marrez, Jean-Claude Bajard, Leonel Sousa

HyPoRes: An Hybrid Representation System for ECC

Paulo Martins*, Jérémy Marrez†, Jean-Claude Bajard†, Leonel Sousa*
*INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
paulo.sergio@netcabo.pt, las@inesc-id.pt
†Sorbonne Université, CNRS, LIP6, Paris, France
jeremy.marrez@lip6.fr, jean-claude.bajard@lip6.fr

***Abstract*—The Residue Number System (RNS) is a numeral representation enabling for more efficient addition and multiplication implementations. However, due its non-positional nature, modular reductions, required for example by Elliptic Curve (EC) Cryptography (ECC), become costlier. Traditional approaches to RNS modular reduction resort to the Montgomery algorithm, underpinned by large basis extensions. Recently, Hybrid-Positional Residue Number Systems (HPRs) have been proposed, providing a trade-off between the efficiency of RNS and the flexibility of positional number representations. Numbers are represented in a positional representation with the coefficients represented in RNS. By crafting primes of a special form, the complexity of reductions modulo those primes is mitigated, relying on extensions of smaller bases. Due to the need of crafting special primes, this approach is not directly extensible to group operations over currently standardised elliptic curves. In this paper, the Hybrid-Polynomial Residue Number System (HyPoRes) is proposed, enabling for improved modular reductions for any prime. Experimental results show that the modular reduction of HyPoRes, although at most 1.4 times slower than HPR for HPR-crafted primes, is up to 1.4 times faster than a generic RNS approach for primes of ECC standards.**

***Index Terms*—Residue Number System, Elliptic Curve Cryptography, Modular Arithmetic**

I. Introduction

The RNS has found extensive application on cryptographic systems [1], since it reduces the complexity of long-integer additions and multiplications. However, due to the non-positional nature of RNS, operations such as modular reductions and comparisons are more difficult to implement in an efficient manner. The complexity of most RNS-based modular multiplications is dominated by RNS basis extensions [2], [3]. In [4], this problem is mitigated through a mixed positional-RNS representation, called Hybrid-Positional Residue Number System (HPR). Numbers are expressed in a positional representation with the coefficients represented in RNS. This approach reduces the size of the RNS bases, thus reducing the complexity of basis extensions, while still benefiting from the arithmetic independence of RNS channels.

Bigou and Tisserand have applied [4] to ECC by crafting special primes with the form $P = B_1^n - \beta$, where B_1 corresponds to the dynamic range of an RNS basis, n is the number of positional digits and β is a small integer. This enables efficient reductions modulo P. After a product, the Most Significant Digits (MSDs) are multiplied by β and added to the Least Significant Digits (LSDs). Then, the size of the coefficients is reduced through carry propagation. Since carry propagation requires computing a division and flooring, basis extensions are still required, but with bases of a smaller size than a generic RNS approach.

While the methods described in [4] are an efficient approach to ECC, one does not often have the opportunity to choose the underlying prime P since these are standardised for most cryptographic applications. Herein, we propose the Hybrid-Polynomial Residue Number System (HyPoRes) that is of an hybrid nature, but allows for efficient reductions for any prime. We achieve so by decoupling the positional representation radix from the RNS dynamic range. After a product, one can still multiply the MSDs by a small constant and add them to the LSDs. However, carry propagation is no longer possible. Hence, to reduce the size of the coefficients, a Montgomery reduction that exploits a redundant representation of zero is used.

Since the Montgomery reduction is more computationally expensive than carry propagation, experimental results show that HyPoRes is at most 1.4 times slower than the HPR. Nevertheless, while the HPR does not support primes currently standardised for ECC, the proposed approach is applicable for any prime, achieving speed-ups of up to 1.4 when compared to generic RNS approaches. These results show a connection between the strength of the assumptions a representation relies on and its efficiency. First, the HPR restricts itself to primes of a special form and achieves the best performance. Second, the HyPoRes relies on the precomputation of values that require one to know the factorisation of the underlying modulus, producing good performance. Finally, RNS generic approaches make no assumption about the underlying modulus, achieving reasonable performance.

The remaining of the paper is organised as follows. Section II provides the necessary background on modular arithmetic, the RNS and lattices to build the proposed HyPoRes. The proposed representation system is described in Section III. Section IV presents the related art. The proposed approach is compared with the related art in Sections V and VI from theoretical and practical perspectives, respectively. Representational techniques to improve resistance against Side-Channel Attacks (SCAs) are introduced in Section VII. Finally, Section VIII concludes the paper.

II. Background

In this section, fundamental results about modular arithmetic, the RNS and lattices are reviewed. In this paper, $[\cdot]_P$ is used to denote the centred remainder of the division by P and lcm the least common multiple. Note that the techniques herein proposed can be slightly modified to handle positive instead of centred remainders.

A. Modular Arithmetic

Most branches of cryptography deal with modular arithmetic. In such a system, two integers $a, b \in \mathbb{Z}$ are said to be congruent if their difference $(a - b)$ is divisible by a modulus P. This is denoted as $a = b \bmod P$. Furthermore, we denote by $\mathbb{Z}_P$ the set of all congruency classes modulo P. After each addition $(a + b)$ or multiplication $(a \times b)$ in $\mathbb{Z}_P$, the results c are mapped to a smaller integer congruent with c modulo P to reduce memory requirements and the computational cost of further operations. Still, in ECC, P is a large prime, typically hundreds of bit wide, and there is a need to map operations modulo P to smaller moduli $b_{i,j}$ that are compatible with current computer architectures. To do so efficiently, the representation system herein proposed requires precomputing n^{th} roots in $\mathbb{Z}_P^\times$. Lemma II.1 states under which conditions a value r has an n^{th} root in $\mathbb{Z}_P^\times$, i.e. there exists $x \in \mathbb{Z}_P^\times$ such that $[x^n]_P = r$. This root may be computed with [5]. Moreover, herein $a^{-1} \bmod P$ and $1/a \bmod P$ are both used to denote the modular inverse of a modulo P, i.e. the integer $-P/2 \leq x < P/2$ such that $xa = 1 \bmod P$.

