An Analysis of FV Parameters Impact Towards its Hardware Acceleration

Joël Cathébras, Alexandre Carbon, Renaud Sirdey, and Nicolas Ventroux

CEA, LIST, F-91191 Gif-sur-Yvette, France.
firstname.lastname@cea.fr

Abstract. The development of cloud computing services is restrained by privacy concerns. Centralized medical services for instance, require a guarantee of confidentiality when using outsourced computation platforms. Fully Homomorphic Encryption is an intuitive solution to address such issue, but until 2009, existing schemes were only able to evaluate a reduced number of operations (Partially Homomorphic Encryption). In 2009, C. Gentry proposed a blueprint to construct FHE schemes from SHE schemes. However, it was not practical due to the huge data size overhead and the exponential noise growth of the initial SHE. Since then, major improvements have been made over SHE schemes and their noise management, and resulting schemes, like BGV and FV, allow to foresee small applications.
Besides scheme improvements, new practical approaches were proposed to bring homomorphic encryption closer to practice. The IV-based stream cipher trans-ciphering approach brought by Canteaut et al. in 2015 reduces the on-line latency of the trans-ciphering process to a simple homomorphic addition. The homomorphic evaluation of stream ciphers, that produces the trans-ciphering keystream, could be computed in an off-line phase, resulting in an almost transparent trans-ciphering process from the user point of view. This approach combined with hardware accelerations could bring homomorphic encryption closer to practice.
This paper deals the choice of FV parameters for efficient implementation of this scheme in the light of related works' common approaches. At first sight, using large polynomial degree to reduce the coefficients size seemed to be advantageous, but further observations contradict it. Large polynomial degrees imply larger ciphertexts and more complex implementations, but smaller ones imply more primes to find for CRT polynomial representation. The result of this preliminary work for the choice of an adequate hardware target motivates the choice of small degree polynomials rather than small coefficients for the FV scheme.

Keywords: Homomorphic evaluation, FV parameters, Chinese Remainder Theorem, Number Theorical Transform

1 Introduction

Privacy is one of the main concerns regarding the development of cloud services in the context of applications handling sensible data. The data privacy on remote

servers is guaranteed with standard cryptography. The issue comes up during the exploitation of these data directly on the outsourced servers. Since 2009 and the thesis of C. Gentry [15], the concept of fully homomorphic encryption introduced by Rivest, Aldeman and Dertouzos in 1978 [22], is not a conjecture any more. Homomorphic encryption schemes guarantee the equivalence of an operation between the clear data and the encrypted data algebraic systems. A fully homomorphic encryption scheme guarantees that an homomorphic equivalent can be found for any function considered over the clear data domain. Homomorphic encryption schemes are emerging along with practical approaches, but the lack of performances of software implementations makes them difficult to use in real life applications. Hardware optimizations for efficient homomorphic encryption should then be explored.

Partial Homomorphic Encryption schemes are able to homomorphically evaluate additions (e.g. Paillier) or multiplications (e.g. RSA). Problems arise to design an homomorphic encryption scheme able to evaluate both additions and multiplications. Indeed, in homomorphic cryptography a noise is added to the encrypted data for security reasons (non-deterministic encryption). If it is possible to construct Somewhat Homomorphic Encryption schemes (that could evaluate both additions and multiplications) the added noise results in a large data size expansion between clear and encrypted data. Moreover, the level of noise grows with operations in the encrypted domain, and especially with multiplications. At a certain level, the decryption primitive does not retrieve the clear data correctly.

Gentry's blueprint to construct FHE schemes is based on the bootstrapping procedure: an SHE scheme that can homomorphically evaluate its own decryption circuit and at least an other operation, becomes a FHE scheme. The first implementations of the bootstrapping procedure were impractical due to the SHE exponential noise growth, their complex decryption circuit, and their large data size expansion [12, 24]. Numerous works proposed new schemes introducing different mechanisms from sub-exponential noise growth [4, 13] to constant noise growth [3, 11]. Despite the improved performances, the bootstrapping procedure is still too complex for them to be practical. Nevertheless these new schemes lead to a compromise: they can evaluate functions with a multiplicative depth under a practical limit (20 to 30) but become impractical beyond it. The FV [11] and BGV [13] schemes are the most accepted today.

In 2013, C. Gentry et al. proposed a new approach revisiting the bootstrapping procedure to construct FHE schemes [14]. This work is followed by promising results for fast bootstrapping primitives [7, 10]. They open interesting perspectives in the definition of efficient FHE systems.

The important data size expansion inherent to homomorphic encryption implies, among other things, an overhead problem in communication costs. To solve this problem, M. Naehrig et al. [17] proposed a practical approach known as trans-ciphering: the owner encrypts its data under a standard symmetric encryption scheme, without data size expansion, and sends them to the server along with an homomorphic encryption of the symmetric key. Once the server possesses the encrypted data, the decryption function of the symmetric scheme is

homomorphically evaluated by the FHE, SHE or L-FHE scheme, resulting in homomorphic encrypted data. This approach has been improved by A. Canteaut et al. [5]. They proposed the use of lightweight additive IV-based stream ciphers as the underlying symmetric schemes. They have shown that using their approach, the trans-ciphering procedure's performance is then dependent of an intensive off-line computation part, reducing the on-line part of trans-ciphering to a simple homomorphic addition. This approach solves only the upward communication overhead as the trans-ciphering is a one-way procedure. Still, it improves the practicability of homomorphic encryption but performances are still not sufficient for software only implementations. For example, it takes ~ 35 minutes on a mid-end 48-core server to generate 57 homomorphic keystream elements that could handle up to 7 additionnal ciphertext multiplication levels [5].

