An asymptotically faster version of FV supported on HPR

Jean-Claude Bajard*, Julien Eynard†, Paulo Martins‡, Leonel Sousa‡ and Vincent Zucca.§
*Sorbonne Université, CNRS, Inria, Institut de Mathématiques de Jussieu – Paris Rive Gauche, France.
jean-claude.bajard@sorbonne-universite.fr
†Univ. Grenoble Alpes, CEA, LETI, DSYS, CESTI, F-38000 Grenoble.
julien.eynard@cea.fr
‡INESC-ID, Instituto Superior Técnico, Universidade de Lisboa.
paulo.sergio@netcabo.pt, las@inesc-id.pt
§imec-COSIC KU Leuven.
vincent.zucca@kuleuven.be

***Abstract*—State-of-the-art implementations of homomorphic encryption exploit the Fan and Vercauteren (FV) scheme and the Residue Number System (RNS). While the RNS breaks down large integer arithmetic into smaller independent channels, its non-positional nature makes operations such as division and rounding hard to implement, and makes the representation of small values inefficient. In this work, we propose the application of the Hybrid Position-Residues Number System representation to the FV scheme. This is a positional representation of large radix where the digits are represented in RNS. It inherits the benefits from RNS and allows to accelerate the critical division and rounding operations while also making the representation of smaller values more compact. This directly benefits the decryption and the homomorphic multiplication procedures, reducing their asymptotic complexity, in dimension n, from $\mathcal{O}(n^2 \log n)$ to $\mathcal{O}(n \log n)$ and from $\mathcal{O}(n^3 \log n)$ to $\mathcal{O}(n^3)$, respectively and has resulted in noticeable speedups when experimentally compared to related art RNS implementations.**

***Index Terms*—Fan-Vercauteren scheme, Residue Number System, Homomorphic Encryption**

I. Introduction

Since Gentry's uncovering of Fully Homomorphic Encryption (FHE) [1], a large amount of research has been dedicated to improving its efficiency. While the Fan and Vercauteren (FV) [2] proposal is one of the most efficient homomorphic schemes, it requires computationally expensive operations on polynomials with large coefficients. The Residue Number System (RNS) [3] allows to break down large integer arithmetic into many independent smaller channels. With it, additions and multiplications are computed independently for each channel, resulting in a reduction of complexity when compared to traditional positional representations. As a result, state-of-the-art implementations of FV are based on the RNS [4], [5].

However, the non-positional nature of RNS makes it more difficult to implement operations such as division and rounding, which are required by both FV homomorphic multiplication and decryption procedures. Furthermore, while for positional representations the size of the representation of a number is proportional to its magnitude, the RNS representation of a number is as large as the largest number representable in it. During the relinearisation procedure, which dominates the asymptotic complexity of the homomorphic multiplication, polynomial coefficients need to be decomposed to values of smaller magnitude. Hence, after applying this decomposition, the RNS becomes a sub-optimal representation.

This work proposes to exploit instead the Hybrid Position-Residues Number System (HPR) [6] representation of numbers to accelerate the cryptographic operations of FV. This hybrid representation is of a positional nature, but each digit is represented in RNS. It not only inherits the parallel nature of RNS but it also makes the relinearisation, and division and rounding operations more efficient. As a result, in dimension n, the complexity of FV decryption is asymptotically reduced from $\mathcal{O}(n^2 \log n)$ to $\mathcal{O}(n \log n)$. Similarly, the complexity of homomorphic multiplications is reduced from $\mathcal{O}(n^3 \log n)$ to $\mathcal{O}(n^3)$. Complexity gains are translated in practice into speedups of up to 8.0 and 8.6 for the decryption operation and of up to 2.0 and 1.6 for the homomorphic multiplication, for the parameters considered in this article and two state-of-the-art schemes, with an upward trend for larger parameters.

The remainder of this article is organised as follows: the FV cryptosystem and the HPR representation are described in Section II. Based on these techniques, our construction is given in Section III. Afterwards, Section IV reviews related art, and Sections V and VI compare the proposed techniques with state-of-the-art methods from theoretical and experimental perspectives, respectively.

II. Background

The ring $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$ can be thought of as a set of polynomials with integer coefficients and degree smaller than n. $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ denotes the polynomials in $\mathcal{R}$ whose coefficients are reduced modulo an integer q. When $q \equiv 1 \pmod{2n}$, elements of $\mathcal{R}_q$ can be represented in the Number Theoretic Transform (NTT)-domain, which allows for coefficient-wise additions and multiplications [7]. Henceforth, boldface letters are used to denote polynomials and the infinity norm $\|\cdot\|_\infty$ is the largest absolute value of the polynomial's coefficients. The expansion factor associated to $\mathcal{R}$ is defined as $\delta_{\mathcal{R}} = \sup\left\{\frac{\|\boldsymbol{a}\cdot\boldsymbol{b}\|_\infty}{\|\boldsymbol{a}\|_\infty\|\boldsymbol{b}\|_\infty}, \text{ for } \boldsymbol{a},\boldsymbol{b} \in \mathcal{R} - \{0\}\right\}$. $\boldsymbol{a} \leftarrow D$ and

$\boldsymbol{a} \leftarrow \chi$ are used to denote that $\boldsymbol{a}$ is drawn uniformly at random from the set D or according to a distribution χ, respectively. The notation $|\cdot|_q$ (resp. $[\cdot]_q$) is used to denote the residue modulo q in $[0, q)$ (resp. in $[-q/2, q/2)$). Moreover, for $a \in \mathbb{Q}$, $b = \lfloor a \rceil$ is used to denote the closest integer $b \in \mathbb{Z}$ to a. The computational complexity is herein evaluated in terms of the amount of Elementary Modular Multiplications (EMMs), whose operands fit a single machine word, floating point operations (FPOs) and forward and inverse NTTs required.

A. FV cryptosystem

A private-key of the FV scheme [2] is generated as a polynomial $\boldsymbol{s} \leftarrow \chi_{key}$, and the corresponding public-key is computed as:

$$\boldsymbol{b} = \left([-(\boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e})]_q, \boldsymbol{a}\right) \in \mathcal{R}_q^2$$

with $\boldsymbol{a} \leftarrow \mathcal{R}_q$ drawn uniformly at random and $\boldsymbol{e} \leftarrow \chi_{err}$. χ_{key} is such that for $\boldsymbol{s} \leftarrow \chi_{key}$, $\|\boldsymbol{s}\|_\infty \leqslant B_{key}$ for a small B_{key}, while the distribution χ_{err} has standard deviation σ_{err} such that elements sampled from this distribution have their infinite norm smaller than $B_{err} = 6\sigma_{err}$ with very high probability.

A relinearisation key is defined with respect to a pair of functions $\mathfrak{D}$ and $\mathfrak{P}$, such that $[\langle \mathfrak{D}(\boldsymbol{a}), \mathfrak{P}(\boldsymbol{b}) \rangle]_q = [\boldsymbol{a} \cdot \boldsymbol{b}]_q$ and the infinite norm of $\mathfrak{D}(\boldsymbol{a})$ is "small". The relinearisation key, which can be interpreted as a vector of pseudo-encryptions of $\mathfrak{P}(\boldsymbol{s}^2)$, is defined as:

$$\overrightarrow{\texttt{rlk}} = \left(\left[\mathfrak{P}(\boldsymbol{s}^2) - (\overrightarrow{\boldsymbol{e}} + \overrightarrow{\boldsymbol{a}} \cdot \boldsymbol{s})\right]_q, \overrightarrow{\boldsymbol{a}}\right) \in \mathcal{R}_q^{2l}$$

where $\overrightarrow{\boldsymbol{e}} \leftarrow \chi_{err}^l$ and $\overrightarrow{\boldsymbol{a}} \leftarrow \mathcal{R}_q^l$. A ciphertext $\texttt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$, encrypting $[\boldsymbol{m}]_t \in \mathcal{R}_t$, satisfies $\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} = \Delta[\boldsymbol{m}]_t + \boldsymbol{v} \bmod q$, where $\Delta = \lfloor q/t \rfloor$. $\texttt{ct}$ can be correctly decrypted as long as the noise $\boldsymbol{v}$ satisfies $\|\boldsymbol{v}\|_\infty < \Delta/2 - |q|_t/2$, through:

$$\boldsymbol{m} = \left[\left\lfloor \frac{t}{q}[\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}]_q \right\rceil\right]_t. \tag{1}$$

The homomorphic addition of two ciphertexts corresponds to the pairwise addition of their coefficients modulo q. The homomorphic multiplication of $\texttt{ct}$ and $\texttt{ct}'$ encrypting $[\boldsymbol{m}]_t$ and $[\boldsymbol{m}']_t$ is performed in two steps. First, we compute the three elements ciphertext:

$$\widehat{\texttt{ct}}_{\text{mult}} = \left(\left[\left\lfloor \frac{t}{q}\boldsymbol{c}_0 \cdot \boldsymbol{c}_0' \right\rceil\right]_q, \left[\left\lfloor \frac{t}{q}(\boldsymbol{c}_0 \cdot \boldsymbol{c}_1' + \boldsymbol{c}_1 \cdot \boldsymbol{c}_0') \right\rceil\right]_q, \left[\left\lfloor \frac{t}{q}\boldsymbol{c}_1 \cdot \boldsymbol{c}_1' \right\rceil\right]_q\right) \in \mathcal{R}_q^3 \tag{2}$$

satisfying $\widehat{\boldsymbol{c}}_0 + \widehat{\boldsymbol{c}}_1\boldsymbol{s} + \widehat{\boldsymbol{c}}_2\boldsymbol{s}^2 = \Delta[\boldsymbol{m} \cdot \boldsymbol{m}']_t + \widehat{\boldsymbol{v}} \bmod q$. In a second step, $\widehat{\texttt{ct}}_{\text{mult}}$ is relinearised, by multiplying $\mathfrak{D}(\widehat{\boldsymbol{c}}_2)$ by $\overrightarrow{\texttt{rlk}}$ and adding the result to $(\widehat{\boldsymbol{c}}_0, \widehat{\boldsymbol{c}}_1)$:

$$\texttt{ct}_{\text{mult}} = \left(\left[\widehat{\boldsymbol{c}}_0 + \left\langle \mathfrak{D}(\widehat{\boldsymbol{c}}_2), \overrightarrow{\texttt{rlk}}_0 \right\rangle\right]_q, \left[\widehat{\boldsymbol{c}}_1 + \left\langle \mathfrak{D}(\widehat{\boldsymbol{c}}_2), \overrightarrow{\texttt{rlk}}_1 \right\rangle\right]_q\right).$$

The result $\texttt{ct}_{\text{mult}}$, can be decrypted with (1) to produce $[\boldsymbol{m} \cdot \boldsymbol{m}']_t$. As homomorphic operations are applied, the noise associated with ciphertexts grows, limiting the number of operations that can be applied.

