# Efficient reductions in cyclotomic rings - Application to Ring-LWE based FHE schemes

Jean-Claude Bajard, Julien Eynard, Anwar Hasan, Paulo Martins, Leonel Sousa, Vincent Zucca

# Efficient reductions in cyclotomic rings - Application to Ring-LWE based FHE schemes

Jean-Claude Bajard[1], Julien Eynard[2], Anwar Hasan[2]
Paulo Martins[3], Leonel Sousa[3], Vincent Zucca[1]

[1] Sorbonne Universités, UPMC, CNRS, LIP6, Paris, France
Email: jean-claude.bajard@lip6.fr; vincent.zucca@lip6.fr
[2]Department of Electrical and Computer Engineering, University of Waterloo
Email: jeynard@uwaterloo.ca; ahasan@uwaterloo.ca
[3]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
Email: paulo.sergio@netcabo.pt; las@inesc-id.pt

**Abstract.** With Fully Homomorphic Encryption (FHE), it is possible to process encrypted data without access to the private-key. This has a wide range of applications, most notably the offloading of sensitive data processing. Most research on FHE has focused on the improvement of its efficiency, namely by introducing of schemes based on Ring-Learning With Errors (RLWE), and techniques such as batching, which allows for the encryption of multiple messages in the same ciphertext. Much of the related research has focused on RLWE relying on power-of-two cyclotomic polynomials. While it is possible to achieve efficient arithmetic with such polynomials, one cannot exploit batching. Herein, the efficiency of ring arithmetic underpinned by non-power-of-two cyclomotic polynomials is analyzed and improved. Two methods for polynomial reduction are proposed, one based on the Barrett reduction and the other on a Montgomery representation. Speed-ups of up to 2.66 are obtained for the reduction operation using an i7-5960X processor when compared with a straightforward implementation of the Barrett reduction. Moreover, the proposed methods are exploited to enhance homomorphic multiplication of Fan-Vercauteren (FV) and Brakerski-Gentry-Vaikuntantahan (BGV) encryption schemes, producing experimental speed-ups of up to 1.37.

**Keywords:** Polynomial Reduction, Number Theoretic Transform, Residue Number Systems, Ring-Learning With Errors, Homomorphic Encryption

## 1 Introduction

There is an increasing conflict between the convenience provided by cloud services and privacy concerns. Privacy can be achieved by encrypting data before uploading it to the cloud. However, with traditional cryptosystems, one is not able to process encrypted data [15], nullifying the benefits of cloud computing. A solution to this problem is the application of FHE, which allows for the creation of malleable cryptograms [15]. Homomorphic operations can afterwards be

applied to these cryptograms. Despite its wide range of applicability, FHE has seldom been applied in practice due to its low computational performance.

Most recent research on FHE has focused on improving its efficiency [23]. LWE [27] and its ring-variant RLWE [22] has been suggested in this context as a framework for improving the complexity of FHE. The benefits brought forth by RLWE are twofold. First, operations are executed in a cyclotomic ring, and therefore benefit from its algebraic structure. Second, the plaintext space is isomorphic to the Cartesian product of multiple smaller spaces, designated batching slots, allowing for multiple messages to be encrypted and processed in a single ciphertext in parallel. Most of the related research has focused on cyclotomic rings of the form $\mathbb{Z}[X]/(X^N + 1)$, for $N$ a power of two, due to the simple arithmetic associated with the resulting ring. However, they only allow for a single batching slot. While RLWE modulo other cyclotomic polynomials, providing for a large number of batching slots has been previously considered [13, 28], it remains not used much because of its less efficient arithmetic. In [22], Lyubashevsky et al. proposed algorithms using a multivariate/tensored representation of the elements of cyclotomic rings resulting in a more efficient arithmetic in the general case than those using the classical univariate representation. The implementations of their work [9, 24] are, at the best of our knowledges, the only publicly available implementations optimized for general cyclotomic rings.

In this work we analyse and improve the arithmetic associated with the univariate representation in general cyclotomic rings with efficient reduction algorithms of same asymptotic complexities than those of [22]. Barrett reduction, which is a generic technique to perform polynomial reduction, had been considered in [10] to perform such reductions. However, the algorithms used in [10] do not take into account the characteristics of cyclotomic polynomials, and hence are sub-optimal. Herein, the degree of the polynomials is reduced using cyclotomic properties before applying Barrett's algorithm, decreasing the complexity of the reduction and leading to theoretical speed-ups of up to 2.06. We also show that a Montgomery representation leads to more efficient reductions than generic Barrett algorithms, leading to theoretical speed-ups of up to 2.63.

These gains have been confirmed in practice, with experimental speed-ups of up to 1.95 and 2.55 for the improved Barrett and Montgomery reductions, respectively, when compared with a straightforward Barrett algorithm in an i7-5960X processor. Moreover we have tested the applicability of our algorithms to two of the most currently used homomorphic encryption schemes, BGV [8] developed in HELib [17] and FV [11] developped in SEAL 2.0 [20] resulting in speed-up of up to 1.37 for homomorphic multiplication. These results should reduce the performance differences between the univiate/multivariate representations and therefore bring more flexibility to future implementations.

## 2 Background

Throughout the paper, $\boldsymbol{\phi}_m(X) \in \mathbb{Z}[X]$ will denote the $m$-th cyclotomic polynomial of degree $n = \varphi(m)$, where $\varphi$ is Euler's totient function. The ring $\mathcal{R} =$

$\mathbb{Z}[X]/(\boldsymbol{\phi}_m(X))$ is the main structure of R-LWE-based schemes such as FV and BGV. An element of $\mathcal{R}$ can be thought of as a polynomial with integer coefficients and a degree strictly smaller than $n$. Unless mentioned otherwise, polynomials are represented in the power-basis $\{1, X, \dots X^{n-1}\}$. For $\boldsymbol{a} = \sum_{i=0}^{n-1} a_i X^i \in \mathbb{Z}[X]$, we denote $\|\boldsymbol{a}\|_\infty = \max\{|a_i|, 0 \leqslant i < n\}$. Finally, the underlying space for ciphertexts is $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[X]/\boldsymbol{\phi}_m(X)$, which is composed of elements of $\mathcal{R}$ with coefficients reduced modulo $q$. The notation $|\cdot|_q$ is used to denote the classical residue modulo $q$ in $[0, q)$, while the centered residue in $[-q/2, q/2)$ is denoted by $[\cdot]_q$. Moreover $\lfloor \cdot \rfloor$ denotes flooring while $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

### 2.1 Residue Number System (RNS)

In practice, the value of $q$ underpinning $\mathcal{R}_q$ is chosen to be the product of small different prime numbers $q = q_1 \cdots q_k$. Therefore, thanks to the Chinese Remainder Theorem (CRT), $\mathcal{R}_q$ splits into a cartesian product of smaller rings through the following isomorphism:

$$\mathrm{RNS}_{q=q_1 \dots q_k} : \left| \begin{array}{l} \mathcal{R}_q \to \mathcal{R}_{q_1} \times \ldots \times \mathcal{R}_{q_k} \\ \quad \boldsymbol{a} \mapsto (\boldsymbol{a} \bmod q_1, \ldots, \boldsymbol{a} \bmod q_k) \end{array} \right. \tag{1}$$

The RNS exploits (1) to transfer the arithmetic modulo $q$ of the coefficients to $k$ smaller arithmetics modulo each $q_i$. Thus, better performance is achieved due to smaller arithmetics and parallel computations over each $\mathcal{R}_{q_i}$.

### 2.2 Product of elements in $\mathcal{R}_q$

Since the degree $n$ of the polynomials is large in practice, the polynomial product is a major bottleneck for the efficiency of R-LWE based schemes. However, when $n$ is close or equal to a power of two, it can be performed efficiently thanks to the Number Theoretic Transform (NTT). Let $\mathcal{N}_2$ be the function defined over $\mathbb{N}$ such that for any $n \in \mathbb{N}$ $\mathcal{N}_2(n)$ is the smallest power of two greater than or equal to $n$. In our context the products have a degree strictly smaller than $2n$, therefore we denote $N = \mathcal{N}_2(2n)$. For a prime $q$, a primitive $N^{\text{th}}$-root of unity $\omega \in \mathbb{Z}_q$ exists if and only if $q \equiv 1 \bmod N$. For a $\mathbb{Z}_q$ equipped with $\omega$, the following ring isomorphism is verified:

$$\mathrm{NTT}_{q,N,\omega} : \left| \begin{array}{c} \mathbb{Z}_q[X]/(X^N - 1) \to \mathbb{Z}_q[X]/(X - 1) \times \cdots \times \mathbb{Z}_q[X]/(X - \omega^{N-1}) \\ \boldsymbol{a} \mapsto \widehat{\boldsymbol{a}} = \left(\boldsymbol{a}(1), \boldsymbol{a}(\omega), \ldots, \boldsymbol{a}(\omega^{N-1})\right) \end{array} \right. \tag{2}$$

Once (2) is applied to obtain the NTT representation of polynomials, their product can be performed coordinate-wise. The time-complexity of arithmetic becomes linear in $N$ and the bottleneck changes to the evaluation of (2), as well as of its inverse whose evaluation require $O(N \log(N))$ multiplications in $\mathbb{Z}_q$ by state-of-the-art algorithms ([18], [21]). When the context is clear, we will denote an NTT transformation of degree $N$ by $\mathrm{NTT}_N$ instead of $\mathrm{NTT}_{q,N,\omega}$.

In order to efficiently compute the product of elements $\boldsymbol{a}$ and $\boldsymbol{b}$ of $\mathcal{R}_q$, seen as polynomials over $\mathbb{Z}_q[X]$ of degree at most $n-1$, one has to compute the NTT representation of $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b}$ of degree $2n-2$ through:

$$\mathrm{NTT}_N(\boldsymbol{c}) = \mathrm{NTT}_N(\boldsymbol{a}) \odot \mathrm{NTT}_N(\boldsymbol{b}) \tag{3}$$

where $\odot$ denotes the component-wise multiplication in $\mathbb{Z}_q$. To obtain the value of $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b} \in \mathcal{R}_q$, a second step is needed, which consists of reducing the result of (3) modulo $\boldsymbol{\phi}_m(X)$. Notice that when $m$ is a power of two ($\boldsymbol{\phi}_m(X) = X^{m/2}+1$), one can use a negatively-wrapped convolution for the evaluation of the NTT and its inverse [21]. This allows us to use an NTT of size $\mathcal{N}_2(n)$ instead of $\mathcal{N}_2(2n)$ for the evaluation of (3) and also to recover the polynomial reduced modulo $X^{m/2}+1$ at the end of (3) just with an inverse NTT of size $\mathcal{N}_2(n)$. However, for other values of $m$ this method cannot be applied and a more complex reduction has to be carried out after applying (3).