Lemma II.1. *$r \in \mathbb{Z}_P^\times$ has an n^{th} root modulo P, where P is a prime number, iff $[r^t]_P = 1$ for $\operatorname{lcm}(n, P-1) = nt$.*

Proof. When $\operatorname{lcm}(n, P-1) = n(P-1)$, $\gcd(n, P-1) = 1$ and $[r^t]_P = \left[r^{P-1}\right]_P = 1$. A root of r can be computed as $\left[r^{[1/n]_{P-1}}\right]_P$. Otherwise, let g be a generator of $\mathbb{Z}_P^\times$. Then $r = g^u \bmod P$ for some $u \in \mathbb{Z}$. $r^t = g^{tu} = 1 \bmod P$ iff $P - 1$ divides tu. In this case,

$$\frac{tu}{P-1} = \frac{ntu}{n(P-1)} = \frac{\operatorname{lcm}(n, P-1)/(P-1)u}{n} \in \mathbb{Z},$$

and so n divides u since $\operatorname{lcm}(n, P-1)/(P-1)$ does not. □

Modular arithmetic extends naturally to polynomials. Two polynomials in $\mathbb{Z}[X]/f(X)$ are congruent when their difference is divisible by $f(X)$; and in $\mathbb{Z}_P[X]/f(X)$ when their difference is divisible by $f(X)$ or their coefficients are congruent in $\mathbb{Z}_P$.

B. RNS

The Chinese Remainder Theorem (CRT) states that for coprime integers $b_{1,0}, \ldots, b_{1,h_1-1}$ and for $B_1 = b_{1,0} \times \ldots \times b_{1,h_1-1}$, the ring $\mathbb{Z}_{B_1}$ is isomorphic to $\mathbb{Z}_{b_{1,0}} \times \ldots \times \mathbb{Z}_{b_{1,h_1-1}}$ with the following map:

$$\begin{aligned} g \quad &: \quad \mathbb{Z}_{B_1} \to \mathbb{Z}_{b_{1,0}} \times \ldots \times \mathbb{Z}_{b_{1,h_1-1}} \\ g(a) \quad &= \quad (a_{1,0}, \ldots, a_{1,h_1-1}) \\ &= \quad (a \bmod b_{1,0}, \ldots, a \bmod b_{1,h_1-1}) \end{aligned}$$

and inverse

$$a = \left[\sum_{i=0}^{h_1-1} \xi_{1,i} \frac{B_1}{b_{1,i}}\right]_{B_1} = \sum_{i=0}^{h_1-1} \xi_{1,i} \frac{B_1}{b_{1,i}} - \alpha B_1 \qquad (1)$$

where $\xi_{1,i} = \left[a_{1,i} \frac{b_{1,i}}{B_1}\right]_{b_{1,i}}$.

More concretely, the RNS exploits the CRT to replace additions, subtractions and multiplications over large integers in $\mathbb{Z}_{B_1}$ by the coefficient-wise additions, subtractions and mutltiplications over the smaller channels $\mathbb{Z}_{b_{1,0}}, \ldots, \mathbb{Z}_{b_{1,h_1-1}}$. While these operations are made faster with the RNS, operations such as divisions and modular reductions are harder to implement. One often has to use basis extensions to deal with these operations. This procedure exploits (1) to extend the representation of a number in basis $\mathcal{B}_1 = \{b_{1,0}, \ldots, b_{1,h_1-1}\}$ to another basis $\mathcal{B}_2 = \{b_{2,0}, \ldots, b_{2,h_2-1}\}$. In cases where an error of αB_1 can be tolerated, extensions may be performed with FastBConv to approximate (1) in an efficient way [2]:

$$a_{2,i} = \operatorname{FastBConv}(a, \mathcal{B}_1) = \sum_{i=0}^{h_1-1} \xi_{1,i} \frac{B_1}{b_{1,i}} \bmod b_{2,i}$$

When such an error cannot be tolerated, an extra residue $a_{\mathsf{sk}} = a \bmod b_{\mathsf{sk}}$ may be computed. This enables the computation of α as

$$\alpha = \left[(\operatorname{FastBConv}(a, \mathcal{B}_1) - a_{\mathsf{sk}}) B_1^{-1}\right]_{b_{\mathsf{sk}}}$$

for $|a| < \lambda B_2$, an integer λ and $b_{\mathsf{sk}} \geq 2(h_2 + \lambda)$ [6, Lemma 6]. In this case, the basis extension may be terminated with FastBConvSK:

$$\begin{aligned} a_{2,i} = \operatorname{FastBConvSK}(a, \mathcal{B}_1, \alpha) &= \\ \sum_{i=0}^{h_1-1} \xi_{1,i} \frac{B_1}{b_{1,i}} &- \alpha B_1 \bmod b_{2,i} \qquad (2) \end{aligned}$$

C. Lattices

Given a basis

$$R = \begin{bmatrix} r_{0,0} & r_{0,1} & \cdots & r_{0,n-1} \\ r_{1,0} & r_{1,1} & \cdots & r_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m-1,0} & r_{m-1,1} & \cdots & r_{m-1,n-1} \end{bmatrix}$$

the lattice $\mathcal{L}(R)$ generated by the rows of $R \in \mathbb{R}^{m \times n}$ corresponds to the following discrete subgroup of $\mathbb{R}^n$:

$$\mathcal{L}(R) = \{v | \exists_{z \in \mathbb{Z}^m} zR = v\}$$

Herein, we will only be dealing with full rank integer lattices, corresponding to the case where $m = n$ and $R \in \mathbb{Z}^{n \times n}$. In this case, the lattice $\mathcal{L}(R)$ induces a congruence relation over $\mathbb{Z}^{n \times n}$. Two vectors are said to be congruent when their difference is in $\mathcal{L}(R)$:

$$v = u \bmod \mathcal{L}(R) \Leftrightarrow v - u \in \mathcal{L}(R)$$

Reducing a vector u modulo a basis R corresponds to finding the vector v satisfying $v = u \bmod \mathcal{L}(R)$ and

$$v = zR \text{ with } -1/2 \leq z_i < 1/2 \, \forall_{0 \leq i < n}$$

III. Proposed HyPoRes System

The proposed representation system can be seen as a generalisation of [7], and is described in Definition III.1. A number $a \in \mathbb{Z}_P$ is represented as a polynomial $A(X)$ with coefficients of norm smaller than ρ that when evaluated in γ produces:

$$A(\gamma) = a \bmod P \tag{3}$$

γ satisfies $[\gamma^n]_P = \beta$ for a small integer β. Thus, operating with these polynomials modulo $X^n - \beta$ is isomorphic to operating with the corresponding integers modulo P. In [7], it is proven that for digits satisfying $|a^{(i)}| < \rho$, a $\rho \geq \beta P^{1/n}$ suffices to represent all congruency classes $a \in \mathbb{Z}_P$. Herein, digits are represented with respect to two RNS bases $\mathcal{B}_1$ and $\mathcal{B}_2$ and a modulus b_{sk}. The need for these moduli will become evident when describing the HyPoRes multiplication algorithm.