B. Hybrid Position-Residue Number System

Using [6] and considering a modulus q such that $q = p^d$, one can represent polynomials $\boldsymbol{a} \in \mathcal{R}_q$ in a positional system of large radix base p – i.e. such that $\boldsymbol{a} = \sum_{i=0}^{d-1} \boldsymbol{a}_i p^i$ with $(\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{d-1}) \in (\mathcal{R}_p)^d$. We view elements representated in HPR as elements of $\mathcal{R}_p[Y]$, leading to:

$$\boldsymbol{a}(Y) = \sum_{i=0}^{d-1} \boldsymbol{a}_i Y^i$$

which produces the value of $\boldsymbol{a}$ when evaluated at $Y = p$.

Moreover, if the modulus p is chosen as a product of small distinct primes, $p = \prod_{p_i \in \mathcal{P}} p_i$ with $\mathcal{P} = \{p_1, \cdots, p_k\}$, the Chinese Remainder Theorem (CRT) states that there is a ring isomorphism between $\mathcal{R}_p$ and $\mathcal{R}_{p_1} \times \ldots \times \mathcal{R}_{p_k}$. In this case, the digits $\boldsymbol{a}_i$s can be represented in RNS as $f_{\mathcal{P}}(\boldsymbol{a}_i) = ([\boldsymbol{a}_i]_{p_1}, \ldots, [\boldsymbol{a}_i]_{p_k})$ which allows to break down the arithmetic over the large coefficients into several smaller channels and thus enhance performance. The RNS representation can be reversed, and $\boldsymbol{a}_i$ recovered, by computing:

$$f_{\mathcal{P}}^{-1}([\boldsymbol{a}_i]_{p_1}, \ldots, [\boldsymbol{a}_i]_{p_k}) = \left[\sum_{i=1}^{k} \left[[\boldsymbol{a}_i]_{p_i} \frac{p_i}{p}\right]_{p_i} \frac{p}{p_i}\right]_p = \sum_{i=1}^{k} \left[\boldsymbol{a}_i \frac{p_i}{p}\right]_{p_i} \frac{p}{p_i} - \boldsymbol{\alpha}_i p \tag{3}$$

with $\|\boldsymbol{\alpha}_i\|_\infty \leq (k-1)/2$. However, since the HPR is of a positional nature, one has to consider the propagation of carries over the digits after additions or multiplications. In order to compute the carries, the digits should be able to grow larger than p. Thus, the digits arithmetic cannot be performed directly modulo p and a second RNS basis $\mathcal{B} = \{b_1, \ldots, b_k\}$ is adjoined to $\mathcal{P}$.

1) Carry Propagation: In order to approximate the carry caused by the ith digit, $\boldsymbol{\nu}_i = \lfloor \frac{\boldsymbol{a}_i}{p} \rceil$, we add a modulus b_{sk} to $\mathcal{B}$ to form $\mathcal{B}_{\text{sk}}$, and use a Shenoy-Kumaresan approach [8]. First, we use a fast base extension (FBE),

$$\texttt{FBE}(\boldsymbol{a}_i, \mathcal{P}, \mathcal{B}_{\text{sk}}) = \left(\sum_{i=1}^{k} \left[\boldsymbol{a}_i \frac{p_i}{p}\right]_{p_i} \cdot \frac{p}{p_i} \bmod b\right)_{b \in \mathcal{B}_{\text{sk}}}$$

to obtain $[\boldsymbol{a}_i]_p + \boldsymbol{\alpha}_i \cdot p$ in base $\mathcal{B}_{\text{sk}}$. Then, the approximated carry $\tilde{\boldsymbol{\nu}}_i = (\boldsymbol{a}_i - \texttt{FBE}(\boldsymbol{a}_i, \mathcal{P}, \mathcal{B}_{\text{sk}}))p^{-1} = \lfloor \frac{\boldsymbol{a}_i}{p} \rceil - \boldsymbol{\alpha}_i$ is produced in $\mathcal{B}_{\text{sk}}$. To finalise the procedure, the value of $\tilde{\boldsymbol{\nu}}_i$ is extended exactly to $\mathcal{P}$. An inexact extension first computes $\texttt{FBE}(\tilde{\boldsymbol{\nu}}_i, \mathcal{B}, \mathcal{P}) = \tilde{\boldsymbol{\nu}}_i + \boldsymbol{\alpha}_i' \cdot b$ in basis $\mathcal{P}$ with $b = b_1 \cdots b_k$ and $\|\boldsymbol{\alpha}_i'\|_\infty \leqslant (k-1)/2$. Then, the extra residue modulo b_{sk} is used to correct $\boldsymbol{\alpha}_i' \cdot b$, by computing $\boldsymbol{\alpha}_i'$ with:

$$[\boldsymbol{\alpha}_i']_{b_{\text{sk}}} = \left[\frac{1}{b}\left(\sum_{i=1}^{\ell} \left[\boldsymbol{a}_i \frac{b_i}{b}\right]_{b_i} \frac{b}{b_i} - [\tilde{\boldsymbol{c}}_i]_{b_{\text{sk}}}\right)\right]_{b_{\text{sk}}}.$$

With this procedure, values $\|\tilde{\boldsymbol{\nu}}_i\| < \lambda B$, for some $\lambda > 0$, can be exactly represented and extended from $\mathcal{B}$ so long as $b_{sk} \geq 2(k + \lambda)$ [4, Lemma 6]. For the parameters considered in this article, b_{sk} can have the same bitwidth as the other moduli in $\mathcal{P}$ and $\mathcal{B}$, which is enough to get the residues of $[\boldsymbol{a}_i]_p + \boldsymbol{\alpha}_i \cdot p$ and of $\tilde{\boldsymbol{\nu}_i} = \boldsymbol{\nu}_i - \boldsymbol{\alpha}_i$ in $\mathcal{P} \cup \mathcal{B}_{\text{sk}}$.

Afterwards, $\tilde{\boldsymbol{\nu}_i}$ can be added to the digit of index $i+1$ and one can re-run the procedure to compute the carry $\tilde{\boldsymbol{\nu}}_{i+1}$ caused by the digit of index $i+1$. If one assumes an element to belong to $\mathcal{R}_q$, the representation of the Most Significant Digit (MSD) in basis $\mathcal{B}$ can be discarded. The MSD is implicitly multiplied by p^{d-1}, so any value larger than $p/2$ in absolute value would lead to representations larger than $q/2 = p \times p^{d-1}/2$, which are redundant.

Computational cost. The computation of a carry for a single HPR digit requires a first fast base extension from $\mathcal{P}$ to the $k+1$ moduli of $\mathcal{B}_{\text{sk}}$ ($nk(k+2)$ EMMs), followed by the product by $1/p$ in base $\mathcal{B}_{\text{sk}}$ ($n(k+1)$ EMMs), and a Shenoy-Kumaresan extension ($n(k^2+3k+1)$ EMMs). Finally, since, when operating modulo q, carries are consecutively computed for all the digits but the last, the carry propagation procedure applied to an element in $\mathcal{R}_p^d$ requires the following amount of EMMs:

$$\begin{cases} \mathcal{C}_{\text{1\_carry}} &= n(2k^2+6k+2) \ \texttt{EMMs} \\ \mathcal{C}_{\text{all\_carries}} &= (d-1)\cdot \mathcal{C}_{\text{1\_carry}} \end{cases}.$$

III. Proposed HPR-based construction

In this section, we propose an efficient HPR variant of the FV scheme. Key generation, encryption and homomorphic addition take place as in the original scheme [2], but with polynomial coefficients represented in HPR.

A. Proposed HPR-based Decryption Algorithm

In this section, an HPR-based method for FV decryption is proposed. We assume that the HPR representation of a ciphertext $(\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$, with:

$$\boldsymbol{c}_i(Y) = \sum_{j=0}^{d-1} \boldsymbol{c}_{i,j} Y^j$$

having digits satisfying $\|\boldsymbol{c}_{i,j}\| \leqslant \beta \cdot p$ for $i = 0,1$ and $0 \leqslant j \leqslant d-2$: β may take the value of $k/2$ if carry propagation has been applied; or might be even slightly larger for cases where the ciphertext represents an homomorphic sum. Lemma 1 proves that, for the considered setting, the value of $\boldsymbol{c}_0(p) + \boldsymbol{c}_1(p)\cdot \boldsymbol{s}$ can be efficiently approximated having only access to the mod-$\mathcal{P}$ representation of the MSD of $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ without causing a large error. This reduces by a factor of d the number of NTTs one needs to compute.