Barrett's strategy for modular reduction over the integers [5] can be adapted to polynomial modular arithmetic to reduce a polynomial $\boldsymbol{c}$ of degree $n+\alpha$ by $\boldsymbol{\phi}_m$ of degree $n$. The quotient of the Euclidean division $\lfloor \boldsymbol{c}/\boldsymbol{\phi}_m \rfloor$ is computed through multiplications by precomputed constants and shifts. We present this strategy in Algorithm 1. The performance of the algorithm is directly related to the size of the polynomial to be reduced: the algorithm is more efficient when $\alpha$ is small. By denoting $\tilde{n} = \mathcal{N}_2(n)$ and $A = \mathcal{N}_2(2\alpha+1)$ the cost of the algorithm is:

$$C_{\mathrm{NTT}}(N) + 2C_{\mathrm{NTT}}(\tilde{n}) + 2C_{\mathrm{NTT}}(A) + (\tilde{n}+A)\mathtt{MMult}_q$$

where $C_{\mathrm{NTT}}(x)$ denotes the cost for evaluating (2) (or its inverse) of size $x$ and $\mathtt{MMult}_q$ the cost of a modular multiplication modulo $q$. One may also notice that Barrett's reduction used in [10] uses NTT of size $N = \mathcal{N}_2(2n)$ to perform the second product while in fact it only requires NTT of size $\mathcal{N}_2(n)$. For the sake of completeness a proof of the correctness of Alg. 1 is given in Appendix A.1.

---

**Algorithm 1** $\mathtt{NTTBarr}(\boldsymbol{P}, \mathbb{Z}_q, \boldsymbol{\phi}_m)$: NTT based Barrett reduction in $\mathbb{Z}_q[X]$, for a prime $q \equiv 1 \bmod N$

---

**Require:** $\boldsymbol{c}_{NTT} = \mathrm{NTT}_N(\boldsymbol{c}) \in \mathbb{Z}_q^N$ with $\deg(\boldsymbol{c}) = n+\alpha < 2n$ with $q$ prime, $n = \deg(\boldsymbol{\phi}_m)$, $\tilde{n} = \mathcal{N}_2(n)$, $A = \mathcal{N}_2(2\alpha+1)$; precomputed $NTT_{\tilde{n}}(\boldsymbol{\phi}_m)$ and $NTT_A(\lfloor X^{n+\alpha}/\boldsymbol{\phi}_m \rfloor)$.

**Ensure:** $\boldsymbol{c} \bmod (q, \boldsymbol{\phi}_m)$ in power-basis.

1: $\boldsymbol{c} \leftarrow \mathrm{NTT}_N^{-1}(\boldsymbol{c}_{NTT})$
2: $\boldsymbol{f} \leftarrow \lfloor \boldsymbol{c}/X^n \rfloor$
3: $\boldsymbol{r} \leftarrow NTT_A^{-1}(NTT_A(\boldsymbol{f}) \odot NTT_A(\lfloor X^{n+\alpha}/\boldsymbol{\phi}_m \rfloor))$
4: $\boldsymbol{r'} \leftarrow \lfloor \boldsymbol{r}/X^{\alpha} \rfloor$
5: $\boldsymbol{d} \leftarrow NTT_{\tilde{n}}^{-1}(NTT_{\tilde{n}}(\boldsymbol{r'}) \odot NTT_{\tilde{n}}(\boldsymbol{\phi}_m))$
6: $\boldsymbol{c'} \leftarrow \boldsymbol{c} \bmod X^{\tilde{n}} - 1$
7: **return** $\boldsymbol{c'} - \boldsymbol{d}$

---

### 2.3 RNS variant of the FV and BGV Encryption Schemes

Fan and Vercauteren [11] have adapted Brakerski's scale invariant FHE scheme [7] to the RLWE framework. More recently, Bajard et al. have provided a full RNS variant of FV [3]. We briefly recall how this RNS variant works.

We first need to introduce the two functions $\xi_q : \mathcal{R}_q \to \mathcal{R}_{q_1} \times \cdots \times \mathcal{R}_{q_k}$ and $\mathcal{P}_{\mathtt{RNS},q} : \mathcal{R}_q \to \mathcal{R}_q^k$ such that for any $\boldsymbol{a} \in \mathcal{R}_q$, $\xi_q(\boldsymbol{a}) = (|\boldsymbol{a} \cdot q/q_i|_{q_i})_{i\in[1,k]}$ and $\mathcal{P}_{\mathtt{RNS},q}(\boldsymbol{a}) = (|\boldsymbol{a} \cdot q/q_i|_q)_{i\in[1,k]}$. It is straightforward to notice that for any $(\boldsymbol{a},\boldsymbol{b}) \in \mathcal{R}_q^2$, $\left\langle \xi_q(\boldsymbol{a}), \mathcal{P}_{\mathtt{RNS},q}(\boldsymbol{b}) \right\rangle \equiv \boldsymbol{ab} \bmod q$. This scalar product occurs in $\mathcal{R}_q$, and so it is composed of polynomial products.

In the context of the FV scheme, the secret-key $\boldsymbol{s} \in \mathcal{R}$ is defined as a "small" polynomial drawn from a distribution $\chi_{key}$. An encryption of $\boldsymbol{m} \in \mathcal{R}_t$ corresponds to a pair of polynomials $\mathtt{ct} = (\boldsymbol{c_0}, \boldsymbol{c_1}) \in \mathcal{R}^2$ satisfying:

$$[\boldsymbol{c_0} + \boldsymbol{c_1 s}]_q = [\lfloor q/t \rfloor \cdot [\boldsymbol{m}]_t + \boldsymbol{v}]_q \tag{4}$$

where $\boldsymbol{v}$ is a noise term that is originally introduced during encryption (which is related to a distribution $\chi_{err}$) and that grows as homomorphic operations are applied. Decryption works correctly so long as this noise is below a certain bound, which limits the amount of operations one can perform.

The homomorphic addition of two ciphertexts corresponds to the pairwise addition of the ciphertexts' polynomials. Regarding homomorphic multiplication, it is useful to see ciphertexts as polynomials of degree 1 with coefficients in $\mathcal{R}$. In this context, homomorphic multiplication takes place in two steps. First, $\mathtt{ct}'_{mult} \leftarrow \left( \left[ \left\lfloor \frac{t}{q} \boldsymbol{c_0^1 c_0^2} \right\rceil \right]_q, \left[ \left\lfloor \frac{t}{q} (\boldsymbol{c_0^1 c_1^2} + \boldsymbol{c_1^1 c_0^2}) \right\rceil \right]_q, \left[ \left\lfloor \frac{t}{q} \boldsymbol{c_1^1 c_1^2} \right\rceil \right]_q \right)$ is computed with a Karatsuba like algorithm. During this procedure, the RNS representations of the input polynomials are extended to bases with larger dynamic ranges so as to compute the products over $\mathcal{R}$ instead of over $\mathcal{R}_q$. Moreover, the division operation is achievable using [3, Section 4.4]. Finally, one has to convert the three-element ciphertext back to a two-element ciphertext, through a process called *relinearization*. This is done by multiplying $\xi_q(\mathtt{ct}'_{mult})$ by pseudo-encryptions of $\mathcal{P}_{\mathtt{RNS},q}(\boldsymbol{s}^2)$ (designated $\overrightarrow{\mathtt{rlk}}$), and adding the result to the other two elements:

$$\mathtt{ct}_{relin} \leftarrow \left( \left[ \boldsymbol{c_0} + \left\langle \xi_q(\boldsymbol{c_2}), \overrightarrow{\mathtt{rlk}_0} \right\rangle \right]_q, \left[ \boldsymbol{c_1} + \langle \xi_q(\boldsymbol{c_2}), \overrightarrow{\mathtt{rlk}_1} \rangle \right]_q \right) \in \mathcal{R}_q^2 \ . \tag{5}$$

The scheme introduced by Brakerski, Gentry and Vainkuntanathan [8] shares many features of FV, and can be similarly adapted to the techniques in [3]. A secret-key is also defined to be a "small" polynomial $\boldsymbol{s} \in \mathcal{R}_q$, and ciphertexts correspond to pairs $(\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}^2$, but messages are encrypted in the Least Significant Bits (LSBs) of (6):

$$[\boldsymbol{c_0} + \boldsymbol{c_1 s}]_q = [[\boldsymbol{m}]_t + t\boldsymbol{v}]_q \tag{6}$$

The change in the positioning of the message bits leads to simpler homomorphic multiplications. First, we compute the degree 2 ciphertext $\mathtt{ct}'_{mult} \leftarrow$

$\left(\left[\boldsymbol{c}_0^1\boldsymbol{c}_0^2\right]_q, \left[(\boldsymbol{c}_0^1\boldsymbol{c}_1^2 + \boldsymbol{c}_1^1\boldsymbol{c}_0^2)\right]_q, \left[\boldsymbol{c}_1^1\boldsymbol{c}_1^2\right]_q\right)$. Since operations are performed modulo $q$, no RNS base extension is required. Afterwards, an operation similar to (5) is applied, so as to convert the three-element ciphertext to a classical two-element ciphertext. Finally, a noise management technique is applied to reduce the growth rate of the norm of $\boldsymbol{v}$ in (6). This technique consists of scaling the ciphertext to a smaller ring $R_{q'}$ with an appropriate rounding, and is performed in two steps:

$$\begin{aligned}&\boldsymbol{\delta}_i \leftarrow t \cdot [-\mathtt{ct}/t]_{q/q'} \text{ for } i = 0, 1\\ &\mathtt{ct} \leftarrow ([q'/q \cdot (\tilde{\boldsymbol{c}}_0 + \boldsymbol{\delta}_0)]_{q'}, [q'/q \cdot (\tilde{\boldsymbol{c}}_1 + \boldsymbol{\delta}_1)]_{q'})\end{aligned}$$

In certain steps of the aforementioned schemes, one needs to add the result of multiple polynomial products. In this case, it is possible to combine the modular reduction of all products involved in the same sum, reducing the overall computational complexity. First, (3) is computed for all products. Then, the NTT representations of the products are added. Finally, a single polynomial reduction method can be applied to the result.