Definition III.1 (HyPoRes). *An Hybrid-Polynomial Residue Number System (HyPoRes) is a sextuple $\mathcal{H} = (P, n, \rho, \mathcal{B}_1, \mathcal{B}_2, b_{\mathsf{sk}})$. β is defined to be the smallest integer that is not an n^{th} power over $\mathbb{Z}$, but that has an n^{th} root modulo P (see Lemma II.1). We identify this root with $\gamma^n = \beta \bmod P$. Positive integers $0 \leq a < P$ are represented as a polynomial of n coefficients $(a^{(0)}, \ldots, a^{(n-1)})$, wherein each coefficient $a^{(i)}$ is represented with respect to the two RNS bases $\mathcal{B}_1 = \{b_{1,0}, \ldots, b_{1,h_1-1}\}$ and $\mathcal{B}_2 = \{b_{2,0}, \ldots, b_{2,h_2-1}\}$ and the modulus b_{sk}, satisfying:*

$$a = \sum_{i=0}^{n-1} \left[\sum_{j=0}^{h_1-1} \xi_{i,1,j} \frac{B_1}{b_{1,j}} \right]_{B_1} \gamma^i \bmod P$$

with $\xi_{i,1,j} = \left[a_{i,1,j} \frac{b_{1,j}}{B_j} \right]_{b_{1,j}}$, $|a^{(i)}| < \rho$ and $a_{i,k,j} = a^{(i)} \bmod b_{k,j}$. We use $a_{i,k}$ to denote $a^{(i)} \bmod B_k$, capital values A to denote a representation of a under $\mathcal{H}$, A_1 to denote the representation of a under $\mathcal{B}_1$, A_2 the representation of a under $\mathcal{B}_2$, a_{sk} the representation of a modulo b_{sk} and define the norm $||A||_\infty = \max(|a^{(0)}|, \ldots, |a^{(n-1)}|)$.

Algorithm 1 Proposed Modular Multiplication Algorithm

```
Require: $||A||_\infty, ||C||_\infty < k\rho$
Require: $M' = -M^{-1} \bmod \mathcal{B}_1$
Ensure: $||R||_\infty < \rho$ with $r = acB_1^{-1} \bmod P$
    $D = A \star C \bmod \mathcal{B}_1 \cup \mathcal{B}_2 \cup \{b_{\mathsf{sk}}\}$
    $Q_1 = D \star M' \bmod \mathcal{B}_1$
    $Q_2 = \mathrm{FastBConv}(q, \mathcal{B}_1) \bmod \mathcal{B}_2$
    $q_{\mathsf{sk}} = \mathrm{FastBConv}(q, \mathcal{B}_1) \bmod b_{\mathsf{sk}}$
    $R_2 = \frac{D + Q \star M}{B_1} \bmod \mathcal{B}_2$
    $r_{\mathsf{sk}} = \frac{d_{\mathsf{sk}} + q_{\mathsf{sk}} \star M}{B_1} \bmod b_{\mathsf{sk}}$
    $\alpha = \left[ (\mathrm{FastBConv}(r, \mathcal{B}_2) - r_{\mathsf{sk}}) B_2^{-1} \right]_{b_{\mathsf{sk}}}$
    $R_1 = \mathrm{FastBConvSK}(r, \mathcal{B}_2, \alpha) \bmod \mathcal{B}_1$
    $R = (R_1, R_2)$
```

A. Proposed Modular Multiplication Algorithm

The proposed modular multiplication can be found in Algorithm 1. Therein, the operation $A \star C$ denotes the following vector-matrix multiplication:

$$A \star C = AC \bmod (X^n - \beta) = \begin{bmatrix} a^{(0)} & a^{(1)} & \ldots & a^{(n-1)} \end{bmatrix} \begin{bmatrix} c^{(0)} & c^{(1)} & \ldots & c^{(n-1)} \\ \beta c^{(n-1)} & c^{(0)} & \ldots & c^{(n-2)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta c^{(1)} & \beta c^{(2)} & \ldots & c^{(0)} \end{bmatrix} \tag{4}$$

where element-wise multiplications are conducted in RNS.

The vector M used in Algorithm 1 corresponds to a small nonzero representation of zero under $\mathcal{H}$. A representation of M with norm smaller than $P^{1/n}$ is guaranteed to exist, as described in Lemma III.1.

Lemma III.1. *A nonzero representation of zero of norm smaller than $P^{1/n}$ exists under $\mathcal{H}$*

Proof. We start by building the lattice $\mathcal{L}(\Gamma)$ of the representations of zero under $\mathcal{H}$ where

$$\Gamma = \begin{bmatrix} P & 0 & \ldots & 0 \\ -\gamma & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -\gamma^n & 0 & \ldots & 1 \end{bmatrix}$$

Each line in Γ corresponds to either $P = 0 \bmod P$ or $-\gamma^i + X^i$, which when evaluated at $X = \gamma$ produces a value congruent with 0. Minskowski's theorem [8] guarantees that $\mathcal{L}(\Gamma)$ contains a nonzero vector of norm at most $(\det \mathcal{L}(\Gamma))^{1/n} = P^{1/n}$. Thus M can be obtained by finding the nonzero shortest lattice point in $\mathcal{L}(\Gamma)$. While this problem is complex in general, herein we are dealing with lattices of a small dimension, making it solvable in a short time. □

In essence, Algorithm 1 starts by computing $D = A \star C \cong A \times C \bmod P$. Then, to reduce the size of the coefficients, a multiple of a nonzero representation of zero M is added to D, making the result divisible by B_1. Since the scaling factor Q is computed modulo B_1, it is first produced in $\mathcal{B}_1$ and then extended to $\mathcal{B}_2$. Afterwards, $R = \frac{D + Q \star M}{B_1}$ is outputted. The division by B_1 guarantees that the norm of the result is small. However, since this division is not possible in $\mathcal{B}_1$, this value is first produced in $\mathcal{B}_2$ and then extended to $\mathcal{B}_1$.