Lemma 1. *For $i = 0,1$, let $\boldsymbol{c}_i$ be such that $\|\boldsymbol{c}_{i,j}\|_\infty \leqslant \beta p$ for $j \in [\![0, d-2]\!]$, then:*

$$[\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s}]_p = \frac{\Delta[\boldsymbol{m}]_t + \boldsymbol{v} + q\boldsymbol{u}}{p^{d-1}} + \boldsymbol{e} + \boldsymbol{\epsilon}\cdot p \quad (4)$$

for some $\boldsymbol{\epsilon} \in \mathcal{R}$, with the error $\boldsymbol{e}$ verifying $\|\boldsymbol{e}\|_\infty \leqslant \frac{p}{p-1}\beta(1+\delta_{\mathcal{R}} B_{key})$.

Proof. In Appendix A. □

Decryption consists of efficiently computing (5) modulo t based on the values of $\boldsymbol{c}_{0,d-1}$ and $\boldsymbol{c}_{1,d-1}$.

$$\left\lfloor t \cdot \frac{[\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s}]_p}{p} \right\rceil = [\boldsymbol{m}]_t + \lfloor \tilde{\boldsymbol{e}} \rceil + t(\boldsymbol{u} + \boldsymbol{\epsilon}) \quad (5)$$

with $\tilde{\boldsymbol{e}} = \left(t\cdot \boldsymbol{v} - |q|_t [\boldsymbol{m}]_t + t p^{d-1}\boldsymbol{e}\right)/q$ which has a norm bounded up by:

$$\|\tilde{\boldsymbol{e}}\|_\infty \leqslant \frac{t}{q}\|\boldsymbol{v}\|_\infty + \frac{t|q|_t}{2q} + \frac{t}{p-1}\beta(1+\delta_{\mathcal{R}} B_{\text{key}}) \quad (6)$$

We use a strategy similar to [4] to compute (5) with a fast basis extension but without any error. By scaling the computation in (5) by an integer γ, any extension errors will be detected since they will be nonzero modulo γ. After efficiently getting $[\gamma t(\boldsymbol{c}_{0,d-1}+\boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s})]_p + \boldsymbol{\alpha} p$ through a fast conversion from $\mathcal{P}$ to $\{\gamma t\}$, one can compute:

$$\begin{aligned}
\boldsymbol{r} &= (\gamma t[\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s}]_p - ([\gamma t(\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s})]_p + \boldsymbol{\alpha} p))/p \\
&= (\gamma t[\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s}]_p - \gamma [t(\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s})]_p)/p - \boldsymbol{\alpha} \\
&\quad + (\gamma [t(\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s})]_p - [\gamma t(\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s})]_p)/p \\
&= \gamma \left\lfloor t \frac{[\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s}]_p}{p} \right\rceil - \boldsymbol{\alpha} + \left\lfloor \gamma \frac{[t(\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s})]_p}{p} \right\rceil \\
&= \gamma\left([\boldsymbol{m}]_t + \lfloor \tilde{\boldsymbol{e}} \rceil\right) + \left\lfloor \gamma \frac{[t(\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s})]_p}{p} \right\rceil - \boldsymbol{\alpha} \\
&= \gamma [\boldsymbol{m}]_t + \gamma \lfloor \tilde{\boldsymbol{e}} \rceil + \left\lfloor \gamma \frac{[p\tilde{\boldsymbol{e}}]_p}{p} \right\rceil - \boldsymbol{\alpha} \bmod \gamma t. \qquad (7)
\end{aligned}$$

Now, through the application of Lemma 2 to (7), $[\boldsymbol{m}]_t$ can be computed as $(\boldsymbol{r} - [\boldsymbol{r}]_\gamma)/\gamma \bmod t$.

Lemma 2. *Let $\gamma > k/2$ and $\|\boldsymbol{\alpha}\|_\infty \leqslant (k-1)/2$. If we have*

$$\|\boldsymbol{v}\|_\infty < \frac{q}{2t}\left(1 - \frac{k}{\gamma} - \frac{2t\beta}{p-1}(1+\delta_{\mathcal{R}} B_{key})\right) - \frac{|q|_t}{2} \quad (8)$$

then $\lfloor \tilde{\boldsymbol{e}} \rceil = 0$, and $\left\lfloor \left\lfloor \gamma \frac{[p\tilde{\boldsymbol{e}}]_p}{p} \right\rceil - \boldsymbol{\alpha} \right\rceil_\gamma = \left\lfloor \gamma \frac{[p\tilde{\boldsymbol{e}}]_p}{p} \right\rceil - \boldsymbol{\alpha}$. Hence, the correction technique from [4] works.

Proof. In Appendix B. □

B. Summarising

Here we summarise the above-described decryption steps. First, $\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s}$ is computed as an approximation to $(\boldsymbol{c}_0(p) + \boldsymbol{c}_1(p)\cdot \boldsymbol{s})/p^{d-1}$ with $k\texttt{NTT} + n\texttt{EMM} + k\texttt{NTT}^{-1}$. Then (7) is computed modulo γt. Notice that since $\gamma t \cdot [\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s}]_p = 0(\text{mod}\gamma t)$, the expression for $\boldsymbol{r}$ can be simplified as:

$$\begin{aligned}
\boldsymbol{r} &= \frac{-\left([\gamma t(\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s})]_p + \boldsymbol{\alpha} p\right)}{p} \bmod \gamma t \\
&= -\texttt{FBE}([\gamma t(\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\cdot \boldsymbol{s})], \mathcal{P}, \gamma t).p^{-1} \bmod \gamma t
\end{aligned}$$

which costs $kn\texttt{EMM} + kn\texttt{EMM}_{\gamma t}$ to compute if γt fits one machine word. Then the message is extracted by using the centred remainder of r modulo γ since $[\boldsymbol{m}]_t = (\boldsymbol{r} - [\boldsymbol{r}]_\gamma)/\gamma \bmod t$ under the conditions of Lemma 2.

Notice that as in [4], if γ is chosen as a power of two, division by γ can be replaced with a shift. Hence, the decryption algorithm has a computational cost of:

$$k \ \texttt{NTT} + 2kn \ \texttt{EMM} + k \ \texttt{NTT}^{-1} + kn \ \texttt{EMM}_{\gamma t}.$$

C. Proposed HPR-based Homomorphic Multiplication

In this section, an HPR-based method for FV homomorphic multiplication is proposed. We consider two ciphertexts $\mathtt{ct} = (\boldsymbol{c}_0(Y), \boldsymbol{c}_1(Y))$ and $\mathtt{ct}' = (\boldsymbol{c}'_0(Y), \boldsymbol{c}'_1(Y))$, with their elements represented in HPR, encrypting respectively $\boldsymbol{m}$ and $\boldsymbol{m}'$, such that $\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} = \Delta[\boldsymbol{m}]_t + \boldsymbol{v} + \boldsymbol{r}q$ and $\boldsymbol{c}'_0 + \boldsymbol{c}'_1 \cdot \boldsymbol{s} = \Delta[\boldsymbol{m}']_t + \boldsymbol{v}' + \boldsymbol{r}'q$ where $\boldsymbol{r}$ and $\boldsymbol{r}'$ satisfy:

$$\|\boldsymbol{r}\|_\infty, \|\boldsymbol{r}'\|_\infty \leqslant \underbrace{\frac{\delta_{\mathcal{R}} B_{key} + 1}{2} + 1}_{r_\infty} . \qquad (9)$$

The polynomial digits are assumed to be bounded in a similar way to how they were bounded during decryption in Section III-A, i.e. for $(i,j) \in \{0,1\} \times \{0,\dots,d-1\}$, $\|\boldsymbol{c}_{i,j}\|_\infty, \|\boldsymbol{c}'_{i,j}\|_\infty \leqslant \beta \cdot p$ with $\beta \simeq k/2$.

Until now, we only needed to represent the digit of index $d-1$ in basis $\mathcal{P}$, since we were dealing with elements of $\mathcal{R}$ modulo q. However, during homomorphic multiplication, the first product in (2) is not reduced modulo q. Therefore we have to extend the digits of index $d-1$ to the basis $\mathcal{P} \times \mathcal{B}_{\text{sk}}$. This extension is done exactly with the approach of [5], denoted HPS in the rest of the paper. First, $\mathtt{FBE}(\boldsymbol{c}_{i,d-1}, \mathcal{P}, \mathcal{B}_{\text{sk}}) = [\boldsymbol{c}_{i,d-1}]_p + \boldsymbol{\alpha}_i \cdot p$ is computed. Then, $\boldsymbol{\alpha}_i$ is produced as:

$$\boldsymbol{\alpha}_i = \left\lfloor \sum_{j=1}^{k} \frac{\left[\boldsymbol{c}_{i,d-1} \frac{p_j}{p}\right]_{p_j}}{p_j} \right\rceil \qquad (10)$$

by using floating point operations to compute the quotients and by rounding to the nearest integer. Finally, the term $\boldsymbol{\alpha}_i \cdot p$ is subtracted from the intermediate result. The lifting procedure of one digit has a practical complexity of $n(k^2+3k+1)$ $\mathtt{EMMs}$ and $n(k+1)$ Floating-Point Operations ($\mathtt{FPOs}$).