### 2.4 Batching

A common way to improve the efficiency of RLWE schemes is to encrypt several plaintexts in a single ciphertext, through a technique called batching [28]. Under some conditions, $\boldsymbol{\phi}_m$ splits modulo $t$ into $\ell$ distinct irreducible polynomials $f_1, \ldots, f_\ell$ of degree $n/\ell$. This leads to the following ring isomorphism of the plaintext space $R_t$: $\mathcal{R}_t \cong \mathbb{Z}_t[X]/(f_1) \times \cdots \times \mathbb{Z}_t[X]/(f_\ell)$. In this manner, $\ell$ plaintexts $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_\ell$ can be compactly represented as a single polynomial $\boldsymbol{m} \in \mathcal{R}_t$. Afterwards $\boldsymbol{m}$ is encrypted and homomorphic operations applied to this ciphertext operate on each slot individually. This technique is mostly used when evaluating Boolean circuits, i.e. with $t = 2$, to pack $\ell$-bits in a single ciphertext. However, since $X^{m/2} + 1 \equiv (X + 1)^{m/2} \bmod 2$, this technique cannot be used when $m$ is a power of two. Thus the efficient arithmetic associated with power-of-two cyclotomic polynomials has limited applicability.

## 3 Improving polynomial reduction modulo $\boldsymbol{\phi}_m$

In this section, we propose two efficient methods to compute polynomial reductions. The first method takes advantage of the properties of the cyclotomic polynomials to improve the efficiency of the Barrett algorithm. The second reduction rests on an adaptation of the Montgomery modular reduction [25] and can be adapted for other polynomials besides cyclotomics.

### 3.1 Improving Barrett's reduction for cyclotomic polynomials

As explained in section 2, the performance of Barrett's algorithm is sensitive to the difference between the degree of the polynomial to be reduced and that of the polynomial we want to reduce by. The smaller the difference, the more

efficient the algorithm will be. Hence, herein we propose an efficient method to reduce this difference.

Polynomials to be reduced modulo $\boldsymbol{\phi}_m$ have a degree of at most $2n-2$. Let $\boldsymbol{c}$ be such a polynomial. If $\boldsymbol{c}$ were reduced by a polynomial $\boldsymbol{Q}_{sp}$ of degree $n+\alpha+1$, the difference between the degree of the polynomial and the degree of $\boldsymbol{\phi}_m$ would be reduced to $\alpha$. However, in order to obtain the correct value of $\boldsymbol{c} \bmod \boldsymbol{\phi}_m$ in the end, $\boldsymbol{\phi}_m$ has to divide $\boldsymbol{Q}_{sp}$. For this reduction to be efficient, $\boldsymbol{Q}_{sp}$ should be sparse enough so that its reduction can be efficiently handled through few operations in $\mathbb{Z}_q$. Thanks to the cyclotomic property $\prod_{d/m} \boldsymbol{\phi}_d(X) = X^m - 1$, $\boldsymbol{Q}_{sp}$ can be taken as the product of $\boldsymbol{\phi}_m$ and some $\boldsymbol{\phi}_d$ for $d$ dividing $m$. Good candidates can be found by recursively using the fact that if $p$ is a prime not dividing $m'$ then $\boldsymbol{\phi}_{m' \cdot p}(X) \cdot \boldsymbol{\phi}_{m'}(X) = \boldsymbol{\phi}_{m'}(X^p)$. If the polynomial $\boldsymbol{Q}_{sp}$ is found in this way with less than 2 recursions it has coefficients in $\{-1, 0, 1\}$ since it will correspond to a cyclotomic with at most two distinct odd prime factors. In this case, the reduction modulo $\boldsymbol{Q}_{sp}$ only requires shifts and additions in $\mathbb{Z}_q$ and can be done very efficiently.

In addition, when $m < 2n-2$, $\boldsymbol{c}$ can initially be reduced by $X^m - 1$ with $2n - m - 1$ additions in $\mathbb{Z}_q$. Since $\boldsymbol{\phi}_m(X) | \boldsymbol{Q}_{sp}(X) | X^m - 1$ the strategy remains correct, and the complexity of the reduction by $\boldsymbol{Q}_{sp}(X)$ is further reduced. Let $\mathtt{HW}(\boldsymbol{Q}_{\mathrm{sp}})$ be the Hamming weight of $\boldsymbol{Q}_{sp}$. The cost of the reduction of $\boldsymbol{c}$ by $\boldsymbol{Q}_{sp}$ is $(\mathtt{HW}(\boldsymbol{Q}_{sp}) - 1)(m - \deg(\boldsymbol{Q}_{sp}))$ additions in $\mathbb{Z}_q$. At this point, we obtain $\boldsymbol{c'} = \boldsymbol{c} \bmod \boldsymbol{Q}_{sp}$ (with $\deg(\boldsymbol{c'}) \leq n + \alpha$) and $\boldsymbol{c'} \equiv \boldsymbol{c} \bmod \boldsymbol{\phi}_m$.

The final algorithm is depicted in Algorithm 2. It starts by recovering $\boldsymbol{c}$ in power basis from the NTT representation outputted by (3). Then it consecutively reduces $\boldsymbol{c}$ of degree $2n-2$ by $X^m-1$ and the sparse polynomial $\boldsymbol{Q}_{sp}$. This allows to recover $\boldsymbol{c'} = \boldsymbol{c} \bmod \boldsymbol{Q}_{sp}$ of degree $n+\alpha$ very efficiently. Afterwards, a procedure identical to the one described in steps 2 to 7 in Alg. 1 is applied to $\boldsymbol{c'}$ to get $\boldsymbol{c''} = \boldsymbol{c} \bmod \boldsymbol{\phi}_m$.

**Algorithm 2** $\mathtt{modBt}_{\boldsymbol{\phi}_m}$: NTT-based Barrett reduction in $\mathbb{Z}_q[X]$, for $q = q_1 \dots q_k$, with prime integers $q_i$, $q_i \equiv 1 \bmod N$, $N = \mathcal{N}_2(2n)$, $\tilde{n} = \mathcal{N}_2(n)$ and $A = \mathcal{N}_2(2\alpha+1)$.

**Require:** $\boldsymbol{c}_{\mathrm{NTT}} = \mathrm{NTT}_N(\boldsymbol{c})$ with $\deg(\boldsymbol{c}) \leqslant 2n-2$
**Ensure:** $\boldsymbol{c''} = \boldsymbol{c} \bmod \boldsymbol{\phi}_m$ in power-basis.

1: $\boldsymbol{c} \leftarrow \mathrm{NTT}_N^{-1}(\boldsymbol{c}_{\mathrm{NTT}})$
2: **if** $m < 2n-2$ **then**
3: $\quad \boldsymbol{c} \leftarrow \boldsymbol{c} \bmod X^m - 1$
4: $\boldsymbol{c'} \leftarrow \boldsymbol{c} \bmod \boldsymbol{Q}_{sp}$ ▷ Reduction by $\boldsymbol{Q}_{sp}$ of degree $n+\alpha+1$
5: $\boldsymbol{f} \leftarrow \lfloor \boldsymbol{c} / X^n \rfloor$
6: $\boldsymbol{r} \leftarrow NTT_A^{-1}(NTT_A(\boldsymbol{f}) \odot NTT_A(\lfloor X^{n+\alpha} / \boldsymbol{\phi}_m \rfloor))$
7: $\boldsymbol{r'} \leftarrow \lfloor \boldsymbol{r} / X^{\alpha} \rfloor$
8: $\boldsymbol{d} \leftarrow NTT_{\tilde{n}}^{-1}(NTT_{\tilde{n}}(\boldsymbol{r'}) \odot NTT_{\tilde{n}}(\boldsymbol{\phi}_m))$
9: $\boldsymbol{c'} \leftarrow \boldsymbol{c} \bmod X^{\tilde{n}} - 1$
10: **return** $\boldsymbol{c'} - \boldsymbol{d}$

The impact of this sparse reduction is illustrated in Table 1, where polynomials $\boldsymbol{Q}_{sp}$ are presented for different cyclotomic polynomials. Cyclotomics have been chosen with a degree $n = \varphi(m)$ close or equal to a power of two. The number of batching slots $\ell$ associated with each cyclotomic is also presented. The degree of $\boldsymbol{Q}_{sp}$ is $n + \alpha + 1$ thus NTTs of size $A = \mathcal{N}_2(2\alpha + 1)$ are required to compute the first polynomial product in Alg. 2. This is in contrast with $N = \mathcal{N}_2(2n)$ which would have been the size required for the Barrett algorithm without using the sparse reduction. In order to highlight the sparsity of $\boldsymbol{Q}_{sp}$ we give $\mathtt{HW}(\boldsymbol{Q}_{sp})$ which is the number of non-zero coefficients of $\boldsymbol{Q}_{sp}$.

**Complexity** Since the complexity of computing multiplications in $\mathbb{Z}_q$ is much higher than additions, the cost of the reduction by the sparse polynomial can be neglected. Moreover, with the RNS, each multiplication in $\mathcal{R}_q$, with $q = q_1 \dots q_k$ can be decomposed into $k$ independent and smaller multiplications. Therefore the cost to reduce the polynomial $\boldsymbol{c}$ outputted by (3) is essentially $k$ times the cost of Alg. 1:

$$k(N \log_2(N) + 2A \log_2(A) + 2\tilde{n} \log_2(\tilde{n}) + A + \tilde{n})$$

While the cost of the method by using directly Barrett's algorithm, i.e. without performing the reduction by the sparse polynomial, is:

$$k(3N \log_2(N) + 2\tilde{n} \log_2(\tilde{n}) + N + \tilde{n})$$

Based on this analysis we also provide in Table 1 the theoretical speed-up obtained with the use of the sparse reduction.

| $m$ | $n$ | $\ell$ | $\boldsymbol{Q}_{\text{sp}}$ | $\deg(\boldsymbol{Q}_{\text{sp}})$ | $\alpha$ | $\mathtt{HW}(\boldsymbol{Q}_{\text{sp}})$ | $A$ | $N$ | Speed-up |
|---|---|---|---|---|---|---|---|---|---|
| 3855 | 2048 | 128 | $\boldsymbol{\phi}_{3\cdot5}(X^{257})$ | 2056 | 7 | 7 | $2^4$ | $2^{12}$ | 2.06 |
| 4369 | 4096 | 256 | $\boldsymbol{\phi}_{17}(X^{257})$ | 4112 | 15 | 17 | $2^5$ | $2^{13}$ | 2.05 |
| 13107 | 8192 | 512 | $\boldsymbol{\phi}_{3}(X^{17\cdot257})$ | 8738 | 545 | 3 | $2^{11}$ | $2^{14}$ | 1.86 |
| 21845 | 16384 | 1024 | $\boldsymbol{\phi}_{5}(X^{17\cdot257})$ | 17476 | 1091 | 5 | $2^{12}$ | $2^{15}$ | 1.86 |
| 32767 | 27000 | 1800 | $\boldsymbol{\phi}_{7}(X^{31\cdot151})$ | 28086 | 1085 | 7 | $2^{12}$ | $2^{16}$ | 1.95 |
| 65535 | 32768 | 2048 | $\boldsymbol{\phi}_{3\cdot5}(X^{17\cdot257})$ | 34952 | 2183 | 7 | $2^{13}$ | $2^{16}$ | 1.85 |

**Table 1.** Sparse polynomials used for partial reduction with their related parameters.

### 3.2 NTT-based Montgomery's reduction

We propose a Montgomery like reduction of a polynomial given in NTT representation inspired by [4]. The bottleneck of our optimized Barrett algorithm is the computation of the inverse NTT of size $N$ of (3). Our Montgomery reduction takes advantage of the presence of the NTT basis of size $N/2$ (seen as an RNS

basis in [4]) in the basis of size $N$ allowing then to perform all the computations, in particular the inverse NTT evaluation, in the basis of size $N/2$ instead of $N$.