Algorithm 1 requires that an inverse of M exists in the ring $\mathbb{Z}_{B_1}[X]/(X^n - \beta)$. Lemma III.2 guarantees that this is the case when $\mathcal{B}_1$ is built from prime numbers $b_{1,i}$ that do not divide the resultant of M and $X^n - \beta$ and $M \neq 0 \bmod B_1$.

Lemma III.2. *When $\mathcal{B}_1 = \{b_{1,0}, \ldots, b_{1,h_1-1}\}$ such that all $b_{1,i}$ are primes not dividing the resultant of M and $X^n - \beta$ and $M \neq 0 \bmod B_1$, M is invertible in $\mathbb{Z}_{B_1}[X]/(X^n - \beta)$.*

Proof. Since $X^n - \beta$ is irreducible, M with $\deg(M) < n$ and $X^n - \beta$ are coprime. Hence, there exists $(U, V) \in \mathbb{Z}[X]^2$ s.t.

$$UM + V(X^n - \beta) = r$$

where r is the resultant of M and $X^n - \beta$ and $r \neq 0$ [9]. We find the inverse of M modulo B_1 and $X^n - \beta$ by computing $Ur^{-1} \mod b_{1,i}$ for $0 \leq i \leq h_1 - 1$ and lifting the result to $\mathbb{Z}_{B_1}$ with the CRT. The resultant r must be invertible modulo $b_{1,i}$, i.e. coprime to $b_{1,i}$. Since $b_{1,i}$ is prime, it must not divide r. □

Finally, Theorem III.1 proves that Algorithm 1 produces the correct result. Even though values represented under $\mathcal{H}$ will normally satisfy $||A||_\infty < \rho$, their norm might grow after non-reduced additions. Hence, we assume that the inputs to Algorithm 1 have their norm bounded by $||A||_\infty < k\rho$. One of the main conditions of Theorem III.1 is that $B_1 > \rho/\epsilon$ for

$$0 < \epsilon < \frac{\rho - \beta n h_1 ||M||_\infty}{\beta n k^2 \rho}$$

Since B_1 should be minimised, it is crucial to pick a large enough ρ such that the second term of $\epsilon < 1/(\beta n k^2) - h_1||M||_\infty/(k^2\rho)$ is small, but not so large that the numerator of $B_1 > \rho/\epsilon$ is greatly increased. A possible choice is of $\rho \sim 10000||M||_\infty/k^2$. Moreover, an appropriate choice of λ might in some cases reduce the number of moduli of $\mathcal{B}_2$ by one when compared with the case $\lambda = 1$.

Theorem III.1. *Algorithm 1 is correct (i.e. it outputs* $R = ACB_1^{-1} \bmod P$ *with* $||R||_\infty < \rho$ *for* $||A||_\infty, ||C||_\infty < k\rho$*) when* $\mathcal{B}_1$ *is built from prime numbers* $b_{1,i}$ *not dividing the resultant of* M *and* $X^n - \beta$*,* $M \neq 0 \bmod B_1$*,* $B_1 > \rho/\epsilon$*,* $B_2 > \rho/\lambda$*,* $\rho > \beta n h_1 ||M||_\infty$*,* $b_{\mathsf{sk}} \geq 2(h_2 + \lambda)$*, for an integer* λ*, pairwise coprime* B_1*,* B_2 *and* b_{sk} *and*

$$0 < \epsilon < \frac{\rho - \beta n h_1 ||M||_\infty}{\beta n k^2 \rho}$$

Proof. To prove that Algorithm 1 produces a correct result, we notice that $D + Q \star M = D - (D \star M^{-1} + \alpha B_1) \star M = 0 \bmod B_1$ (for a polynomial error α that results from an inexact basis extension) and hence $D + Q \star M$ is divisible by B_1. Moreover, since M is a representation of zero modulo P, we have that $R \equiv \frac{D + Q \star M}{B_1} = \frac{D}{B_1} = DB_1^{-1} \bmod P$.

We assume that the inputs to Algorithm 1 satisfy $||A||_\infty, ||C||_\infty < k\rho$. Furthermore, we assume that $\rho < \epsilon B_1$. Notice that since the value of Q is extended from $\mathcal{B}_1$ to $\mathcal{B}_2$ in an inexact way, its norm is bounded by $h_1 B_1$. In this case, the norm of R will satisfy

$$\begin{aligned} ||R||_\infty &= \left|\left| \frac{A \star C + Q \star M}{B_1} \right|\right|_\infty \\ &< \frac{\beta n k^2 \rho^2 + \beta n h_1 B_1 ||M||_\infty}{B_1} < \beta n \epsilon k^2 \rho + \beta n h_1 ||M||_\infty \end{aligned} \tag{5}$$

Since we require that $||R|| < \rho$, ϵ should satisfy:

$$0 < \epsilon < \frac{\rho - \beta n h_1 ||M||_\infty}{\beta n k^2 \rho}$$

As a result, Algorithm 1 produces values with the expected norm as long as $B_1 > \rho/\epsilon$ and $\rho > \beta n h_1 ||M||_\infty$. In addition, FastBConvSK produces the correct value when $\rho < \lambda B_2$ and $b_{\mathsf{sk}} \geq 2(h_2 + \lambda)$. □

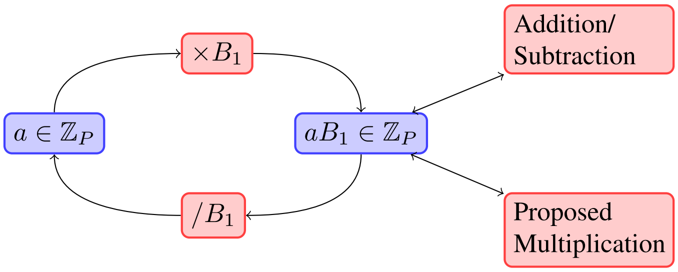

Fig. 1: Values in $\mathbb{Z}_P$ are premultiplied by B_1 before the proposed arithmetic routines are applied

B. Other Operations

Since the modular multiplications in Algorithm 1 are affected by a B_1^{-1} factor, one should multiply values by $B_1 \bmod P$ before representing them in HyPoRes, as represented in Fig. 1. Additions and subtractions are performed without any modular reduction. While this implies that coefficients may grow, we have taken this into account during the design of the modular multiplication algorithm. Therein, one assumes the inputs' coefficients to be bounded by $k\rho$ where ρ is the maximum norm of the coefficients of the output. It is clear that the format aB_1 is maintained after additions since $aB_1 + bB_1 = (a + b)B_1$. This format is also maintained after modular multiplications. In particular, Algorithm 1 computes $(aB_1)(bB_1)B_1^{-1} = abB_1 \bmod P$.