Note that the approximation of (10) with floating point arithmetic may lead to an erroneous computation of $\boldsymbol{\alpha}_i$. There is a region $\mathbb{Z} + \frac{1}{2} \pm \epsilon$, where ϵ is related to the used precision and the number of moduli in the basis $\mathcal{P}$ ($|\epsilon| < 2^{-50}$ for IEEE 754 double precision and $k < 8$), to which the value computed in (10) may belong before rounding, causing a possible error of at most 1. While one can detect when this happens, and redo the computation with a higher precision to reduce ϵ, we have chosen, as in [5], to skip this error detection. We will discuss the possible effects of this choice in Remark 1.

At this point we aim at computing (2). To do so we start by computing the $\mathtt{NTTs}$ of all the digits of the two input ciphertexts ($4d(2k+1)$ $\mathtt{NTTs}$ in total). Then the product $(\widetilde{\boldsymbol{c}}_0(Y), \widetilde{\boldsymbol{c}}_1(Y), \widetilde{\boldsymbol{c}}_2(Y))$ is computed with a Karatsuba pattern.

The products of polynomials in Y are performed as usual, i.e. in the power basis, and the products of digits, which are in $\mathtt{NTT}$ form, are made component-wise. Since the product is not reduced modulo q, no reduction modulo Y^d is done. Thus we obtain polynomials of degree $(2d-2)$ in Y. The division by $q = p^d$ and the rounding can be approximated by computing only the $d-1$ MSDs of the products. Nonetheless, in order to reduce the noise added by this approximation, we also keep the carry coming from the digits of index $d-1$.

Lemma 3. *Let $i \in \{0,1,2\}$ and $\lfloor \widetilde{\boldsymbol{c}}_{i,d-1}/p \rceil - \boldsymbol{\alpha}_{i,d-1}$ be the carry coming from $\widetilde{\boldsymbol{c}}_{i,d-1}$ with $\|\boldsymbol{\alpha}_{i,d-1}\|_\infty \leqslant (k-1)/2$ (see Section II-B1). We have:*

$$\left\lfloor \frac{\widetilde{\boldsymbol{c}}_{i,d-1}}{p} \right\rceil - \boldsymbol{\alpha}_{i,d-1} + \widetilde{\boldsymbol{c}}_{i,d} + \dots + \widetilde{\boldsymbol{c}}_{i,2d-2} \cdot p^{d-2} = \frac{\widetilde{\boldsymbol{c}}_i}{q} + \boldsymbol{e}_i \quad (11)$$

$$\text{with: } \|\boldsymbol{e}_i\|_\infty \leqslant \underbrace{6\delta_{\mathcal{R}} t \beta^2 p \left(\frac{d-1}{p-1} - \frac{1-p^{-d}}{(p-1)^2}\right) + \frac{k}{2}}_{B_e} . \quad (12)$$

Proof. In Appendix C. ☐

The three polynomials of degree $d-1$ in Y described in Lemma 3 approximate (2) with an error $\boldsymbol{e}_i$. They may be interpreted as encrypting $[\boldsymbol{m} \cdot \boldsymbol{m}']_t$. The next proposition states the inherent noise of this new ciphertext, depending on the input noises $\boldsymbol{v}$ and $\boldsymbol{v}'$.

Proposition 1. *For $i \in \{0,1,2\}$, let $\hat{\boldsymbol{c}}_i = \left\lfloor \frac{\widetilde{\boldsymbol{c}}_{i,d-1}}{p} \right\rceil - \boldsymbol{\alpha}_{i,d-1} + \sum_{j=d}^{2d-2} \widetilde{\boldsymbol{c}}_{i,j} \cdot p^{j-d}$ as in (11), then we have:*

$$\hat{\boldsymbol{c}}_0 + \hat{\boldsymbol{c}}_1 \cdot \boldsymbol{s} + \hat{\boldsymbol{c}}_2 \cdot \boldsymbol{s}^2 \equiv \Delta \cdot [\boldsymbol{m} \cdot \boldsymbol{m}']_t + \hat{\boldsymbol{v}} \bmod q$$

$$\begin{aligned} \text{with: } \|\hat{\boldsymbol{v}}\|_\infty \leqslant & \delta_{\mathcal{R}} t \left(r_\infty + 1/2\right) \left(\|\boldsymbol{v}\|_\infty + \|\boldsymbol{v}'\|_\infty\right) \\ & + \delta_{\mathcal{R}} \min(\|\boldsymbol{v}\|_\infty, \|\boldsymbol{v}'\|_\infty)/2 + t^2 \delta_{\mathcal{R}} (1 + r_\infty) \\ & + B_e \left(1 + \delta_{\mathcal{R}} B_{key} + \delta_R^2 B_{key}^2\right) + 1/2(t+1) \end{aligned} \qquad (13)$$

and where B_e is as in (12) and r_∞ is as in (9).

Proof. In Appendix D. ☐

Remark 1. *In the case where an error has occurred in the $\mathtt{HPS}$ basis extension, before the multiplication, then we would operate on $\mathbf{c}_i + q\boldsymbol{u}$ with $\|\boldsymbol{u}\|_\infty = 1$ (or $\mathbf{c}'_i + q\boldsymbol{u}'$ with $\|\boldsymbol{u}'\|_\infty = 1$). In this case r_∞ would increase to $3(\delta_{\mathcal{R}} B_{key} + 1)/2 + 1$. As explained in [5, Section 4.5], this would only impact the size of $\delta_{\mathcal{R}} t \left(r_\infty + 1/2\right) \left(\|\boldsymbol{v}\|_\infty + \|\boldsymbol{v}'\|_\infty\right)$ by less than 3% in the worst case. Considering that this error occurs with probability smaller than $4n \cdot 2^{-50} \leq 2^{-32}$ on each multiplication (for $n \leq 2^{16}$), its effect on the noise growth in practice is negligible.*

In practice, after having computed $(\widetilde{\boldsymbol{c}}_0(Y), \widetilde{\boldsymbol{c}}_1(Y), \widetilde{\boldsymbol{c}}_2(Y))$, the carries of $\widetilde{\boldsymbol{c}}_{0,d-1}$ and $\widetilde{\boldsymbol{c}}_{1,d-1}$ are computed and stored apart. These are then added after the relinearisation step to the final output to reduce the overall computational complexity. The computation of the two carries requires $2(2k+1)$ $\mathtt{NTT}^{-1}$ to leave the $\mathtt{NTT}$ representation, and $2 \times \mathcal{C}_{\text{1\_carry}}$. Moreover, the carry generated by $\widetilde{\boldsymbol{c}}_{2,d-1}$ is only computed and propagated during the relinearisation step. Therefore, at this point the three elements ciphertext has the $d-1$ MSDs of $\widetilde{\boldsymbol{c}}_0$ and $\widetilde{\boldsymbol{c}}_1$ and the d MSDs of $\widetilde{\boldsymbol{c}}_2$ in $\mathtt{NTT}$ representation; and the two carries stored apart in coefficient representation.

Computational cost. Overall, assuming that each vector-matrix product is performed with a naive algorithm, the homomorphic multiplication (before relinearisation) requires:

$$\begin{aligned} \mathcal{C}_{\text{mult.}} \;=\; & n(4k^2 + 12k + 4)\mathtt{EMMs} + n(4k+4)\mathtt{FPOs} \\ & + 4d(2k+1)\ \mathtt{NTTs} + 2(2k+1)\ \mathtt{NTTs}^{-1} \\ & + 3nd(2k+1)(d+3)/2\ \mathtt{EMMs} + 2 \cdot \mathcal{C}_{\text{1\_carry}} \end{aligned} .$$

D. Relinearisation

At the beginning of relinearisation, the d MSDs of $\tilde{\boldsymbol{c}}_2$ are converted from the NTT to the canonical domain at a cost of $d(2k+1)$ $\texttt{NTT}^{-1}$. Then, carry propagation is applied and the Least Significant Digit (LSD) is discarded, producing $\hat{\boldsymbol{c}}_2$, as per Proposition 1. By construction, the digits $\boldsymbol{a} = ([\boldsymbol{a}_{0,\mathcal{P}}, \boldsymbol{a}_{0,\mathcal{B}}], \ldots, [\boldsymbol{a}_{d-1,\mathcal{P}}, \boldsymbol{a}_{d-1,\mathcal{B}}])_{\text{HPR}}$ that result from propagating carries satisfy:

$$\left[Y^i f^{-1}_{\mathcal{P}\cup\mathcal{B}}([\boldsymbol{a}_{i,\mathcal{P}}, \boldsymbol{a}_{i,\mathcal{B}}])\right]_q = \left[Y^i \texttt{FBE}(\boldsymbol{a}_{i,\mathcal{P}}, \mathcal{P}, \{q\})\right]_q \quad (14)$$

where $f^{-1}_{\mathcal{P}\cup\mathcal{B}}$ corresponds to reverting the RNS representation with respect to the $(\mathcal{P}, \mathcal{B})$ basis (see (3)). Concretely, this means that once the carry propagation has been done, the residues of $\hat{\boldsymbol{c}}_2$ in base $\mathcal{B}$ are no longer required for the relinearisation process. $[\hat{\boldsymbol{c}}_2 \cdot \boldsymbol{s}^2]_q$ can be rewritten as:

$$[\hat{\boldsymbol{c}}_2 \cdot \boldsymbol{s}^2]_q = \left[\sum_{i=0}^{d-1} \texttt{FBE}(\hat{\boldsymbol{c}}_{2,i}, \mathcal{P}, \{q\}) \cdot \boldsymbol{s}^2 Y^i\right]_q = \left[\sum_{i=0}^{d-1}\sum_{j=1}^{k} \left[\hat{\boldsymbol{c}}_{2,i}\frac{p_j}{p}\right]_{p_j} \frac{p}{p_j} Y^i \boldsymbol{s}^2\right]_q . \quad (15)$$

Thus, we can define $\mathfrak{D}(\hat{\boldsymbol{c}}_2)$ and $\mathfrak{P}(\boldsymbol{s}^2)$ as the vectors with entries $\boldsymbol{r}_{i,j} = \left[\hat{\boldsymbol{c}}_{2,i}\frac{p_j}{p}\right]_{p_j}$ and $\boldsymbol{\psi}_{i,j} = \left[\frac{p}{p_j} Y^i \boldsymbol{s}^2\right]_q$, respectively, for $0 \leqslant i \leqslant d-1$ and $1 \leqslant j \leqslant k$. Clearly, the $\|\boldsymbol{r}_{i,j}\|_\infty < p_j/2$ are small, and $\left[\langle \mathfrak{D}(\hat{\boldsymbol{c}}_2), \mathfrak{P}(\boldsymbol{s}^2)\rangle\right]_q = [\hat{\boldsymbol{c}}_2 \cdot \boldsymbol{s}^2]_q$.