The NTT representation of a polynomial of size $N$ was defined in (2) as the tuple of values $\{\boldsymbol{c} \bmod X - \omega^j | 0 \leq j < N\}$. This representation can be seen as a polynomial-RNS representation of $\boldsymbol{c} \bmod X^N - 1$ since $X^N - 1 = \prod_{0 \leq j < N}(X - \omega^j) \bmod q$, with respect to the NTT-basis:

$$\mathcal{B}_{\omega,N} = \{|X - 1|_q, |X - \omega|_q, \ldots, |X - \omega^{N-1}|_q\}$$

As $X^N - 1$ splits in $(X^{N/2} - 1)(X^{N/2} + 1)$ when $N$ is even, half of the $\text{NTT}_N$ representation of $\boldsymbol{c}$ corresponds to its $\text{NTT}_{N/2}$ representation. Hence, the basis $\mathcal{B}_{\omega,N}$ is split along even and odd powers of $\omega$. We can then define two sub-bases defining two polynomials:

$$\begin{cases} \mathcal{B}^{(e)}_{\omega,N} = \{[X - \omega^{2j}|_q, 1 \leq j \leq N/2\} & ; \boldsymbol{\Psi}^{(e)} = |\prod_{j=1}^{N/2}(X - \omega^{2j})|_q \\ \mathcal{B}^{(o)}_{\omega,N} = \{|X - \omega^{2j+1}|_q, 0 \leq j \leq N/2 - 1\} & ; \boldsymbol{\Psi}^{(o)} = |\prod_{j=0}^{N/2-1}(X - \omega^{2j+1})|_q \end{cases} \tag{7}$$

It is straightforward to notice that $\boldsymbol{\Psi}^{(e)} \equiv X^{N/2} - 1 \bmod q$ and $\boldsymbol{\Psi}^{(o)} \equiv X^{N/2} + 1 \bmod q$. We also note that since $N$ is a power of two, one has $\boldsymbol{\Psi}^{(o)} \equiv \boldsymbol{\phi}_N \bmod q$. Thanks to Lemma 1, whose first point is a direct consequence of Lemma 2 in [12], we can choose $X^{N/2} + 1$ as the Montgomery factor.

**Lemma 1.** *Let $\boldsymbol{\phi}_m$ be the $m$-th cyclotomic polynomial of degree $n$ and $N$ be the smallest power of two greater than or equal to $2n$. If $m$ is not a power of two then:*

- *there exists $(\boldsymbol{U}, \boldsymbol{V}) \in \mathbb{Z}[X]^2$ such that $\boldsymbol{U}(X)\boldsymbol{\phi}_m(X) + \boldsymbol{V}(X)\boldsymbol{\phi}_N(X) = 1$.*
- *for any prime $p$, $\boldsymbol{\phi}_m$ is coprime with $X^N - 1$ in $\mathbb{Z}_p$. In particular $\boldsymbol{\phi}_m$ is a unit in $\mathbb{Z}_p[X]/(\boldsymbol{\phi}_N)$.*

One can extract from the coordinates of $\boldsymbol{c}$ in $\mathcal{B}_{\omega,N}$ the representation $\widehat{\boldsymbol{c}}^{(e)}$ of $\boldsymbol{c}$ in $\mathcal{B}^{(e)}_{\omega,N}$ (resp. $\widehat{\boldsymbol{c}}^{(o)}$ in $\mathcal{B}^{(o)}_{\omega,N}$). So, given $\widehat{\boldsymbol{c}}^{(o)}$ and $\widehat{\boldsymbol{c}}^{(e)}$, we can use the NTT operator to get:

$$\text{NTT}^{-1}_{N/2}(\widehat{\boldsymbol{c}}^{(e)}) = \boldsymbol{c} \bmod (q, X^{N/2} - 1)$$

**Definition 1.** *We define the following function, which takes in as input the residues of the polynomial $\boldsymbol{c}$ ($\deg(\boldsymbol{c}) < N$) modulo a prime $p$:*

$$\mathit{modMong}_{\boldsymbol{\phi}_m,\boldsymbol{\Psi}^{(o)},p}(\boldsymbol{c}) = \frac{\boldsymbol{c} + \boldsymbol{\phi}_m \times |-\boldsymbol{c}/\boldsymbol{\phi}_m|_{\boldsymbol{\Psi}^{(o)}}}{\boldsymbol{\Psi}^{(o)}} \bmod p. \tag{8}$$

The $\texttt{modMong}_{\boldsymbol{\phi}_m,\boldsymbol{\Psi}^{(o)},p}$ function defined in (8) is a classical Montgomery reduction with factor $\boldsymbol{\Psi}^{(o)}$ consisting in an exact polynomial division. It always outputs a polynomial congruent to $|\boldsymbol{c}/\boldsymbol{\Psi}^{(o)}|_{\boldsymbol{\phi}_m}$ but when $\deg(c) \leqslant N/2 + n - 1$ the output is exactly $|\boldsymbol{c}/\boldsymbol{\Psi}^{(o)}|_{\boldsymbol{\phi}_m}$.

**Lemma 2.** *If $\deg(\boldsymbol{c}) \leqslant N/2+n-1$, then $\mathit{modMg}_{\boldsymbol{\phi}_m,\boldsymbol{\Psi}^{(o)},p}(\widehat{\boldsymbol{c}}) = \boldsymbol{c}/\boldsymbol{\Psi}^{(o)} \bmod (p, \boldsymbol{\phi}_m)$.*

First the degree of the numerator in (8) is bounded $\max(\deg(\boldsymbol{c}), \deg(\boldsymbol{\phi}_m) + \deg(\boldsymbol{\Psi}^{(o)}) - 1) \leqslant n + N/2 - 1$. Thus, the degree of the resulting quotient is bounded by $n - 1 < N/2$. Therefore, the output is $|\boldsymbol{c}/\boldsymbol{\Psi}^{(o)}|_{\boldsymbol{\phi}_m}$ and the computation of (8) can be made modulo $X^{N/2} - 1$, i.e. in an NTT representation of size $N/2$ when using primes $q_i \equiv 1 \bmod N$.

Algorithm 3 details the computation of (8). The following precomputations are used therein:

$$\begin{cases} \widehat{\boldsymbol{W}}^{(o)} : (-1/\boldsymbol{\phi}_m) \bmod (q, \boldsymbol{\Psi}^{(o)}) \text{ in base } \mathcal{B}^{(o)}_{\omega,N} \\ \widehat{\boldsymbol{Y}}^{(e)} : (1/\boldsymbol{\Psi}^{(o)}) \bmod (q, \boldsymbol{\Psi}^{(e)}) \text{ in base } \mathcal{B}^{(e)}_{\omega,N} \\ \widehat{\boldsymbol{Z}}^{(e)} : (\boldsymbol{\phi}_m/\boldsymbol{\Psi}^{(o)}) \bmod (q, \boldsymbol{\Psi}^{(e)}) \text{ in base } \mathcal{B}^{(e)}_{\omega,N} \end{cases}$$

**Algorithm 3** $\texttt{modMg}_{\boldsymbol{\phi}_m, \boldsymbol{\Psi}^{(o)}}$: NTT-based Montgomery reduction in $\mathbb{Z}_q[X]$, for $q = q_1 \dots q_k$, with prime integers $q_i$, and $q_i \equiv 1 \bmod \mathcal{N}_2(2n)$

**Require:** $\widehat{\boldsymbol{c}} = \texttt{NTT}_{q,N}(\boldsymbol{c})$ (ie $\boldsymbol{c}$ in base $\mathcal{B}^{(o)}_{\omega,N} \cup \mathcal{B}^{(e)}_{\omega,N}$), with $N = \mathcal{N}_2(2n)$ and $\deg(\boldsymbol{c}) \leqslant 2n - 2 < N/2 + n - 1$.

**Ensure:** $\boldsymbol{R} = (\boldsymbol{c}/\boldsymbol{\Psi}^{(o)}) \bmod (q, \boldsymbol{\phi}_m)$ in power-basis.

1: $(\widehat{\boldsymbol{c}}^{(e)}, \widehat{\boldsymbol{c}}^{(o)}) \leftarrow \texttt{Split}(\widehat{\boldsymbol{c}})$ ▷ Split the NTT coeff. wrt parity of indexes
2: $\widehat{\boldsymbol{Q}}^{(o)} \leftarrow \widehat{\boldsymbol{c}}^{(o)} \odot \widehat{\boldsymbol{W}}^{(o)}$
3: $\widehat{\boldsymbol{Q}}^{(e)} \leftarrow \texttt{BaseConv}(\widehat{\boldsymbol{Q}}^{(o)})$ ▷ base conversion from $\mathcal{B}^{(o)}_{\omega,N}$ to $\mathcal{B}^{(e)}_{\omega,N}$
4: $\widehat{\boldsymbol{T}}^{(e)} \leftarrow \widehat{\boldsymbol{c}}^{(e)} \odot \widehat{\boldsymbol{Y}}^{(e)} + \widehat{\boldsymbol{Q}}^{(e)} \odot \widehat{\boldsymbol{Z}}^{(e)}$
5: $\boldsymbol{R} \leftarrow \texttt{NTT}^{-1}_{N/2}(\widehat{\boldsymbol{T}}^{(e)})$
6: **return** $\boldsymbol{R}$

In line 4 of Alg. 3, we require an operator which takes in as input a vector of coefficients in base $\mathcal{B}^{(o)}_{\omega,N}$. This vector defines a unique polynomial $\boldsymbol{c}$ with $\deg(\boldsymbol{c}) < N/2$. Then the operator must output the vector of coefficients of $\boldsymbol{c}$ in base $\mathcal{B}^{(e)}_{\omega,N}$. More precisely, the function `BaseConv` works as follows, for any $\boldsymbol{c}$ with $\deg(\boldsymbol{c}) < N/2$:

$$\texttt{BaseConv} : (\boldsymbol{c}(\omega), \boldsymbol{c}(\omega^3), \dots, \boldsymbol{c}(\omega^{N-1})) \mapsto (\boldsymbol{c}(1), \boldsymbol{c}(\omega^2), \dots, \boldsymbol{c}(\omega^{N-2})) \pmod{q}. \tag{9}$$

In [4], (9) is computed with a classical Lagrange interpolation. Our context is more specific, because the points in which polynomials are evaluated are powers of a $N^{\text{th}}$ root of unity $\omega$. With this purpose, Alg. 4 implements such base conversion by only using NTTs of degree $N/2$, with $\omega^2$ as a primitive $N/2^{\text{th}}$ root of unity. The proof of correctness of Alg. 4 is given in Appendix A.3.