A first approach to convert a value a from a binary representation to the HyPoRes system would start by considering the lattice generated by $M \star X^i$ where M is the nonzero representation of zero referred to in Lemma III.1 and $0 \leq i < n$. This lattice has a basis

$$R = \begin{bmatrix} m^{(0)} & m^{(1)} & \dots & m^{(n-1)} \\ \beta m^{(n-1)} & m^{(0)} & \dots & m^{(n-2)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta m^{(1)} & \beta m^{(2)} & \dots & m^{(0)} \end{bmatrix}$$

with vectors of norm at most $\beta||M||_\infty$. Since $M \star X^i$ are representations of 0 under $\mathcal{H}$, the vector U that results from the reduction of $(a, 0, \dots, 0)$ modulo R represents a under $\mathcal{H}$. Moreover, since the vectors $M \star X^i$ have norm at most $\beta||M||_\infty$, so does U. A standard HyPoRes representation is achieved by reducing the coefficients of U modulo each value in $\mathcal{B}_1$, $\mathcal{B}_2$ and b_{sk}.

A second, more efficient approach, relies on the precomputation of the values $T[i] = [2^{i\lceil \log_2 P \rceil/n} B_1^2]_P$ under $\mathcal{H}$ using the aforementioned conversion. With this second approach, the multiplication by B_1 is integrated in the conversion process and does not need to be performed in a separate step. The value a to be converted is decomposed into n words of $\lceil \log_2 P \rceil / n$ bits each:

$$a = \sum_{i=0}^{n-1} a[i] 2^{i\lceil \log_2 P \rceil/n}$$

Each $a[i]$ can be directly represented in $\mathcal{H}$ by reducing it modulo each value in $\mathcal{B}_1$, $\mathcal{B}_2$ and b_{sk}, and associating it with the first entry of the vector $A[i] \in \mathcal{H}$, which has 0 in all the other entries. Then, $[aB_1]_P$ is computed as

$$A = \sum_{i=0}^{n-1} \text{HyPoRes-mul}(A[i], T[i])$$

IV. Related Art

The optimisation of modular reductions has often focused on primes of particular forms [10], [4]. A first approach [10] induces a quadratic-time multiplication algorithm. A value a is represented as a polynomial A in a ring $\mathbb{Z}[X]/\langle X^b - \beta \rangle$ such that $A(\gamma) = a \bmod P$. The coefficients of the polynomials are represented as a single computer word; and P and γ are chosen such that $\gamma^n - \beta = 0 \bmod P$ and 2^k can be represented as a polynomial M with small coefficients. After computing the product $D = A \star C$, the magnitude of the coefficients of D is recursively reduced by rewriting D as $D = D_L + D_H 2^k$ with $||D_L|| < 2^k$, and updating D with the value of $D_L + D_H \star M$ using only shifts and additions. A second approach [4] induces a subquadratic-time multiplication algorithm. It similarly represents a as a polynomial A in a ring $\mathbb{Z}[X]/\langle X^b - \beta \rangle$ such that $A(\gamma) = a \bmod P$. However, coefficients are represented in RNS with two bases of dynamic range B_1 and B_2; with $\gamma = B_1$ and $P = B_1^n - \beta$. After computing the product $D = d^{(0)} + d^{(1)}X + \ldots + d^{(n-1)}X^{n-1} = A \star C$, the norm of the coefficients of D is reduced through two rounds of carry propagation. Carries are computed approximately with basis extensions. In particular, for $i \in \{0, \ldots, n-1\}$, the carry $e_{i,2}$ is computed as $e_{i,2} = \frac{d_{i,2} - \text{FastBConv}(d, \mathcal{B}_1)}{B_1} \bmod \mathcal{B}_2$, and then $e_{i,1}$ is computed through an exact extension from $\mathcal{B}_2$ to $\mathcal{B}_1$. Afterwards, $d^{(i+1)}$ is updated with $d^{(i+1)} + e^{(i)}$ if $i < n-1$, or $d^{(0)}$ is updated with $d^{(0)} + \beta e^{(n-1)}$ if $i = n-1$. A second round of carry propagation is necessary to achieve small enough coefficients. In this case, the carries are small enough that it suffices to compute $e_{i,2} = \frac{d_{i,2} - \text{FastBConv}(d, \mathcal{B}_1)}{B_1}$ in a single modulus of $\mathcal{B}_2$, and then copy the residue to the remaining moduli in $\mathcal{B}_1$ and $\mathcal{B}_2$.

The two above described modular reduction methods do not extend to primes currently used by ECC standards. First, most standards follow the approach in [11] by choosing a P that is optimised for binary representation systems, i.e. $P = f(2)$ for a very sparse polynomial f. Second, isogeny-based ECC [12] is based on primes built as $P = l_A^{e_A} l_B^{e_B} f \pm 1$, for two small different primes l_A and l_B, two large exponents e_A and e_B and a small cofactor f. Both these cases are incompatible with [10], [4]. In contrast, [7], [3] are suitable for any underlying prime. [7] builds polynomial systems as in [10] but for $\gamma^n = \beta$ and any P. After a product $D = A \star C$, the coefficients of D are reduced by recursively rewriting D as $D = D_L + D_H 2^k$ with $||D_L|| < 2^k$, and updating D with the value of $D_L + D'_H$, where $D'_H = D_H \bmod \langle X^n - \text{mod}, P \rangle$ with $||D'_H|| < \beta P^{1/n}$ precomputed for all possible $D_H 2^k$. The iterative nature of this approach, along with the need to look up precomputed tables make it more appropriate for hardware implementations. Finally, [3] can be seen as a specific case of HyPoRes with $n = 1$, i.e. when polynomial reductions play no role in the algorithm.

While [7], [3] can handle any P, both lead to quadratic-time modular multiplication algorithms. In contrast, it will be seen in Section V that HyPoRes achieves a subquadratic-time complexity for any prime P. Comparisons in Sections V and VI will focus on [4], [3] since [4] also achieves subquadratic-time complexity but for specially crafted primes, and [3] works for any moduli, even when their factorisation is not known, enabling us to evaluate the impact of the assumptions one is allowed to make on the performance of the resulting systems.