The relinearisation-key ($\overrightarrow{\texttt{rlk}}$) is composed of pseudo-encryptions of the $\boldsymbol{\psi}_{i,j}$ values. To produce the pseudo-encryption with respect to $\boldsymbol{\psi}_{i,j}$, a Ring Learning with Errors (RLWE) sample $([-(\boldsymbol{a}_{i,j} \cdot \boldsymbol{s} + \boldsymbol{e}_{i,j})]_q)$, $\boldsymbol{a}_{i,j})$ is drawn, and $\frac{p}{p_j} \cdot \boldsymbol{s}^2$ is added to the i-th digit of the first element. Then, the whole relinearisation key is stored in NTT form.

Remark 2. *Note that, since $\|\boldsymbol{s}^2\|_\infty \leqslant \delta_{\mathcal{R}} B^2_{key} < p_j$ for moduli p_j of practical size, $\frac{p}{p_j} \cdot \boldsymbol{s}^2$ is representable as a single HPR digit. Thus, the RLWE samples used to mask the $\boldsymbol{\psi}_{i,j} = \frac{p}{p_j} \cdot Y^i \cdot \boldsymbol{s}^2$, for a fixed i and $1 \leq j \leq k$, do not need to have a degree higher than i in Y. The $d-1-i$ MSDs of the k elements of the relinearisation-key associated with these $\psi_{i,j}$ can be set to zero. In practice, RLWE samples are drawn with respect to a modulus $p^i \leqslant p^d$, but with an error of same standard deviation σ_{err}. This technique allows us to not only reduce the size of $\overrightarrow{\texttt{rlk}}$, but also to further reduce the practical complexity of the relinearisation.*

When a small polynomial $\boldsymbol{r}_{i,j}$ has been extracted from a residue modulo p_j, it is easily extended to the base $(\mathcal{P}, \mathcal{B}_{\text{sk}})$ through a simple copy-paste operation. Then, since its representation is a single digit, the computation of its NTT form only requires $(2k+1)$ NTTs. In the next step, the polynomials $\boldsymbol{r}_{i,j}$ are multiplied by the corresponding $\overrightarrow{\texttt{rlk}}$ elements and the results are accumulated in $\hat{\boldsymbol{c}}_0, \hat{\boldsymbol{c}}_1$.

The relinearisation step causes an increase in the noise associated with the ciphertext. More concretely, for a homomorphic multiplication result $(\widehat{\boldsymbol{c}}'_0, \widehat{\boldsymbol{c}}'_1)$, one gets:

$$\begin{aligned}
[\widehat{\boldsymbol{c}}'_0 + \widehat{\boldsymbol{c}}'_1 \boldsymbol{s}]_q &= \big[\hat{\boldsymbol{c}}_0 + \hat{\boldsymbol{c}}_1 \cdot \boldsymbol{s} + \langle \mathfrak{D}(\hat{\boldsymbol{c}}_2), \mathfrak{P}(\boldsymbol{s}^2) - \\
&\qquad (\overrightarrow{\boldsymbol{e}} + \overrightarrow{\boldsymbol{a}} \cdot \boldsymbol{s})\rangle + \langle \mathfrak{D}(\hat{\boldsymbol{c}}_2), \overrightarrow{\boldsymbol{a}}\rangle \cdot \boldsymbol{s}\big]_q \\
&= \left[\hat{\boldsymbol{c}}_0 + \hat{\boldsymbol{c}}_1 \cdot \boldsymbol{s} + \hat{\boldsymbol{c}}_2 \cdot \boldsymbol{s}^2 - \langle \mathfrak{D}(\hat{\boldsymbol{c}}_2), \overrightarrow{\boldsymbol{e}}\rangle\right]_q \\
&= \left[\hat{\boldsymbol{c}}_0 + \hat{\boldsymbol{c}}_1 \cdot \boldsymbol{s} + \hat{\boldsymbol{c}}_2 \cdot \boldsymbol{s}^2 + \boldsymbol{e}_{\text{relin}}\right]_q .
\end{aligned}$$

The extra noise caused by this step is bounded by:

$$\|\boldsymbol{e}_{\text{relin}}\|_\infty \leqslant \delta_{\mathcal{R}} \cdot d \cdot B_{err} \cdot (p_1 + \cdots + p_k). \quad (16)$$

Computational cost. After the carry propagation of the d MSDs of $\tilde{\boldsymbol{c}}_2$, we have to compute the NTTs of 1-digit polynomials $\boldsymbol{r}_{i,j}$ in base $(\mathcal{P}, \mathcal{B}_{\text{sk}})$ for a total of $dk(2k+1)$ NTTs. Then, for each j, the single-digit polynomials $\boldsymbol{r}_{i,j}$ are multiplied by two relinearisation-key elements of degree i, resulting in a total of $nkd(d+1)(2k+1)$ elementary products. The results of these products are added to $\widehat{\boldsymbol{c}}_0$ and $\widehat{\boldsymbol{c}}_1$. Once all intermediate products have been accumulated, $2d(2k+1)$ $\texttt{NTT}^{-1}$s are used to obtain the result in the canonical domain. Finally, carry propagation is applied to $\widehat{\boldsymbol{c}}'_0$ and $\widehat{\boldsymbol{c}}'_1$. Thus, the total cost of the relinearisation step is:

$$\begin{aligned}
\mathcal{C}_{\text{relin.}} \;=\; & dk(2k+1)\ \texttt{NTTs} + 3d(2k+1)\ \texttt{NTTs}^{-1} \\
& + nkd((d+1)(2k+1)+1)\ \texttt{EMMs} \\
& + (3d-2)\mathcal{C}_{1\_\text{carry}}
\end{aligned} .$$

Remark 3. *The bit size of the noise grows linearly with the multiplicative depth L. Consequently, when the depth L reaches around $L_{\max}/d$, the noise affects more than one digit. At this point one can apply a coarse grain decomposition to decrease the practical complexity of the relinearisation. The sum with index j in* (15) *can be computed by the decomposition function prior to the product with the relinearisation key. With this technique, the decomposition polynomials become $\boldsymbol{r}_i = \sum_{j=1}^{k}\left[\hat{\boldsymbol{c}}_{2,i}\frac{p_j}{p}\right]_{p_j}\frac{p}{p_j}$, and the new relinearisation key contains $\boldsymbol{\psi}_i = \left[Y^i \boldsymbol{s}^2\right]_q$. Since $\boldsymbol{r}_i$ is no longer a single residue, its conversion toward base $\mathcal{B}$ is not a simple copy-paste. However, the carry propagation performed prior to the relinearisation already involves such a base conversion. So, $\boldsymbol{r}_i$ is obtained in the full RNS base at no extra cost. Moreover, one may disregard an increasingly larger number of LSDs during relinearisation as noise grows. Assuming that only the d' MSDs are taken into account for $\hat{\boldsymbol{c}}_{2,i}$ (in the case where $L \simeq (d-d')\frac{L_{\max}}{d}$), the cost boils down to:*

$$\begin{aligned}
\mathcal{C}_{relin.,d'} \;=\; & d'(2k+1)\ \mathit{NTTs} + 3d'(2k+1)\ \mathit{NTTs}^{-1} \\
& + nd'(d'+1)(2k+1)\ \mathit{EMMs} \\
& + (3d'-2)\mathcal{C}_{1\_carry}
\end{aligned} .$$

IV. Related Art

A first adaptation of FV to the RNS was achieved in [4]. Due to the reduced complexity, speedups from 2 up to 4 and from 5 up to 20 were achieved for the homomorphic multiplication and decryption operations, respectively, in comparison to an implementation based on a generic multiprecision library. Later, [5] proposed modifications to [4], namely by computing part of the RNS basis extensions with floating point arithmetic.

It has been noticed in [9] that for similar cryptographic parameters, [4] offers a lower multiplicative depth than [5]. However as pointed out in [10], this observation results from a misimplementation of the techniques of [4] in the `SEAL` and `PALISADE` libraries. Thus, since there is no clear advantage of using one version over the other, herein, we make use of techniques from both [4] and [5].

V. Complexity Analysis

The complexities of our HPR-based variant of the FV scheme and of [4], [5] are presented in Table I which, for simplicity, associates both forward and inverse NTTs with the NTTs column. K_{RNS} corresponds to the size of the RNS bases of [4], [5]. Both [4], [5] make use of secondary bases of size $K_{RNS}+1$. In HPR, each polynomial coefficient is associated with d digits, each of them represented in an RNS basis of size k, and a secondary RNS basis with $k+1$ moduli. If both systems target the same security level and use moduli of similar bitwidth, the value of k will be about d times smaller than K_{RNS}, since $q_1 \dots q_{K_{RNS}} \approx (p_1 \dots p_k)^d$.