**Complexity** The total cost of Alg. 3 in terms of modular multiplications is:

$$k\,(3\mathcal{N}_2(n)\log_2(\mathcal{N}_2(n)) + 4\mathcal{N}_2(n))$$

One can find in Table 2 the predicted speed-up of the proposed Montgomery reduction. Despite its better complexity, the Montgomery algorithm suffers from

**Algorithm 4** `BaseConv`

**Require:** $(\boldsymbol{c}(\omega), \boldsymbol{c}(\omega^3), \ldots, \boldsymbol{c}(\omega^{N-1}))$ mod $q_i$, for $\deg(\boldsymbol{c}) < N/2$, $N$ a power of 2 and $\omega$ a primitive $N^{\text{th}}$ root of unity in $\mathbb{Z}_q$.
**Ensure:** $(\boldsymbol{c}(1), \boldsymbol{c}(\omega^2), \ldots, \boldsymbol{c}(\omega^{N-2}))$ mod $q$.
1: $(c'_0, c'_1, \ldots, c'_{N/2-1}) \leftarrow \mathtt{NTT}^{-1}_{N/2,\omega^2}(\boldsymbol{c}(\omega), \boldsymbol{c}(\omega^3), \ldots, \boldsymbol{c}(\omega^{N-1}))$
2: $(p_0, p_1, \ldots, p_{N/2-1}) \leftarrow (c'_0, c'_1, \ldots, c'_{N/2-1}) \odot (1, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-(N/2-1)})$
3: $(r_0, r_1, \ldots, r_{N/2-1}) \leftarrow \mathtt{NTT}_{N/2,\omega^2}(p_0, p_1, \ldots, p_{N/2-1})$
4: **return** $(r_0, r_1, \ldots, r_{N/2-1})$

one main drawback which is the presence of the Montgomery factor in the output. In the following section, modifications to the BGV and FV cryptosystems are proposed to handle this factor.

| $m$ | 3855 | 4369 | 13107 | 21845 | 32767 | 65535 |
|---|---|---|---|---|---|---|
| $n$ | 2048 | 4096 | 8192 | 16384 | 27000 | 32768 |
| Speed-up | 2.62 | 2.62 | 2.63 | 2.63 | 2.63 | 2.63 |

**Table 2.** Theoretical speed-up of Alg. 3 when compared with Alg. 1

## 4 Adaptation of FV and BGV to the Montgomery representation

In this section we show how the Montgomery representation impacts the BGV and FV schemes, and propose modifications to handle these changes. For simplicity, we denote by $\boldsymbol{M}$ the Montgomery factor $X^{N/2} + 1 \bmod \boldsymbol{\phi}_m$. Thanks to Lemma 1 we also know that $\boldsymbol{M}^{-1}$ exists in $\mathcal{R}$. We assume that ciphertexts $\tilde{\mathtt{ct}}$ are given in Montgomery representation such that $\tilde{\mathtt{ct}} = (\boldsymbol{c}_0\boldsymbol{M}, \boldsymbol{c}_1\boldsymbol{M})$. The conversion to the Montgomery domain can be integrated in the encryption procedure for increased efficiency, and the $\boldsymbol{M}$ factor can be removed during decryption by applying a Montgomery reduction to $[\tilde{\boldsymbol{c}}_\mathbf{0} + \tilde{\boldsymbol{c}}_\mathbf{1}\boldsymbol{s}]_q$. The Montgomery reduction only impacts procedures involving multiplications in $\mathcal{R}_q$ by multiplying the product by $\boldsymbol{M}^{-1}$. Therefore the Montgomery representation is stable with respect to multiplication. Homomorphic additions are not affected by the change in representation. Thus the only impact one has to consider is on homomorphic multiplication. Moreover, we recall that the expansion factor of the ring $\mathcal{R}$ is the quantity defined by $\delta_\mathcal{R} = \sup\{\|\boldsymbol{ab}\|_\infty / \|\boldsymbol{a}\|_\infty \|\boldsymbol{b}\|_\infty \ (\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R} - \{0\}\}$.

### 4.1 Impact of the Montgomery representation in FV

We note that the first step of the FV homomorphic multiplication corresponds to an extension of the polynomials of the ciphertexts to a larger RNS basis, so that

multiplications are computed over $\mathcal{R}$ instead of $\mathcal{R}_q$. In order to improve efficiency, an approximate extension is used [3] and thus the norm of the polynomials is bounded by $\frac{q}{2}(1+\rho)$ for a parameter $\rho > 0$ [3]. A bound on the noise growth of $\tilde{\mathtt{ct}}_{mult} \leftarrow \left(\left[\left\lfloor \frac{t}{q} M \boldsymbol{c}_0^1 \boldsymbol{c}_0^2 \right\rceil\right]_q, \left[\left\lfloor \frac{t}{q} M (\boldsymbol{c}_0^1 \boldsymbol{c}_1^2 + \boldsymbol{c}_1^1 \boldsymbol{c}_0^2) \right\rceil\right]_q, \left[\left\lfloor \frac{t}{q} M \boldsymbol{c}_1^1 \boldsymbol{c}_1^2 \right\rceil\right]_q\right)$ is given in Proposition 1 whose proof can directly be derivated from the one of [3].

**Proposition 1.** *Let* $(\tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1, \tilde{\boldsymbol{c}}_2) = \tilde{\mathtt{ct}}_{mult}$, $r_\infty = \frac{1+\rho}{2}(1+\delta_{\mathcal{R}}\|\boldsymbol{s}\|_\infty)+\delta_R\|M\|_\infty$ *and* $\boldsymbol{v}_i$ *be the inherent noise of* $\mathtt{ct}_i$. *Then* $(\tilde{\boldsymbol{c}}_0+\tilde{\boldsymbol{c}}_1\boldsymbol{s}+\tilde{\boldsymbol{c}}_2\boldsymbol{s}^2)\boldsymbol{M}^{-1} \equiv \Delta\,[\boldsymbol{m}_1\boldsymbol{m}_2]_t+\tilde{\boldsymbol{v}}_{mult}[q]$ *with:*

$$\begin{aligned}\|\tilde{\boldsymbol{v}}_{mult}\|_\infty < \delta_{\mathcal{R}} t(\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty r_\infty + \tfrac{1}{2})(\|\boldsymbol{v}_1\|_\infty + \|\boldsymbol{v}_2\|_\infty) + \tfrac{\delta}{2}\min\|\boldsymbol{v}_i\|_\infty + \delta t|q|_t(r_\infty+1)\\ + \tfrac{\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty}{2}(1+\delta_{\mathcal{R}}\|\boldsymbol{s}\|_\infty(1+\delta_{\mathcal{R}}\|\boldsymbol{s}\|_\infty)) + \tfrac{|q|_t}{2} + 1. \qquad (10)\end{aligned}$$

Now, we assume that we need to relinearize the ciphertext $\tilde{\mathtt{ct}}_{mult} = (\widetilde{\boldsymbol{c}}_0, \widetilde{\boldsymbol{c}}_1, \widetilde{\boldsymbol{c}}_2)$. Before, the following dot products were computed over $\mathcal{R}_q$: $\langle \xi_q(\boldsymbol{c}_2), \mathtt{rlk}_i\rangle$, where $\mathtt{evk}_0 = [\mathcal{P}_{RNS,q}(\boldsymbol{s}^2) + \overrightarrow{\boldsymbol{a}}\boldsymbol{s} + \overrightarrow{\boldsymbol{e}}]_q$ and $\mathtt{evk}_1 = [-\overrightarrow{\boldsymbol{a}}]_q$. The goal of relinearisation is to obtain $\langle \xi_q(\boldsymbol{c}_2), \mathcal{P}_{RNS,q}(\boldsymbol{s}^2)\rangle \equiv \boldsymbol{c}_2\boldsymbol{s}^2 \bmod q$ with a limited increase of the noise. Indeed, modulo $q$ we can write:

$$\begin{cases}\langle \xi_q(\boldsymbol{c}_2), \mathtt{evk}_0\rangle \equiv \boldsymbol{c}_2\boldsymbol{s}^2 + \langle \xi_q(\boldsymbol{c}_2), \overrightarrow{\boldsymbol{a}}\rangle\boldsymbol{s} + \langle \xi_q(\boldsymbol{c}_2), \overrightarrow{\boldsymbol{e}}\rangle \equiv \boldsymbol{c}_2\boldsymbol{s}^2 + \boldsymbol{a'}\boldsymbol{s} + \boldsymbol{e'}\\ \langle \xi_q(\boldsymbol{c}_2), \mathtt{evk}_1\rangle \equiv -\langle \xi_q(\boldsymbol{c}_2), \overrightarrow{\boldsymbol{a}}\rangle \equiv -\boldsymbol{a'}\end{cases} \qquad (11)$$

Now, we need to obtain the Montgomery representation of the output of this relinearization, i.e. a cryptogram like $((\boldsymbol{c}_2\boldsymbol{s}^2 + \boldsymbol{a'}\boldsymbol{s} + \boldsymbol{e'})\boldsymbol{M}, -\boldsymbol{M}\boldsymbol{a'})$.