V. Computational Complexity

The efficiency of Algorithm 1 can be evaluated in terms of the amount of Single-precision Modular Multiplications (SMMs). We assume that in (4) multiplications by β can either be computed through shifts and additions, since β is a small integer, or, when multiplying by M' and M, β can be integrated on the precomputed M' and M. Hence, (4) requires n^2 multiplications for each moduli it is being operated on. Part of the constants needed for the basis extensions, related for instance with the multiplication of the residues of q by $b_{1,j}/B_1$ in preparation of the extension of q to $\mathcal{B}_2$ can also be integrated in the precomputation of M' and dealt with at no cost. A further nh_2 multiplications may be saved by storing the values $\xi_{i,2,j} = a_{i,2,j} b_{2,j} / B_2 \bmod b_{2,j}$ instead of $a_{i,2,j} \bmod b_{2,j}$ in basis $\mathcal{B}_2$ for any a represented under $\mathcal{H}$. One can conclude that the cost of Algorithm 1 in terms of SMMs is:

$$2n^2(h_1 + h_2 + 1) + 2nh_1h_2 + 2n(h_2 + h_1 + 1)$$

To achieve a fairer comparison, the approach of [4] has been adapted to make use of the basis extensions described in Section II-B. In this case, the amount of SMMs required to compute a multiplication modulo P, including the two rounds of carry propagation described in Section IV, is:

$$n^2(h_1 + h_2 + 1) + 2nh_1h_2 + n(4h_1 + 3h_2) + 3n - 2h_1 - h_2 - 1$$

Under similar assumptions, a classical RNS modular multiplication, as described in [3], with RNS bases $\mathcal{B}_1$ and $\mathcal{B}_2$ with $H_1 \sim h_1 n$ and $H_2 \sim h_2 n$ moduli would require

$$2H_1H_2 + 4H_1 + 3H_2 + 3$$

SMMs.

Asymptotically, if one chooses $n \sim h_1 \sim h_2 \sim \log_2^{1/2} P$, both the proposed scheme and [4] have complexities of $\mathcal{O}(\log_2^{3/2} P)$ SMMs. In contrast, with a pure RNS approach, one has that $H_1 \sim H_2 \sim \log_2 P$, leading to a complexity of $\mathcal{O}(\log_2^2 P)$.

VI. Experimental Results

The proposed method for modular multiplication was described in C++. Also, [4], [3] were implemented for comparison. The pure RNS-based multiplication can be seen as a simplification of the proposed method when $n = 1$, and thus $M = P$ and γ and β play no role. We have

Prime	Parameters
$P_{383} = 2^{383} - 187$[13]	$\gamma = 157516587865170260770044116534390462053813572368725849125618118780149572186398098534089825901086085134348010297436 89$ $n = 2$ $\beta = 3$ $m = -99115188531749068587753731999455148585263155251298263175 1 + 366898661104865555403938150878354627805167553995546070069 1X$ $\mathcal{B}_1$ = [4294967291, 4294967279, 4294967231, 4294967197, 4294967161, 4294967111, 4294967087] $\mathcal{B}_2$ = [4294967295, 4294967293, 4294967287, 4294967281, 4294967273, 4294967269] $b_{\mathsf{sk}} = 2^{32}$
$P_{448} = 2^{448} - 2^{224} - 1$[14]	$\gamma = 617037013236874150081232979704524838882978450634686916049074439753925120009974044466434996308660572821900599056417811283480361754355140$ $n = 3$ $\beta = 2$ $m = -35740007252133200888609789334941745184954843 5 + 4584110981517745795407999650205328543654874 20X - 46904587435178900582720820417746113472149713 1X^2$ $\mathcal{B}_1$ = [4294967197, 4294967161, 4294967029, 4294966981, 4294966927, 4294966813] $\mathcal{B}_2$ = [4294967295, 4294967293, 4294967291, 4294967287] $b_{\mathsf{sk}} = 2^{32}$
$P_{521} = 2^{521} - 1$[13]	$\gamma = 23945242826029513411849172299223580994042798784118784$ $n = 3$ $\beta = 2$ $m = -1 + 119726214130147567059245861496117904970213993920593 92X^2$ $\mathcal{B}_1$ = [4294967197, 4294967161, 4294967029, 4294966981, 4294966927, 4294966813] $\mathcal{B}_2$ = [4294967295, 4294967293, 4294967291, 4294967287, 4294967281] $b_{\mathsf{sk}} = 2^{32}$

TABLE I: The HyPoRes parameters used to evaluate the performance of the proposed method for the primes P_{383}, P_{448} and P_{521}

considered the primes P_{383}, P_{448} and P_{521}, and the HyPoRes parameters described in Table I. Notice that the primes P_{383}, P_{448} and P_{521} have been used to define the elliptic curves M-383, Ed448-Goldilocks and E-521 [13], [14], respectively. Moreover, HPR-crafted primes P_{384}, P'_{448} and P_{512} of 384, 448 and 512 bits, respectively, have been considered for the implementation of [4]. The bases $\mathcal{B}_1$ and $\mathcal{B}_2$ in Table I, as well as those chosen for [4], [3], are composed of integers of the form $b_{i,j} = 2^{32} - c_{i,j}$ for small $c_{i,j}$, enabling for fast reductions [15]. In particular, a number a resulting from a product is rewritten as $a = a_0 + 2^{32}a_1$, and the equality $2^{32} = c \bmod b_{i,j}$ is applied iteratively as $a = a_0 + ca_1$ to reduce the magnitude of a.

Fig. 2 presents the required amount of elementary multiplications for the proposed approach with the parameters described in Table I; for a pure-RNS approach with equivalent parameters ($h_1 = 13$ for P_{383}; $h_1 = 15$ for P_{448}; and $h_1 = 17$ for P_{521}); and for HPR with the primes P_{384} ($n = 2$ and $h_1 = 6$), P'_{448} ($n = 2$ and $h_1 = 7$) and P_{512} ($n = 4$ and $h_1 = 4$). Moreover, the above-described code was compiled with gcc 4.8.5 with the `-Ofast` and `-march=native` flags and executed on a i7-3770K processor with 8GB of main memory operated by CentOS 7.3. No parallelism was exploited. The average modular multiplication time for the HyPoRes, pure-RNS and HPR representations can be seen in Fig. 3.