A. Decryption

In all schemes, the computational cost is dominated by the amount of NTTs. A radix-2 NTT requires $\frac{n}{2}\log_2 n$ `EMMs` [11]. In practice, [4], [5] require K_{RNS} forward NTTs and K_{RNS} inverse NTTs; whereas k forward NTTs and k inverse NTTs are required for the HPR-based decryption. As a consequence, the cost of decryption in HPR is nearly d times smaller than that of [4], [5]. For security reasons, K_{RNS} and $K = d\times k$ are in $\mathcal{O}(n)$. Hence, since NTTs have a quasi-linear complexity, the asymptotic complexity of the decryption is reduced from:

$$\mathcal{C}(\texttt{Dec}_{\text{RNS}}) \in \mathcal{O}(n^2 \log n) \text{ to } \mathcal{C}(\texttt{Dec}_{\text{HPR}}) \in \mathcal{O}(n^{1+\varepsilon}\log n)$$

when $k = K^{\varepsilon}$, with $0 \leqslant \varepsilon \leqslant 1$ ($\varepsilon = 1$ being the RNS variant, and $\varepsilon = 0$ the radix-p_1 positional variant). As a consequence, when the number of digits d increases, k and ε decrease, which results in a more efficient decryption algorithm. In particular for $d = K$, and thus $k = 1$, we obtain a quasi-linear complexity.

B. Homomorphic Multiplication

Using the same method than previously, since K_{RNS} and $K \in \mathcal{O}(n)$, the asymptotic complexity of [4], [5] is:

$$\mathcal{C}(\texttt{Mult}_{\text{RNS}}) \in \mathcal{O}(n^3 \log n)$$

while for our HPR variant it is reduced to:

$$\mathcal{C}(\texttt{Mult}_{\text{HPR}}) \in \mathcal{O}(n^{2+\varepsilon}\log n + n^3) = \mathcal{O}(n^3).$$

Since asymptotically the choice of ε does not impact the complexity of homomorphic multiplication, it would seem best to take $\varepsilon = 0$ so as to minimise the cost of the decryption procedure. However, homomorphic multiplication is executed more often and is the practical bottleneck of these schemes. Hence, one should rather minimise its computational cost by considering the optimal k_n, or equivalently d_n, in dimension n, that minimises the costs presented in Table I.

Remark 4. *When Remark 3 is considered, the asymptotic complexity of relinearisation can be reduced to*

$$\mathcal{O}(dkn\log(n) + nd^2k) = \mathcal{O}(n^2\log(n) + n^{3-\varepsilon}) = \mathcal{O}(n^{3-\varepsilon}).$$

VI. Experimental Evaluation

The NTT-based Fast Lattice (NFL) library [7] was adapted to handle moduli with the bitwidths featured in Table II. Our HPR-based variant and [4], [5] were implemented in C++, using the modified NFL library [7][1]. They were compiled with gcc 8.3.0 and executed on a single thread on an Intel Core i7-6700K processor running at 4.0GHz with 32GB of main memory, operated by CentOS 7.3. Experimental results for the proposed HPR-based scheme and [4], [5], for the cryptographic parameters presented in Table II, are depicted in Table III.

Asymptotically, decryption has a speed-up that is linear with n. This behaviour is more noticeable for $n \geq 2^{14}$ in Table III because therein $k = 1$ is featured for $n = 2^{13}$ and $k = 3$ for $n \geq 2^{14}$, since k was chosen to minimise the complexity of homomorphic multiplication. Maximum speed-ups of 8.6 and 8.0 are achieved when the proposed HPR-based scheme is compared with [4] and [5], respectively, for $n = 2^{16}$.

Table III suggests that the theoretically predicted $\mathcal{O}(\log n)$ speed-up for homomorphic multiplication is only achieved in practice for large parameters. While the novel HPR-based representation brings performance improvements to the multiplication part of the procedure (i.e. without relinearisation), that are noticeable for most parameters, the complexity improvements pertaining to relinearisation only take effect for $n \geq 2^{16}$. This suggests that the proposed HPR-based scheme is preferable to [4], [5] for applications relying on sums of products, wherein relinearisation is only applied once, after computing the homomorphic sum of the optimised products, or when large parameters ($n \geq 2^{16}$) are considered. Maximum speed-ups regarding the whole procedure of 2.0 and 1.6 are achieved when the proposed scheme is compared with [5] and [4], respectively.

Moreover, we have measured the depth reached in practice for each scheme by squaring several ciphertexts until they did not decrypt correctly and keeping the smallest value as the practical multiplicative depth. We have not observed any significant difference between the HPR and the RNS variants, which confirms the conclusions of [10].

Conclusion

Whereas state-of-the-art implementations of FV make use of the RNS, the FV scheme requires operations that are inefficient to implement in this representation. In particular, division and rounding require large basis extensions, and the representation of small values is ample. Herein, an alternative approach to accelerate FV was proposed, supported on HPR. Divisions are approximated by considering only the MSDs of HPR values, reducing the asymptotic complexity of decryption from

[1]The code is available at https://gitlab.com/fvhpr/hare

Scheme	DECRYPTION			MULTIPLICATION & RELINEARISATION		
	NTTs	EMMs	FPOs	NTTs	EMMs	FPOs
[4]	$2dk$	$3ndk$	$-$	$(dk)^2 + 16dk + 7$	$n(12(dk)^2 + 35dk + 17)$	-
[5]	$2dk$	ndk	$n(1 + dk)$	$(dk)^2 + 16dk + 7$	$n(12(dk)^2 + 35dk + 14)$	$n(10dk + 13)$
HPR	$2k$	$3nk$	$-$	$2dk^2 + (23d + 4)k + 7d + 2$	$n((2d^2 + 8d + 4)k^2 + (5d^2 + 28d + 8)k + 2d^2 + 10d)$	$n(4k + 4)$

Table I: Computational costs of decryption and homomorphic multiplication for the proposed HPR-based scheme, [4] and [5].

Par.	n	$\log_2 q$	K_{RNS}	$\log_2 q_i$ (RNS)	$K = k \times d$	$\log_2 p_i$ (HPR)
A	2^{13}	275	5	55	$5 = 1 \times 5$	55
B	2^{14}	549	9	61	$9 = 3 \times 3$	61
C	2^{15}	1098	18	61	$18 = 3 \times 6$	61
D	2^{16}	2196	36	61	$36 = 3 \times 12$	61

Table II: Parameters achieving at least 100 bits of security according to the estimator of [12] with $\sigma_{err} = 8.0$. Values of k and d were computed by minimizing the cost of homomorphic multiplication presented in Table I, assuming that 1 NTT = $n \log_2(n)$ EMMs and 1 FPO = 1 EMM.

Par.	Scheme	Dec.	Mult.	Rel.	Mult. + Relin
A	HPR	**0.75**	32.0	24.7	56.7
A	[4]	2.64	**28.3**	13.3	**41.6**
A	[5]	2.36	36.8	**13.2**	50.0
B	HPR	**6.46**	**132**	135	267
B	[4]	12.8	157	97	**254**
B	[5]	12.1	199	**95**	294
C	HPR	**14.7**	**705**	892	**1,597**
C	[4]	64.5	840	**775**	1,615
C	[5]	58.5	1,164	784	1,948
D	HPR	**33.2**	**3,925**	**5,688**	**9,613**
D	[4]	285	8,227	7,118	15,345
D	[5]	267	12,298	7,018	19,316

Table III: Execution time (ms) of decryption and homomorphic multiplication for the HPR-based variant, [4] and [5].

$\mathcal{O}(n^2 \log n)$ to $\mathcal{O}(n \log n)$. Since the homomorphic multiplication operation also depends on division and rounding, it also benefits from the use of HPR. Moreover, since in HPR small values can be represented as a single digit, the amount of NTTs one needs to compute during relinearisation is significantly reduced. As NTTs no longer dominate the asymptotic complexity of homomorphic multiplication, its complexity is reduced from $\mathcal{O}(n^3 \log n)$ to $\mathcal{O}(n^3)$, when compared to RNS-based approaches. Experimental speedups of up to 8.0 and 8.6 for the decryption operation and of up to 2.0 and 1.6 for the homomorphic multiplication are achieved when compared with two state-of-the-art schemes. The experimental results also confirm the asymptotic gain since the speed-ups increase as n grows.

Acknowledgements

This work was partially supported by Portuguese funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 and 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 (CAMPUS-FRANCE); by EU's Horizon 2020 research and innovation programme under grant agreement No. 779391 (FutureTPM). This work was also partially supported by CyberSecurity Research Flanders with reference number VR20192203, by the Office of the Director of National Intelligence (ODNI) and by Intelligence Advanced Research Projects Activity (IARPA) via Contract No. 2019-1902070006. J. Eynard was initially supported by University of Toulon (France) for this project.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ODNI or IARPA.

The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

[1] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford, CA, USA, 2009, aAI3382729.

[2] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," Cryptology ePrint Archive, Report 2012/144, 2012, http://eprint.iacr.org/.

[3] L. Sousa, S. Antao, and P. Martins, "Combining residue arithmetic to design efficient cryptographic circuits and systems," *IEEE Circuits and Systems Magazine*, vol. 16, no. 4, pp. 6–32, Fourthquarter 2016.