When the Montgomery representation is used, $\widetilde{\boldsymbol{c}}_2$ replaces $\boldsymbol{c}_2$ in (11). Hence, the relinerization key has to be modified as follows:

$$\mathtt{evkM}_0 = [\boldsymbol{M}^2(\mathcal{P}_{RNS,q}(\boldsymbol{s}^2/\boldsymbol{M}) + \overrightarrow{\boldsymbol{a}}\boldsymbol{s} + \overrightarrow{\boldsymbol{e}})]_q,\ \mathtt{evkM}_1 = [-\boldsymbol{M}^2\overrightarrow{\boldsymbol{a}}]_q$$

In the following equations, we simulate the effect of the Montgomery reduction by introducing a factor $\boldsymbol{M}^{-1} (\mathrm{mod}\, \boldsymbol{\phi}_m)$. Then, we can easily establish the following equalities:

$$\begin{aligned}\langle \xi_q(\widetilde{\boldsymbol{c}}_2), \mathtt{evkM}_0\rangle\boldsymbol{M}^{-1} &= \left(\widetilde{\boldsymbol{c}}_2\boldsymbol{s}^2\boldsymbol{M} + \langle \xi_q(\widetilde{\boldsymbol{c}}_2), \overrightarrow{\boldsymbol{a}}\rangle\boldsymbol{s}\boldsymbol{M}^2 + \langle \xi_q(\widetilde{\boldsymbol{c}}_2), \overrightarrow{\boldsymbol{e}}\rangle\boldsymbol{M}^2\right)\boldsymbol{M}^{-1}\\ &= \widetilde{\boldsymbol{c}}_2\boldsymbol{s}^2 + (\boldsymbol{a''}\boldsymbol{s} + \boldsymbol{e''})\boldsymbol{M}\\ &= (\boldsymbol{c}_2\boldsymbol{s}^2 + \boldsymbol{a''}\boldsymbol{s} + \boldsymbol{e''})\boldsymbol{M}\end{aligned}$$

Similarly, we get $\langle \xi_q(\widetilde{\boldsymbol{c}}_2), \mathtt{evkM}_1\rangle\boldsymbol{M}^{-1} = -\boldsymbol{a''}\boldsymbol{M}$. Hence, we have obtained the Montgomery representation of the output of the relinearization step at no extra cost - both computationally and in terms of noise growth.

### 4.2 Impact of the Montgomery representation in BGV

For the first step of the BGV homomorphic multiplication, no scaling operation is required, and hence noise is not affected by a change in representation. Next, relinearization is applied. An analysis similar to the one in Section 4.1 can be performed, with minor adaptations to the relinearization key. Similarly, one

concludes that the Montgomery reduction introduces no cost neither in terms of computation nor in noise growth.

Finally, one needs to apply scaling so as to manage noise growth. We consider the ciphertext $(\widetilde{\boldsymbol{c}}_0, \widetilde{\boldsymbol{c}}_1)$ in Montgomery representation encrypting $m$. Let $q' \mid q$ and $\boldsymbol{\delta}_i = [-\widetilde{\boldsymbol{c}}_i/t]_{q/q'} \times t$. Then the BGV-scaling function applied to $\widetilde{\boldsymbol{c}}_i$ outputs $\widehat{\boldsymbol{c}}_i = (\widetilde{\boldsymbol{c}}_i + \boldsymbol{\delta}_i) \times \frac{q'}{q}$.

**Lemma 3.** *If* $\|[\boldsymbol{c}_0+\boldsymbol{c}_1\cdot\boldsymbol{s}]_q\|_\infty < \frac{q}{2} - \delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty \frac{q}{q'}\frac{t}{2}(1+\delta_{\mathcal{R}}\cdot\|\boldsymbol{s}\|_\infty)$ *and* $q = q' \bmod t$, *then*

$$[(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1}]_{q'} = [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_q \bmod t \tag{12}$$

$$\|[(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1}]_{q'}\|_\infty \leqslant \frac{q'}{q}\|[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_q\|_\infty + \delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty \frac{t}{2}(1 + \delta_{\mathcal{R}}\|\boldsymbol{s}\|_\infty) \tag{13}$$

*Proof.* The proof is similar to the proof of lemma 4 of BGV original paper.

By definition of $\widetilde{\boldsymbol{c}}_i$, we have:

$$[(\widetilde{\boldsymbol{c}}_0 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1}]_q = [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_q = \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} - q\boldsymbol{u}.$$

By definition of $\widehat{\boldsymbol{c}}_i$, we can write:

$$\begin{aligned}(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1} &= \tfrac{q'}{q}(\widetilde{\boldsymbol{c}}_0 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s} + \boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1} \\ &= \tfrac{q'}{q}(\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}) + \tfrac{q'}{q}(\boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1} \\ &= \tfrac{q'}{q}[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_q + q'\boldsymbol{u} + \tfrac{q'}{q}(\boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1}.\end{aligned} \tag{14}$$

Moreover, since $\|\boldsymbol{\delta}_i\|_\infty \leqslant \frac{qt}{2q'}$, we get the following bound $\|(\boldsymbol{\delta}_0 + \boldsymbol{\delta}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1}\|_\infty \leqslant \delta_{\mathcal{R}}\frac{qt}{2q'}\|\boldsymbol{M}^{-1}\|_\infty(1 + \delta_{\mathcal{R}}\|\boldsymbol{s}\|_\infty)$. Thus, from the above and the hypothesis on $\|[(\widetilde{\boldsymbol{c}}_0 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1}]_q\|_\infty = \|[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_q\|_\infty$, we deduce that

$$\|(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1} - q'\boldsymbol{u}\|_\infty < q'/2$$

and then that

$$(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1} - q'\boldsymbol{u} = [(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1}]_{q'}$$

Hence, from this previous equality and by bounding the norm of last member of (14), we obtain (13).

Finally, we get (12) by:

$$\begin{aligned}[(\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1}]_{q'} &= (\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1 \cdot \boldsymbol{s})\boldsymbol{M}^{-1} - q'\boldsymbol{u} \\ &= (\widetilde{\boldsymbol{c}}_0 + \widetilde{\boldsymbol{c}}_1 \cdot \boldsymbol{s})/\boldsymbol{M} - q\boldsymbol{u} \bmod t \ (\widehat{\boldsymbol{c}}_i = \widetilde{\boldsymbol{c}}_i \bmod t; q = q' \bmod t) \\ &= \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} - q\boldsymbol{u} \bmod t \ (\text{def. of } \widetilde{\boldsymbol{c}}) \\ &= [\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_q \bmod t\end{aligned}$$

From this lemma, we can see that the Montgomery representation of the ciphertext impacts the scaling by adding an extra factor $\delta_{\mathcal{R}}\|\boldsymbol{M}^{-1}\|_\infty$ to the last term on the bound of the hypothesis and of (13).

### 4.3 Overall impact on noise growth

For both BGV and FV, the norms $\|\boldsymbol{M}\|_\infty$, $\|\boldsymbol{M}^{-1}\|_\infty$ and the expansion factor $\delta_\mathcal{R}$ are involved in the noise growth due to the scaling steps performed with the Montgomery reduction. When $m$ is a power of two, $\delta_\mathcal{R}$ is equal to $n$. However, for other $m$ it can be larger than that. Let us consider $\mathcal{F}_m : \mathbb{Q}_{2n-2}[X] \to \mathbb{Q}[X]/(\boldsymbol{\phi}_m(X))$, so that $\mathcal{F}_m(\boldsymbol{a}) = \boldsymbol{a} \bmod \boldsymbol{\phi}_m$ for every $\boldsymbol{a} \in \mathbb{Q}[X]$ of degree lesser than or equal to $2n-2$.

**Lemma 4.** *Let $m$ be a positive integer and let $\mathcal{R} = \mathbb{Z}[X]/(\boldsymbol{\phi}_m(X))$, with $\deg(\boldsymbol{\phi}_m) = n$. If $\delta_\mathcal{R}$ denotes the expansion factor of the ring $\mathcal{R}$, then $\delta_\mathcal{R} \leq n \cdot \|\mathcal{F}_m\|_\infty$ .*

These three parameters are given in Table 3 for the different cyclotomic polynomials considered in this paper. Assuming that distributions $\chi_{key}$ and $\chi_{err}$ output elements whose infinite norms are bounded by $B_{key}$ and $B_{err} = 6\sigma_{err}$, we analyse which depth can be reached in a multiplicative tree.

**FV** : The initial noise of a ciphertext is at most $V_{init} = B_{err}(1 + 2\delta B_{key})$ [11]. We recall that $r_\infty = \frac{1+\rho}{2}(1 + \delta_\mathcal{R} B_{key}) + \delta_R \|\boldsymbol{M}\|_\infty$, the output of a tree of depth $L$ has a noise bounded by $C_1^L V + LC_1^{L-1}C_2$ (cf. [6], Lem. 9) with:

$$\begin{cases} C_1 = 2\delta_\mathcal{R} t(\delta_\mathcal{R}\|\boldsymbol{M}^{-1}\|_\infty r_\infty + \frac{1}{2}) + \frac{\delta_\mathcal{R}}{2} \\ C_2 = \delta_\mathcal{R} t|q|_t(r_\infty + 1) + \delta_\mathcal{R}\|\boldsymbol{M}^{-1}\|_\infty \frac{1+\delta_\mathcal{R} B_{key}(1+\delta B_{key})}{2} + \frac{|q|_t}{2} + 1 \\ \qquad +k(1 + \delta_\mathcal{R} B_{key}(1 + \delta_\mathcal{R} B_{key})) + \delta_\mathcal{R} k B_{err} 2^{\nu+1} \end{cases}$$

We denote by $L_{max} = \max\{L \in \mathbb{N} \mid C_1^L V + LC_1^{L-1}C_2 \leqslant B_{dec}\}$ the depth allowed by the homomorphic multiplication where $B_{dec}$ corresponds to the decryption bound given by the full RNS version of FV [3].

**BGV** : As long as a ciphertext satisfies the condition on Lemma 3, one can perform a scaling operation and thus an homomorphic multiplication. Initially $\|\boldsymbol{c}_0 + \boldsymbol{c}_1\boldsymbol{s}\|_\infty \leqslant V_{init} = t/2 + tB_{err}(2\delta_\mathcal{R} B_{key} + 1)$. By assuming that after each scaling, the size of $q$ is reduced by $\omega$ bits and let $l_{\omega,q} = \lceil \log_\omega q \rceil$, then the growth of the size of $\boldsymbol{c}_0 + \boldsymbol{c}_1\boldsymbol{s}$, denoted $V$, after one multiplication can be expressed with:

$$\frac{q'}{q}(\delta_\mathcal{R} V^2 + B_{relin}) + \delta_R \|M^{-1}\|_\infty t \frac{1+\delta_\mathcal{R} B_{key}}{2}$$

where $B_{relin} = \frac{\delta_\mathcal{R}}{2}\ell_{\omega,q}\omega B_{err} B_{key}$ represents the noise caused by the relinearisation step.

In Table 3 we present the maximal theoretical depths for BGV and FV with or without the use of the Montgomery reduction. For these computations we have taken parameters $B_{key} = 1$, $\sigma_{err} = 2\sqrt{n}$ and a number $k$ of 62-bits moduli to get the largest size for $q$ ensuring at least 80-bits of security according to [2].