Figures 2 and 3 suggest that, although a similar performance is attained for both HyPoRes and a pure-RNS approach for the prime P_{383}, the HyPoRes system has a better scalability as the bit-length of the primes increases. This behaviour was predicted in Section V. Moreover, while second-order factors, such as the number of additions, limit the obtained speed-up of HyPoRes when compared with a pure-RNS approach for the smaller primes, when comparing the theoretical predictions of Fig. 2 with the experimental results of Fig. 3, a maximum speed-up of approximately 1.4 is obtained in both cases for P_{521}.

While Figures 2 and 3 show that the performance of HyPoRes is slightly worse than HPR, it relies on weaker assumptions (since it does not require the use of specially crafted primes), making it more flexible and applicable in practice. Fig. 4 emphasises the relation between the assumptions pure-RNS, HyPoRes and HPR rely on and the performance of the resulting system. Since HPR relies on primes of a particular kind, this makes it hardly practical, because the primes for cryptographic applications have already been standardised with a different shape. In contrast, HyPoRes can be used whenever one knows the factoring of the underlying modulus. While the applicability to ECC has been herein demonstrated, HyPoRes can also be applied to ElGamal [16] and Rivest-Shamir-Adleman (RSA) [17] decryption and signing. In addition, while HyPoRes is less widely applicable than a pure-

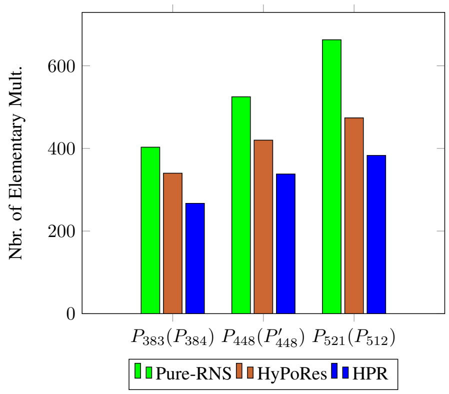

Fig. 2: Number of elementary multiplications required by a pure-RNS and the proposed approaches, as well as with HPR with specially crafted primes. The settings in parenthesis were used for the HPR modular multiplication

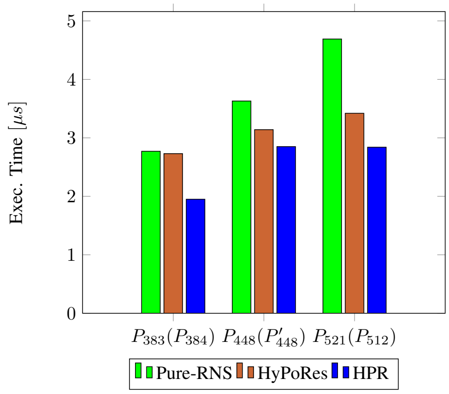

Fig. 3: Average execution time of a pure-RNS and the proposed approaches, as well as of with HPR with specially crafted primes. The settings in parenthesis were used for the HPR modular multiplication

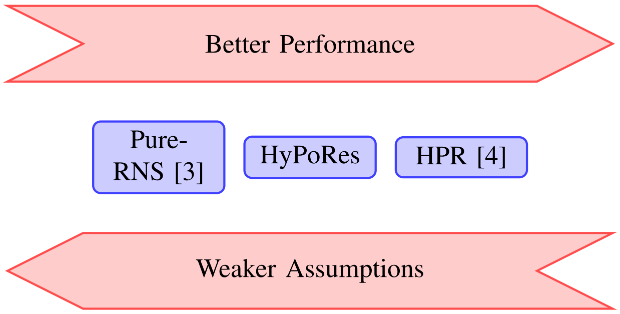

Fig. 4: Qualitative comparison of HyPoRes with related art [3], [4]

RNS approach, it makes the application of small RNS bases of near power-of-two moduli [18], typically employed in signal processing, viable for cryptographic applications.

VII. Beyond Performance: Protection against SCAs

SCAs exploit weaknesses in the implementation of cryptosystems to derive sensitive information from power traces, timing analysis and other physical sources of information. In the context of ECC, Simple Power Analyses (SPAs), wherein one tries to distinguish between EC point-doubling and adding by directly analysing power traces, can be mitigated through the use of formulae in which the two operations are realised with the same basic operations [19]. In contrast, Differential Power Analyses (DPAs) [20] predict power consumption based on an hypothesis for a subset of the bits of the sensitive information and correlate it with actual power measurements to find the most likely hypothesis, requiring a large number of power traces to retrieve the private data. Certain techniques [21] use internal correlations to reduce the necessary number of traces for a successful attack. A single trace may sometimes suffice.

The resistance against this type of attacks may be improved through message blinding, wherein the representation of the values being operated on is randomised, to prevent the prediction of the consumed power. First, resistance against DPAs [20] may be achieved by randomising the representation of values at the beginning of EC point multiplication. Second, if one wants to protect an implementation against [21], values should be randomised during point multiplication.

A. Generalisation of the Reduction Polynomial

Definition III.1 is herein relaxed to allow for multiple representations of the same value, and enable randomisation through an arbitrary choice of one of them. Instead of selecting γ as a root of $X^n - \beta$ modulo P, γ is defined to be a root of $E(X) = e^{(0)} + \ldots + e^{(n-1)} X^{n-1} + X^n$ modulo P, where E is an irreducible polynomial over $\mathbb{Z}[X]$ with small coefficients. In this case, Lemma III.1 is still applicable since it is independent of the underlying E, ensuring the existence of a small nonzero representation of zero M. Furthermore, multiplication of A by X is achieved with the following vector-matrix multiplication:

$$A \star X = \begin{bmatrix} a^{(0)} & a^{(1)} & \ldots & a^{(n-1)} \end{bmatrix} \times \\ \underbrace{\begin{bmatrix} 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 \\ -e^{(0)} & -e^{(1)} & -e^{(2)} & \ldots & -e^{(n-2)} & -e^{(n-1)} \end{bmatrix}}_{D} \quad (6)$$

Thus, (4) may be generalised to

$$A \star C = AC \bmod E = \\ \begin{bmatrix} a^{(0)} & a^{(1)} & \ldots & a^{(n-1)} \end{bmatrix} \begin{bmatrix} C \star X^0 \\ C \star X^1 \\ \vdots \\ C \star X^{n-1} \end{bmatrix} \quad (7)$$

where the multiplication by X^i is achieved by multiplying C by the i-th power of D in (6). Since E has small coefficients, the multiplication by D^i can be implemented with only shifts and additions. Therefore, the complexity of Algorithm 1 is maintained for the new definition of $\star$.