[4] J.-C. Bajard, J. Eynard, M. A. Hasan, and V. Zucca, "A Full RNS Variant of FV Like Somewhat Homomorphic Encryption Schemes," in *Selected Areas in Cryptography – SAC 2016*, R. Avanzi and H. Heys, Eds. Cham: Springer International Publishing, 2017, pp. 423–442.

[5] S. Halevi, Y. Polyakov, and V. Shoup, "An Improved RNS Variant of the BFV Homomorphic Encryption Scheme," in *Topics in Cryptology – CT-RSA 2019*, M. Matsui, Ed. Cham: Springer International Publishing, 2019, pp. 83–105.

[6] K. Bigou and A. Tisserand, "Hybrid Position-Residues Number System," in *ARITH: 23rd Symposium on Computer Arithmetic*, J. Hormigo, S. Oberman, and N. Revol, Eds. Santa Clara, CA, United States: IEEE, Jul. 2016. [Online]. Available: https://hal.inria.fr/hal-01314232

[7] C. Aguilar-Melchor, J. Barrier, S. Guelton, A. Guinet, M.-O. Killijian, and T. 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*. Cham: Springer International Publishing, 2016, ch. NFLlib: NTT-Based Fast Lattice Library, pp. 341–356. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-29485-8_20

[8] A. P. Shenoy and R. Kumaresan, "Fast Base Extension Using a Redundant Modulus in RNS," *IEEE Trans. Comput.*, vol. 38, no. 2, pp. 292–297, Feb. 1989. [Online]. Available: http://dx.doi.org/10.1109/12.16508

[9] A. Qaisar Ahmad Al Badawi, Y. Polyakov, K. M. M. Aung, B. Veeravalli, and K. Rohloff, "Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption Scheme," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2019.

[10] J.-C. Bajard, J. Eynard, P. Martins, L. Sousa, and V. Zucca, "Note on the noise growth of the RNS variants of the BFV scheme," Cryptology ePrint Archive, Report 2019/1266, 2019, https://eprint.iacr.org/2019/1266.

[11] T. Pöppelmann and T. Güneysu, "Towards Efficient Arithmetic for Lattice-Based Cryptography on Reconfigurable Hardware," in *Progress in Cryptology – LATINCRYPT 2012*, A. Hevia and G. Neven, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 139–158.

[12] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of Learning with Errors." *Journal of Mathematical Cryptology*, vol. 9, pp. 169–203, October 2015.

[13] J. Bos, K. Lauter, J. Loftus, and M. Naehrig, “Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme,” in *Cryptography and Coding*, ser. Lecture Notes in Computer Science, M. Stam, Ed. Springer Berlin Heidelberg, 2013, vol. 8308, pp. 45–64. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-45239-0_4

APPENDIX A
PROOF OF LEMMA 1

A valid encryption of $[\boldsymbol{m}]_t$ satisfies $\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} = \Delta[\boldsymbol{m}]_t + \boldsymbol{v} + \boldsymbol{u}q$. Thus we have that the approximation in (4) satisfies:

$[\boldsymbol{c}_{0,d-1} + \boldsymbol{c}_{1,d-1}\boldsymbol{s}]_p = \frac{\Delta[\boldsymbol{m}]_t+\boldsymbol{v}+q\boldsymbol{u}}{p^{d-1}} - \sum_{i=0}^{d-2}(\boldsymbol{c}_{0,i} + \boldsymbol{c}_{1,i} \cdot \boldsymbol{s})p^{i-d+1} + \boldsymbol{\epsilon}p$

with $\boldsymbol{e} = -\sum_{i=0}^{d-2}(\boldsymbol{c}_{0,i} + \boldsymbol{c}_{1,i} \cdot \boldsymbol{s})p^{i-d+1}$. The norm of $\boldsymbol{e}$ can thus be bounded as

$\|\boldsymbol{e}\|_\infty \leq \sum_{i=0}^{d-2}(\beta p + \delta_{\mathcal{R}}\beta p B_{\text{key}})p^{i-d+1} \leq (\beta p + \delta_{\mathcal{R}}\beta p B_{\text{key}})\frac{p^{d-1}-1}{p-1}\frac{1}{p^{d-1}} \leq \frac{p}{p-1}\beta(1 + \delta_{\mathcal{R}}B_{\text{key}})$. □

APPENDIX B
PROOF OF LEMMA 2

First, we show that if

$$\|\tilde{\boldsymbol{e}}\|_\infty < \frac{1}{2}\left(1 - \frac{k}{\gamma}\right) \tag{17}$$

is satisfied then $\lfloor\tilde{\boldsymbol{e}}\rceil = 0$ and $\left\|\left\lfloor\gamma\left[p\tilde{\boldsymbol{e}}\right]_p/p\right\rceil - \boldsymbol{\alpha}\right\|_\infty < \gamma/2$. In particular, $\lfloor\tilde{\boldsymbol{e}}\rceil = 0$ since $k, \gamma > 0$ and hence $\|\tilde{\boldsymbol{e}}\|_\infty < 1/2$. In addition, $[p\tilde{\boldsymbol{e}}]_p = p\tilde{\boldsymbol{e}}$ since $\|p\tilde{\boldsymbol{e}}\|_\infty < p/2$. Thus,

$\left\|\left\lfloor\gamma\left[p\tilde{\boldsymbol{e}}\right]_p/p\right\rceil - \boldsymbol{\alpha}\right\|_\infty = \left\|\frac{\gamma p\tilde{\boldsymbol{e}}-[\gamma p\tilde{\boldsymbol{e}}]_p}{p} - \boldsymbol{\alpha}\right\|_\infty < \frac{\gamma p(1-k/\gamma)+p}{2p} + \frac{k-1}{2} \leq \gamma/2$.

Finally, it will be shown that (8) implies (17). This is achieved by exploiting (6) to bound $\|\tilde{\boldsymbol{e}}\|_\infty$, and as consequence $\|\boldsymbol{v}\|_\infty$:

$$\|\tilde{\boldsymbol{e}}\|_\infty \leqslant \frac{t}{q}\|\boldsymbol{v}\|_\infty + \frac{t|q|_t}{2q} + \frac{t}{p-1}\beta(1+\delta_{\mathcal{R}}B_{\text{key}}) < \frac{1}{2}\left(1 - \frac{k}{\gamma}\right).$$

Through a manipulation of the previous expression, (8) is finally reached:

$\frac{t}{q}\|\boldsymbol{v}\|_\infty + \frac{t|q|_t}{2q} + \frac{t}{p-1}\beta(1+\delta_{\mathcal{R}}B_{\text{key}}) < \frac{1}{2}\left(1 - \frac{k}{\gamma}\right) \Leftrightarrow \|\boldsymbol{v}\|_\infty < \frac{q}{2t}\left(1 - \frac{k}{\gamma} - \frac{2t\beta}{p-1}(1 + \delta_{\mathcal{R}}B_{key})\right) - \frac{|q|_t}{2}$. □

APPENDIX C
PROOF OF LEMMA 3

For $i \in \{0, 1, 2\}$ we have by definition: $\widetilde{\boldsymbol{c}}_i = \sum_{j=0}^{2d-2}\widetilde{\boldsymbol{c}}_{i,j} \cdot p^j$. Moreover, given that the digits of $\boldsymbol{c}_l$ and $\boldsymbol{c}'_l$, for $l = 0, 1$, have their norm smaller than βp then:

$$\begin{cases} \text{for } i \in \{0, 2\} \text{ and for } j \in \{0, \ldots, d-2\}, \\ \qquad\qquad \|\widetilde{\boldsymbol{c}}_{i,j}\|_\infty \leqslant \delta_{\mathcal{R}}t(j+1)\beta^2p^2; \\ \text{for } j \in \{0, \ldots, d-2\}, \|\widetilde{\boldsymbol{c}}_{1,j}\|_\infty \leqslant 6\delta_{\mathcal{R}}t(j+1)\beta^2p^2. \end{cases} \tag{18}$$

Now by using the second bound for every $\widetilde{\boldsymbol{c}}_{i,j}$ we can write:

$\|\widetilde{\boldsymbol{c}}_{i,0} + \cdots + \widetilde{\boldsymbol{c}}_{i,d-2} \cdot p^{d-2} + ([\widetilde{\boldsymbol{c}}_{i,d-1}]_p + p\boldsymbol{\alpha}_{i,d-1}) \cdot p^{d-1}\|_\infty$
$\leqslant 6\delta_{\mathcal{R}}t\beta^2p^2(1 + 2p + \cdots + (d-1)p^{d-2}) + \left(\frac{1}{2} + \frac{k-1}{2}\right)p^d$
$\leqslant 6\delta_{\mathcal{R}}t\beta^2p^2\left(\frac{(d-1)p^{d-1}}{p-1} - \frac{p^d-1}{p(p-1)^2}\right) + \frac{k}{2}p^d$.

Recalling that $p^d = q$, we have:

$\left(\left\lfloor\frac{\widetilde{\boldsymbol{c}}_{i,d-1}}{p}\right\rceil - \boldsymbol{\alpha}_{i,d-1} + \widetilde{\boldsymbol{c}}_{i,d} + \widetilde{\boldsymbol{c}}_{i,d+1} \cdot p + \cdots + \widetilde{\boldsymbol{c}}_{i,2d-2} \cdot p^{d-2}\right)$
$= \frac{\widetilde{\boldsymbol{c}}_i}{q} - \frac{\widetilde{\boldsymbol{c}}_{i,0}+\cdots+\widetilde{\boldsymbol{c}}_{i,d-2}\cdot p^{d-2}}{p^d} - \frac{[\widetilde{\boldsymbol{c}}_{i,d-1}]_p\cdot p^{d-1}+\boldsymbol{\alpha}_{i,d-1}\cdot p^d}{p^d}$

and then we deduce (11) with:

$\boldsymbol{e}_i = -\frac{\widetilde{\boldsymbol{c}}_{0,0}+\cdots+\widetilde{\boldsymbol{c}}_{0,d-2}\cdot p^{d-2}}{p^d} - \frac{[\widetilde{\boldsymbol{c}}_{0,d-1}]_p\cdot p^{d-1}+\boldsymbol{\alpha}_{0,d-1}\cdot p^d}{p^d}$.