We notice that the depths of BGV are almost unchanged with the usage of the Montgomery reduction. However for FV the depths are far smaller with a Montgomery representation. This behavior has been confirmed in practice.

| $m$ | $n$ | $k$ | $\|\boldsymbol{M}\|_\infty$ | $\|\boldsymbol{M}^{-1}\|_\infty$ | $\delta_{\mathcal{R}}$ | $L_{BGV}$ | $L^M_{BGV}$ | $L_{FV}$ | $L^M_{FV}$ |
|---|---|---|---|---|---|---|---|---|---|
| 4369 | 4096 | 2 | 1 | 1 | $35n$ | 1(1) | 1(1) | 1(4) | 1(3) |
| 13107 | 8192 | 5 | 2 | 1 | $205n$ | 4(4) | 4(4) | 6(17) | 4(10) |
| 21845 | 16384 | 11 | 2 | 1 | $739n$ | 10(10) | 6(10) | 13(40) | 8(22) |
| 32767 | 27000 | 18 | 1 | 9 | $2621n$ | 8(17) | 7(17) | 19(66) | 12(39) |
| 65535 | 32768 | 22 | 4 | 1 | $9886n$ | 7(21) | 7(21) | 22(80) | 14(45) |

**Table 3.** Theoretical depths with and without Montgomery reduction. Values in parenthesis are the depths observed in practice.

### 4.4 Mixing Optimized Barrett and Montgomery reductions

Considering the non-negligible impact of the Montgomery representation on the multiplicative depth of FV, a more robust strategy for this cryptosystem corresponds to a mixed Barrett/Montgomery approach. Alg. 2 is used during the first stage of homomorphic multiplication, with ciphertexts not exploiting a Montgomery representation. This avoids the noise growth caused by the Montgomery factor. Nonetheless, the Montgomery reduction can still be used during the relinearization stage, since we have seen that this does not cause a larger noise growth. To obtain a valid result, the relinearisation key needs to be modified, by replacing the factor $\boldsymbol{M}^2$ of the Montgomery approach by $\boldsymbol{M}$.

## 5 Experimental Results

The proposed methods for polynomial reduction were implemented using C++, and compiled with `gcc` using the optimization flag `-O3`. All the experimental results presented herein were measured on an i7-5960X, running at 3.0 GHz with 32 GB of main memory. No parallelism was exploited.

In Figure 1, one can find the execution timings of polynomial reduction, using NFL for power-of-two cyclotomics [1]; the unoptimized and optimized Barrett reductions and the Montgomery reduction for non-power-of-two cyclotomics. In order to highlight the gain brought by our algorithms compare to generic ones we also compare with NTL's reduction using preconditioning [16]. All timings were normalized based on the number of batching slots $\ell$, and executed for a single moduli of 62-bits. One find the straightforward application of Barrett reduction, to be more efficient than the preconditioned methods employed in the NTL library. Moreover, speed-ups of up to 1.95 and 2.55 were achieved for the optimized Barrett and Montgomery algorithms when compared with the unoptimized Barrett reduction. The figure suggests that using power-of-two cyclotomics is not scalable with respect to the throughput. In contrast, the remaining approaches present very little variation when considering the execution timing per batching slot for the different $m$ given in Table 3. It should be noted that using larger values of $m$ enables FHE parameters with a larger multiplicative depth.

The aforementioned reduction methods were used to implement the homomorphic multiplication operations of the FV and BGV schemes. One can find in

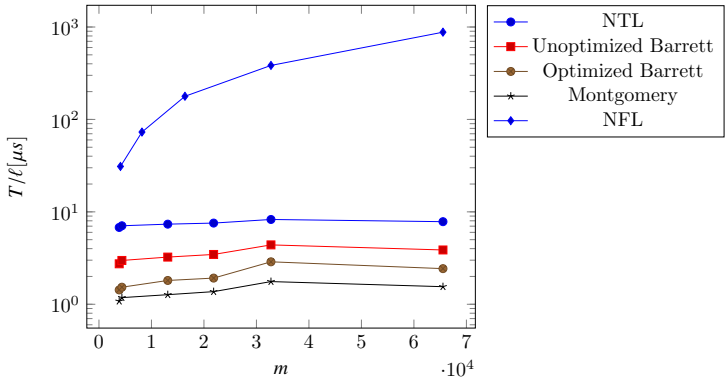


**Fig. 1.** Execution time per batching slot $T/\ell[\mu s]$ for multiple reduction strategies and $m$th cyclotomic polynomials. The y-axis is in logarithmic scale

Figures 2 and 3 the execution times of the homomorphic multiplication of two freshly encrypted ciphertexts for FV and BGV, respectively with parameters given in Table 3. The experimental results for NFL are omitted due to its low performance with respect to the timing per batching slot.

Unlike with Figure 1, the execution time of homomorphic multiplication increases significantly with increasing $m$. This trend is explained by the relinearization procedure, which requires a number of NTTs that increases quadratically with $\log_2 q$. Nevertheless, the employed reduction procedure plays a preponderant role in the efficiency of the homomorphic multiplication. Indeed, one achieves speed-ups of up to 1.37 and 1.24 when comparing the homomorphic multiplication exploiting the optimized Barrett reduction with the one exploiting the unoptimized Barrett method for the FV and BGV schemes, respectively. Since with the mixed Barrett/Montgomery approach, required by the FV scheme, one is not able to fully take advantage of the gains brought forth by the Montgomery representation, one achieves speed-ups similar to those of the optimized Barrett reduction. In contrast, for BGV, one can exploit the Montgomery representation throughout the whole procedure, leading to speed-ups of up to 1.32.

The speed-up of the proposed methods decreases as the degree $n$ of the cyclotomic, and thus $\log_2 q$, get larger due to the increasing complexity of the relinearization procedure. This suggests that they are most beneficial when one needs to homomorphically evaluate small circuits. Since most of practical applications of FHE [14, 19, 26] have circuits with small depth, the proposed methods have a wide range of applicability.

## Conclusion

In this paper, the arithmetic of non-power-of-two cyclotomics has been considered and improved. Two methods for polynomial reduction have been proposed:

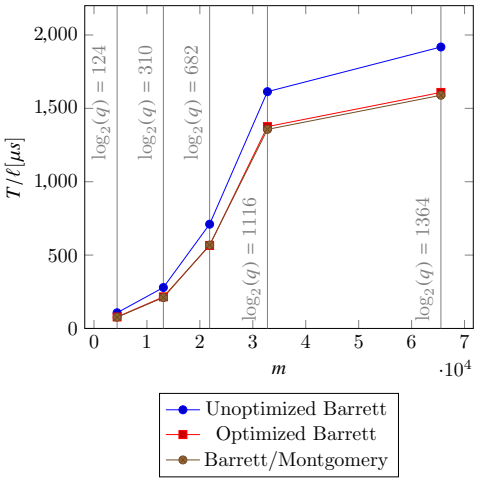


**Fig. 2.** Execution time per batching slot $T/\ell[\mu s]$ for the homomorphic multiplication operation of FV with several reduction strategies and $m$th cyclotomic polynomials.

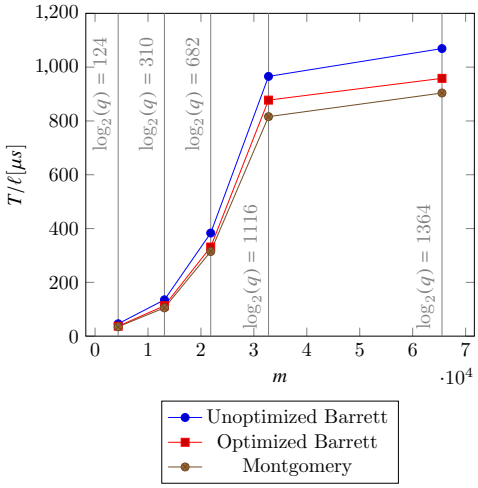


**Fig. 3.** Execution time per batching slot $T/\ell[\mu s]$ for the homomorphic multiplication operation of BGV with several reduction strategies and $m$th cyclotomic polynomials.

one based on the Barrett reduction and the other on a Montgomery representation. The optimized Barrett algorithm does not offer a better computational complexity than the Montgomery reduction. However, since it does not require changes in the representation of ciphertexts, it provides for a slower noise growth, making it more amenable to application in the FV homomorphic scheme than the Montgomery approach. In contrast, the Montgomery approach is more suitable for the BGV scheme. Moreover, since the Montgomery reduction does not rely on the properties of cyclomic polynomials, it can be used on schemes relying on other kinds of polynomials. Experimental results have shown the proposed methods to be more efficient than those employed in the NTL library. Furthermore, speed-ups of up to 1.95 and 2.55 have been obtained in an i7-5960X when comparing the optimized Barrett and Montgomery reductions with the unoptimized Barrett reduction, respectively. Finally, the polynomial reductions have been incorporated into the homomorphic multiplication procedures of FV and BGV, producing speed-ups of up to 1.37.

## References

1. Carlos Aguilar-Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrède Lepoint. *Topics in Cryptology - CT-RSA 2016: The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, chapter NFLlib: NTT-Based Fast Lattice Library, pages 341–356. Springer International Publishing, Cham, 2016.

2. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9:169–203, October 2015.
3. Jean-Claude Bajard, Julien Eynard, M. Anwar Hasan, and Vincent Zucca. A Full RNS Variant of FV like Somewhat Homomorphic Encryption Schemes. *Selected Areas in Cryptography - SAC*, 2016.
4. Jean-Claude Bajard, Laurent Imbert, and Christophe Negre. Arithmetic Operations in Finite Fields of Medium Prime Characteristic Using the Lagrange Representation. *IEEE Transactions on Computers*, 55:1167–1177, 2006.
5. P. Barrett. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 311–323. Springer, 1986.
6. JoppeW. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In Martijn Stam, editor, *Cryptography and Coding*, volume 8308 of *Lecture Notes in Computer Science*, pages 45–64. Springer Berlin Heidelberg, 2013.
7. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *In Advances in Cryptology - Crypto 2012, volume 7417 of Lecture*, 2012.
8. Zvika Brakerski, Vinod Vaikuntanathan, and Craig Gentry. Fully homomorphic encryption without bootstrapping. In *In Innovations in Theoretical Computer Science*, 2012.
9. Eric Crockett and Chris Peikert. $\lambda \circ \lambda$: Functional lattice cryptography. Cryptology ePrint Archive, Report 2015/1134, 2015. `http://eprint.iacr.org/2015/1134`.
10. Wei Dai and Berk Sunar. *cuHE: A Homomorphic Encryption Accelerator Library*, pages 169–186. Springer International Publishing, Cham, 2016.
11. Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptology ePrint Archive*, 2012.
12. Michael Filaseta. On coverings of the integers associated with an irreducibility theorem of A. Schinzel. Number theory for the millennium, II (Urbana, IL, 2000), A K Peters, Natick, MA, 2002, 1-24.
13. Craig Gentry, Shai Halevi, and Nigel P. Smart. *Homomorphic Evaluation of the AES Circuit*, pages 850–867. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
14. Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 201–210. JMLR.org, 2016.
15. Sigrun Goluch. The development of homomorphic cryptography. Master's thesis, Vienna University of Technology, Austria, 2011.
16. Shai Halevi, Tzipora Halevi, Victor Shoup, and Noah Stephens-Davidowitz. Implementing bp-obfuscation using graph-induced encoding. Cryptology ePrint Archive, Report 2017/104, 2017. `http://eprint.iacr.org/2017/104`.
17. Shai Halevi and Victor Shoup. Algorithms in helib. In *CRYPTO*, pages 554–571. Springer, 2014.
18. David Harvey. Faster arithmetic for number-theoretic transforms. *CoRR*, abs/1205.2926, 2012.
19. Alhassan Khedr, P. Glenn Gulak, and Vinod Vaikuntanathan. SHIELD: scalable homomorphic implementation of encrypted data-classifiers. *IACR Cryptology ePrint Archive*, 2014:838, 2014.