The parameters of an EC may be precomputed under multiple HyPoRes associated with different reducing polynomials E. The prevention of DPA may be achieved by selecting a random E thereof at the beginning of point multiplication. In this case, a conversion from the binary representation system to a generic HyPoRes is required. When preventing attacks like [21], conversions between HyPoRes of different E are required throughout the EC point multiplication. We now consider these two types of conversions. The first deals with the conversion of a binary value to an HyPoRes with generic E. In this case the strategy proposed in Section III-B is still applicable by replacing R with

$$R = \begin{bmatrix} M \star X^0 \\ M \star X^1 \\ \vdots \\ M \star X^{n-1} \end{bmatrix}$$

The second type deals with conversions from an HyPoRes associated with a polynomial E, $\mathcal{H}_E$, to another associated with E', $\mathcal{H}_{E'}$. By noticing that the representation of $[aB_1]_P$ under $\mathcal{H}_E$ satisfies

$$A = \sum_{i=0}^{n-1} A[i]\gamma^i \bmod P,$$

the conversion to $\mathcal{H}_{E'}$ is achieved as

$$A' = A[0] + \sum_{i=1}^{n-1} \text{HyPoRes-mul}_{E'}(A[i], T_{E\rightarrow E'}[i])$$

where $T_{E\rightarrow E'}[i]$ is a representation of $\gamma^i B_1 \bmod P$ under $\mathcal{H}_{E'}$.

VIII. Conclusion

While the HPR has made subquadratic-time multiplication algorithms viable for ECC, the need to use primes of a special form makes it hardly practical. Through a weakening of the underlying assumptions, the HyPoRes has been herein proposed. It not only achieves a similar subquadratic time complexity, but it also supports any prime, making it compatible with standardised elliptic curves. This results in a slow-down of at most 1.4 when HyPoRes is compared with HPR for HPR-crafted primes, but produces an acceleration of up to 1.4 when compared with generic RNS based approaches for primes of ECC standards. The implications of HyPoRes are wide-spreading. While it is less applicable than pure-RNS approaches, it makes the application of small RNS bases of near power-of-two moduli, typically employed in signal processing, viable for cryptographic applications. Also, since it reduces the complexity of basis extensions when compared with a pure-RNS approach, HyPoRes is more amenable to parallelism at a smaller scale. Finally, through a generalisation of the reducing polynomial, redundant representations are introduced, providing for resistance against SCAs.

Acknowledgement

This work was partially supported by national Portuguese funds through Fundação para a Ciência e a Tecnologia (FCT) by the Ph.D. grant with reference SFRH/BD/103791/2014; by the ANR grant ARRAND 15-CE39-0002-01; through the Pessoa/Hubert Curien programme with reference 4335 (FCT)/40832XC (CAMPUSFRANCE); and by EU's Horizon 2020 research and innovation programme under grant agreement No. 779391 (FutureTPM).

References

[1] L. Sousa, S. Antao, and P. Martins. Combining residue arithmetic to design efficient cryptographic circuits and systems. *IEEE Circuits and Systems Magazine*, 16(4):6–32, Fourthquarter 2016.

[2] Jean-Claude Bajard and Laurent Imbert. A full RNS implementation of rsa. *IEEE Trans. Comput.*, 53(6):769–774, June 2004.

[3] Samuel Antao, Jean Claude Bajard, and Leonel Sousa. RNS-based elliptic curve point multiplication for massive parallel architectures. *The Computer Journal*, 55:629–647, 05 2012.

[4] K. Bigou and A. Tisserand. Hybrid position-residues number system. In *IEEE 23nd Symposium on Computer Arithmetic (ARITH)*, pages 126–133, July 2016.

[5] Anna M. Johnston. A generalized qth root algorithm. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, pages 929–930, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[6] Jean-Claude Bajard, Julien Eynard, Anwar Hasan, and Vincent Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. Cryptology ePrint Archive, Report 2016/510, 2016. https://eprint.iacr.org/2016/510.

[7] J.-C. Bajard, L. Imbert, and T. Plantard. Arithmetic operations in the polynomial modular number system. In *17th IEEE Symposium on Computer Arithmetic (ARITH'05)*, pages 206–213, June 2005.

[8] Jiří Matoušek. *Lattices and Minkowski's Theorem*, pages 17–28. Springer New York, New York, NY, 2002.

[9] Joe Buhler. Resultants, discriminants, bezout, nullstellensatz, etc. http://people.reed.edu/~jpb/alg/notes/101.pdf, 2001. Reed College.

[10] Jean-Claude Bajard, Laurent Imbert, and Thomas Plantard. Modular number systems: Beyond the mersenne family. In *Selected Areas in Cryptography, LNCS 3357*, pages 159–169. Springer-Verlag, 2004.

[11] Jerome A. Solinas. Generalized mersenne numbers. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.2133, 10 1999.

[12] Luca De Feo. Mathematics of isogeny based cryptography. *CoRR*, abs/1711.04062, 2017.

[13] Diego F. Aranha, Paulo S. L. M. Barreto, Geovandro C. C. F. Pereira, and Jefferson E. Ricardini. A note on high-security general-purpose elliptic curves. Cryptology ePrint Archive, Report 2013/647, 2013. https://eprint.iacr.org/2013/647.

[14] Mike Hamburg. Ed448-goldilocks, a new elliptic curve. Cryptology ePrint Archive, Report 2015/625, 2015. https://eprint.iacr.org/2015/625.

[15] Richard Crandall and Carl Pomerance. *Prime Numbers: a Computational Perspective*. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2001.

[16] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18. Springer Berlin Heidelberg, 1985.

[17] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[18] A. S. Molahosseini, A. A. E. Zarandi, P. Martins, and L. Sousa. A multifunctional unit for designing efficient RNS-based datapaths. *IEEE Access*, 5:25972–25986, 2017.

[19] Marc Joye and Jean-Jacques Quisquater. Hessian elliptic curves and side-channel attacks. volume 2162, pages 402–410, 01 2001.

[20] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[21] C. D. Walter. Sliding windows succumbs to big mac attack. In *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 286–299. Springer Berlin Heidelberg, 2001.