In particular, we obtain (12): $\|\boldsymbol{e}_i\|_\infty \leqslant 6\delta_{\mathcal{R}}t\beta^2p\left(\frac{d-1}{p-1} - \frac{1-p^{-d}}{(p-1)^2}\right) + \frac{k}{2}$. □

APPENDIX D
PROOF OF PROPOSITION 1

Knowing that $\Delta t = q - |q|_t$, $[\boldsymbol{m}]_t \cdot [\boldsymbol{m}']_t = [\boldsymbol{m}\cdot\boldsymbol{m}']_t + t\boldsymbol{r}_m$ and $\boldsymbol{v}\cdot\boldsymbol{v}' = \Delta\boldsymbol{r}_v + [\boldsymbol{v}\cdot\boldsymbol{v}']_\Delta$, we can write:

$\frac{t}{q}(\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}) \cdot (\boldsymbol{c}'_0 + \boldsymbol{c}'_1 \cdot \boldsymbol{s}) = \frac{t}{q}(\Delta \cdot [\boldsymbol{m}]_t + \boldsymbol{v} + \boldsymbol{r}q) \cdot (\Delta \cdot [\boldsymbol{m}']_t + \boldsymbol{v}' + \boldsymbol{r}'q) = \frac{t}{q}\left(\Delta^2[\boldsymbol{m}]_t \cdot [\boldsymbol{m}']_t\right.$
$+\Delta([\boldsymbol{m}]_t \cdot \boldsymbol{v}' + [\boldsymbol{m}']_t \cdot \boldsymbol{v}) + \Delta q([\boldsymbol{m}]_t \cdot \boldsymbol{r}' + [\boldsymbol{m}']_t \cdot \boldsymbol{r}) + \boldsymbol{v}\cdot\boldsymbol{v}' + q(\boldsymbol{v}\cdot\boldsymbol{r}' + \boldsymbol{v}'\cdot\boldsymbol{r}) + \left.q^2\boldsymbol{r}\cdot\boldsymbol{r}'\right)$
$= \frac{q-|q|_t}{q}\left(\Delta\left([\boldsymbol{m}\cdot\boldsymbol{m}']_t + t\boldsymbol{r}_m\right) + [\boldsymbol{m}]_t\cdot\boldsymbol{v}' + [\boldsymbol{m}']_t\cdot\boldsymbol{v} + \boldsymbol{r}_v\right)$
$+t(\boldsymbol{v}\cdot\boldsymbol{r}' + \boldsymbol{v}'\cdot\boldsymbol{r}) + (q-|q|_t)([\boldsymbol{m}]_t\cdot\boldsymbol{r}' + [\boldsymbol{m}']_t\cdot\boldsymbol{r}) + \frac{t}{q}[\boldsymbol{v}\cdot\boldsymbol{v}']_\Delta + tq\boldsymbol{r}\cdot\boldsymbol{r}' = \Delta[\boldsymbol{m}\cdot\boldsymbol{m}']_t + [\boldsymbol{m}]_t\cdot\boldsymbol{v}' + [\boldsymbol{m}']_t\cdot\boldsymbol{v} + \boldsymbol{r}_v + t(\boldsymbol{v}\cdot\boldsymbol{r}' + \boldsymbol{v}'\cdot\boldsymbol{r})$
$+\frac{t}{q}[\boldsymbol{v}\cdot\boldsymbol{v}']_\Delta - \frac{|q|_t}{q}\left(\Delta[\boldsymbol{m}\cdot\boldsymbol{m}']_t + [\boldsymbol{m}]_t\cdot\boldsymbol{v}' + [\boldsymbol{m}']_t\cdot\boldsymbol{v} + \boldsymbol{r}_v\right)'$
$+tq\boldsymbol{r}\cdot\boldsymbol{r} + (q-|q|_t)([\boldsymbol{m}]_t\cdot\boldsymbol{r}' + [\boldsymbol{m}']_t\cdot\boldsymbol{r}) + q\boldsymbol{r}_m - 2|q|_t\boldsymbol{r}_m$
$+\frac{|q|_t^2}{q}\boldsymbol{r}_m = \Delta[\boldsymbol{m}\cdot\boldsymbol{m}']_t + \boldsymbol{v}_{\text{mult}} + q([\boldsymbol{m}]_t\cdot\boldsymbol{r}' + [\boldsymbol{m}']_t\cdot\boldsymbol{r} + \boldsymbol{r}_m + t\boldsymbol{r}\cdot\boldsymbol{r}')$,

with: $\boldsymbol{v}_{\text{mult}} = ([\boldsymbol{m}]_t\cdot\boldsymbol{v}' + [\boldsymbol{m}']_t\cdot\boldsymbol{v} + \boldsymbol{r}_v)\left(1 - \frac{|q|_t}{q}\right)$
$+t(\boldsymbol{v}\cdot\boldsymbol{r}' + \boldsymbol{v}'\cdot\boldsymbol{r}) + \frac{t}{q}[\boldsymbol{v}\cdot\boldsymbol{v}']_\Delta - \frac{|q|_t}{q}\Delta[\boldsymbol{m}\cdot\boldsymbol{m}']_t$
$-|q|_t\left([\boldsymbol{m}]_t\cdot\boldsymbol{r}' + [\boldsymbol{m}']_t\cdot\boldsymbol{r}\right) - \boldsymbol{r}_m|q|_t\left(2 - \frac{|q|_t}{q}\right)$

As shown in [13], we have: $\|\boldsymbol{r}_m\|_\infty \leqslant \frac{\delta_{\mathcal{R}}t}{2}$ and $\|\boldsymbol{r}_v\|_\infty \leqslant \frac{\delta_{\mathcal{R}}}{2}\min(\|\boldsymbol{v}\|_\infty, \|\boldsymbol{v}'\|_\infty)$. Therefore:

$\|\boldsymbol{v}_{\text{mult}}\|_\infty$
$\leqslant \frac{t}{2}\delta_{\mathcal{R}}(\|\boldsymbol{v}'\|_\infty + \|\boldsymbol{v}\|_\infty) + \frac{\delta_{\mathcal{R}}}{2}\min(\|\boldsymbol{v}\|_\infty, \|\boldsymbol{v}'\|_\infty)$
$+t\delta_{\mathcal{R}}r_\infty(\|\boldsymbol{v}'\|_\infty + \|\boldsymbol{v}\|_\infty) + \frac{t}{q}\frac{\Delta}{2} + \frac{t^2}{2q}\Delta + t^2\delta_{\mathcal{R}}r_\infty + t^2\delta_{\mathcal{R}}$
$\leqslant \delta_{\mathcal{R}}t\left(r_\infty + \frac{1}{2}\right)(\|\boldsymbol{v}\|_\infty + \|\boldsymbol{v}'\|_\infty)$
$+\frac{\delta_{\mathcal{R}}}{2}\min(\|\boldsymbol{v}\|_\infty, \|\boldsymbol{v}'\|_\infty) + \frac{1}{2} + t\left(\frac{1}{2} + t\delta_{\mathcal{R}}(1 + r_\infty)\right)$.

However in the conditions herein considered, we need also to take into account the errors $\boldsymbol{e}_i$ coming from (11), thus:

$$\begin{aligned} & \hat{\boldsymbol{c}}_0 + \hat{\boldsymbol{c}}_1 \cdot \boldsymbol{s} + \hat{\boldsymbol{c}}_2 \cdot \boldsymbol{s}^2 \\ = \;& \frac{t}{q}\boldsymbol{c}_0 \cdot \boldsymbol{c}'_0 + \boldsymbol{e}_0 + \left(\frac{t}{q}(\boldsymbol{c}_0 \cdot \boldsymbol{c}'_1 + \boldsymbol{c}_1 \cdot \boldsymbol{c}_0) + \boldsymbol{e}_1\right) \cdot \boldsymbol{s} \\ & + \left(\frac{t}{q}\boldsymbol{c}_1 \cdot \boldsymbol{c}'_1 + \boldsymbol{e}_2\right) \cdot \boldsymbol{s}^2 \\ = \;& \frac{t}{q}\left(\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}\right) \cdot \left(\boldsymbol{c}'_0 + \boldsymbol{c}'_1 \cdot \boldsymbol{s}\right) + \boldsymbol{e}_0 + \boldsymbol{e}_1 \cdot \boldsymbol{s} + \boldsymbol{e}_2 \cdot \boldsymbol{s}^2. \end{aligned}$$

Therefore, $\hat{\boldsymbol{v}} = \boldsymbol{v}_{\text{mult}} + \boldsymbol{e}_0 + \boldsymbol{e}_1 \cdot \boldsymbol{s} + \boldsymbol{e}_2 \cdot \boldsymbol{s}^2$, with $\|\boldsymbol{e}_0 + \boldsymbol{e}_1 \cdot \boldsymbol{s} + \boldsymbol{e}_2 \cdot \boldsymbol{s}^2\|_\infty \leqslant B_e\left(1 + \delta_{\mathcal{R}}B_{key} + \delta_{\mathcal{R}}^2B_{key}^2\right)$. We obtain (13) by summing the last two bounds. □