20. Kim Laine and Rachel Player. Simple encrypted arithmetic library - seal (v2.0). Technical report, September 2016.
21. Patrick Longa and Michael Naehrig. *Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography*, pages 124–139. Springer International Publishing, Cham, 2016.
22. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *On Ideal Lattices and Learning with Errors over Rings*, pages 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
23. Paulo Martins and Leonel Sousa. *Enhancing Data Parallelism of Fully Homomorphic Encryption*, pages 194–207. Springer International Publishing, Cham, 2017.
24. Christoph M. Mayer. Implementing a toolkit for ring-lwe based cryptography in arbitrary cyclotomic number fields. Cryptology ePrint Archive, Report 2016/049, 2016. `http://eprint.iacr.org/2016/049`.
25. Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.
26. Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, New York, NY, USA, 2011. ACM.
27. Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.
28. N. P. Smart and F. Vercauteren. Fully homomorphic simd operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.

# A Proofs

## A.1 Correctness of Alg. 1

Since ring $\mathbb{F}_q[X]$ is Euclidean, we can write Euclidean division of $\boldsymbol{c}$ by $\boldsymbol{\Phi}_m$, with a unique remainder $\boldsymbol{r}$ of degree at most $n-1$: $\boldsymbol{c} = \lfloor \boldsymbol{c}/\boldsymbol{\Phi}_m \rfloor \boldsymbol{\Phi}_m + \boldsymbol{r}$. Let $a, b \geqslant 0$ be integers. From the previous equation we get the following equivalent equations over the field of rational fractions of $\mathbb{F}_q[X]$:

$$
\begin{aligned}
& \frac{X^{n+a}}{\boldsymbol{\Phi}_m} \cdot \frac{\boldsymbol{c}}{X^{n-b}} && = \left( \left\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r}}{\boldsymbol{\Phi}_m} \right) \cdot X^{a+b} \\
\Leftrightarrow \quad & \left( \left\lfloor \frac{X^{n+a}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r_1}}{\boldsymbol{\Phi}_m} \right) \cdot \left( \left\lfloor \frac{\boldsymbol{c}}{X^{n-b}} \right\rfloor + \frac{\boldsymbol{r_2}}{X^{n-b}} \right) && = \left( \left\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r}}{\boldsymbol{\Phi}_m} \right) \cdot X^{a+b} \\
\Leftrightarrow \quad & \left\lfloor \frac{X^{n+a}}{\boldsymbol{\Phi}_m} \right\rfloor \cdot \left\lfloor \frac{\boldsymbol{c}}{X^{n-b}} \right\rfloor + \boldsymbol{r_a} + \boldsymbol{r_{\alpha+b}} + \boldsymbol{r'} && = \left( \left\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r}}{\boldsymbol{\Phi}_m} \right) \cdot X^{a+b} \\
\Leftrightarrow \quad & \left\lfloor \frac{\lfloor X^{n+a}/\boldsymbol{\Phi}_m \rfloor \cdot \lfloor \boldsymbol{c}/X^{n-b} \rfloor}{X^{a+b}} \right\rfloor X^{a+b} + \boldsymbol{r''} + \boldsymbol{r_a} + \boldsymbol{r_{\alpha+b}} + \boldsymbol{r'} && = \left( \left\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r}}{\boldsymbol{\Phi}_m} \right) \cdot X^{a+b} \\
\Leftrightarrow \quad & \left\lfloor \frac{\lfloor X^{n+a}/\boldsymbol{\Phi}_m \rfloor \cdot \lfloor \boldsymbol{c}/X^{n-b} \rfloor}{X^{a+b}} \right\rfloor + \frac{\boldsymbol{r''} + \boldsymbol{r_a} + \boldsymbol{r_{\alpha+b}} + \boldsymbol{r'}}{X^{a+b}} && = \left\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \right\rfloor + \frac{\boldsymbol{r}}{\boldsymbol{\Phi}_m} \quad .
\end{aligned}
$$

Furthermore, $\deg(\boldsymbol{r_1}), \deg(\boldsymbol{r_2}) < n$, $\deg(\boldsymbol{r_a}) < a$, $\deg(\boldsymbol{R_{\alpha+b}}) < \alpha + b$, $\deg(\boldsymbol{R'}) < 0$, and $\deg(\boldsymbol{R''}) < a+b$. By choosing $b \geqslant 0$ and $a \geqslant \alpha$, the right term of left member of last equation above have a degree smaller than 0. In this case, we obtain an equality between the two floored polynomials. So, we take $b = 0$ and $a = \alpha$, and we get that $\lfloor \frac{\boldsymbol{c}}{\boldsymbol{\Phi}_m} \rfloor$ is equal to the flooring of the left part of last equation, which is precisely what Alg. 1 computes. As the product $\lfloor X^{n+\alpha}/\boldsymbol{\Phi}_m \rfloor \cdot \lfloor \boldsymbol{c}/X^n \rfloor$

is of degree strictly smaller than $2\alpha + 1$, the computation can be done with an NTT of size $A = \mathcal{N}_2(2\alpha + 1)$.

To finish, we notice that the result of the computation of $\boldsymbol{r} = \boldsymbol{c} - \lfloor \boldsymbol{c}/\boldsymbol{\Phi}_m \rfloor \times \boldsymbol{\Phi}_m$ has a degree strictly smaller than $n$. Moreover, the polynomial $\boldsymbol{c'}$ at line 5 is nothing but $\lfloor \boldsymbol{c}/\boldsymbol{\Phi}_m \rfloor \times \boldsymbol{\Phi}_m \bmod X^{\tilde{n}} - 1$. Indeed, the reduction modulo $X^{\tilde{n}} - 1$ is a consequence of the NTT based polynomial product in dimension $\tilde{n}$. Thus, at the end we have that $\boldsymbol{c'} - \boldsymbol{d} = (\boldsymbol{c} - \lfloor \boldsymbol{c}/\boldsymbol{\Phi}_m \rfloor \times \boldsymbol{\Phi}_m) \bmod (X^{\tilde{n}} - 1) = \boldsymbol{c} \bmod \boldsymbol{\Phi}_m$. The last equality holds precisely because the degree of $\boldsymbol{c} \bmod \boldsymbol{\Phi}_m$ is strictly smaller than $\tilde{n}$.

### A.2 Proof of Lemma 1

The first point is a direct consequence from Lemma 2 in [12]. Since $m$ is not a power of two, $m$ cannot divide $N$. By denoting $m = 2^r m'$ with $m' > 1$ an odd integer we have $n = 2^{r-1}\varphi(m')$, thus $2n = 2^r \varphi(m')$ and then if $N$ divides $m$, $\mathcal{N}_2(\varphi(m')) = 1$ which is not possible since $m' \geqslant 3$. Therefore $N$ and $m$ do not divide each others and we can apply Lemma 2 from [12].

Let $\alpha$ be a root of $\boldsymbol{\phi}_m$ in the algebraic closure of $\mathbb{Z}_p$. If $\alpha$ is also a root of $X^N - 1$ then $\alpha^N = 1$, since $\alpha$ is of order $m$ by definition of $\boldsymbol{\phi}_m$ it implies that $m$ divide $N$ which is impossible since $N$ is a power of two and $m$ is not. So, $\boldsymbol{\phi}_m$ and $X^N - 1$ are coprime on the algebraic closure of $\mathbb{Z}_p$ thus in $\mathbb{Z}_p$. The second point comes from Bezout equality in $\mathbb{Z}_p$ and from the fact that $X^N - 1 \equiv (X^{N/2} - 1)(X^{N/2} + 1) \bmod p$.

### A.3 Correctess of Alg. 4

First we notice that since $\boldsymbol{c}$ is a polynomial of degree smaller than $N/2$ we have $(\boldsymbol{c}(\omega), \ldots, \boldsymbol{c}(\omega^{N-1})) = \mathtt{NTT}_{N/2,\omega^2}(\xi_{N/2,\omega}(\boldsymbol{c}))$ where $\xi_{N/2,\omega}(c_0, c_1, \ldots, c_{N/2-1}) = (c_0, c_1\omega, \ldots, c_{N/2-1}\omega^{N/2-1})$. Therefore the polynomial $\boldsymbol{c'}$ recovered at the first line of Alg. 4 is nothing but $\xi_{N/2,\omega}(\boldsymbol{c})$. The second line of the algorithm is just the computation of $\xi_{N/2,\omega^{-1}}(\boldsymbol{c'})$ to recover $\boldsymbol{c}$ from $\boldsymbol{c'}$. Once $\boldsymbol{c}$ recovered we just need to compute and return $\mathtt{NTT}_{N/2,\omega^2}(\boldsymbol{c})$ which is precisely the final step of the algorithm.

### A.4 Proof of Lemma 4

Let $\boldsymbol{a}$ and $\boldsymbol{b}$ two elements of $\mathcal{R} - \{0\}$. They naturally embed in $\mathbb{Z}_{n-1}[X] \subset \mathbb{Q}_{2n-2}[X]$. We can write $\|\boldsymbol{ab}\|_\infty \leq n\|\boldsymbol{a}\|_\infty\|\boldsymbol{b}\|_\infty$. As the product $\boldsymbol{ab}$ has degree at most $2n - 2$ with coefficients in $\mathbb{Z}$, it belongs to $\mathbb{Q}_{2n-2}[X]$. Since $\mathcal{F}_m$ is a linear map between two vector spaces of finite dimension it is continuous, then we obtain $\|\mathcal{F}_m(\boldsymbol{ab})\|_\infty \leq \|\mathcal{F}_m\|_\infty \cdot \|\boldsymbol{ab}\|_\infty \leq n \cdot \|\mathcal{F}_m\|_\infty \cdot \|\boldsymbol{a}\|_\infty\|\boldsymbol{b}\|_\infty$ .