The SHE scheme FV [11] handles polynomials with modular integer coefficients (500 to 5000 bits) modulo a fixed degree polynomial which is in practice a cyclotomic polynomial of rather large degree (128 to 32768). Manipulation of such polynomials is expensive, especially during multiplications. This issue is already addressed with the hardware optimization of lattice based cryptography [21]. The opportunity of lattice-based homomorphic encryption helps to extend the previous work to the context of homomorphic encryption, and in particular of homomorphic evaluation [9, 19, 23].

In lattice-based cryptography, the parameter selection is difficult in practice. To the best of our knowledge, most of works related to the hardware acceleration of homomorphic primitives tend to select previously used parameter sets. We assume it is done for comparison purposes, but the choice of parameters could have a significant impact on the correct exploitation of available hardware resources. It motivates the work presented in this paper which makes an analysis of the FV parameters for adequacy of hardware architecture and algorithm.

In this paper, we exploit the distinction between the application parameters and the implementation parameters of the FV scheme. When the security and multiplicative depth requirements are fixed (application parameters), we still have one degree of freedom to choose the cyclotomic polynomial's degree N and the size of the modulus q. By examining the algorithms of recent hardware acceleration work, both these parameters impact the resulting implementation complexities.

In a first section, the mathematical notations and the FV evaluation primitives are presented. The second section presents the profiling results that motivate focus on polynomial multiplications, and then describes the approaches proposed in hardware optimization studies to implement efficiently these operations. The third section derives from the inter-dependency of FV parameters the impacts of the degree N and the size of the modulus q on the implementation strategy. Finally, the fourth section concludes this paper.

2 Preliminaries on the FV Scheme

This section has two objectives, first to get used to the notations, and second to make the distinction between a ciphertext multiplication and polynomial multiplications occurring in ciphertext multiplication and ciphertext relinearisation. In a first subsection the mathematical representation used in this paper are presented, and the second subsection reminds the FV primitives. The third subsection presents the set of FV parameters that interests us.

2.1 Mathematical Notations

Algebraic Structure: The cyclotomic polynomial of order m is denoted $\Phi_m(X)$ and ϕ is the Euler totient function. $R_m = \mathbb{Z}[X]/(\Phi_m(X))$ refers to the ring of the polynomial classes of degree less than $N = \phi(m)$ with integer coefficients. In practice, m is selected as power of 2 and it follows that $\Phi_m(X) = X^N + 1$ and $N = m/2$.

Elements of the ring R_m are noted in lowercase bold (e.g $\mathbf{a} \in R_m$) and their coefficients in indexed lowercase (e.g. $a_i \, \forall \, i \in (0, 1, ..., N-1)$). The notation $\mathbf{a}$ is used indifferently for the polynomial or its N-point sequence of coefficients.

For an integer $q > 1$, $\mathbb{Z}_q$ is the set of integers $[-q/2, q/2)$. The unique integer in $\mathbb{Z}_q$ such that $[a]_q = a \bmod q$, $\forall \, a \in \mathbb{Z}$ is noted $[a]_q$. By extension, $R_{m,q}$ is the set of polynomials in R_m with coefficients in $\mathbb{Z}_q$. For a polynomial $\mathbf{a} \in R_m$, $[\mathbf{a}]_q$ is the polynomial in $R_{m,q}$ obtained by applying $[\cdot]_q$ to all its coefficients. The notation $[\mathbf{a}]_q$ is used indifferently for the polynomial or its N-point sequence of coefficients.

Plaintext and Ciphertext Spaces: The plaintext space of the FV scheme is defined with respect to an integer $t > 1$, and it is the set of polynomial in $R_{m,t}$ (e.g. $t = 2$).

The ciphertext space is also defined with respect to an integer $q > 1$. A ciphertext is a pair of polynomials in $R_{m,q}$. Let c be a ciphertext, its canonical form is noted $c = (\mathbf{c}_0, \mathbf{c}_1) \in R^2_{m,q}$. After multiplications, ciphertexts are in a non-canonical form that requires a relinearisation procedure. Such ciphertexts are noted $\tilde{c}$ with $\tilde{c} = (\tilde{\mathbf{c}}_0, \tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2) \in R^3_{m,q}$.

2.2 FV Primitives

In the context of stream cipher trans-ciphering, both the off-line part and on-line part are based on homomorphic evaluations. During the off-line phase, the homomorphic scheme evaluates the IV-based stream cipher, and during the on-line phase it evaluates the application required by the user. This paper focuses on the FV primitives specific to homomorphic evaluation. A complete presentation of the scheme could be found in the original work [11].

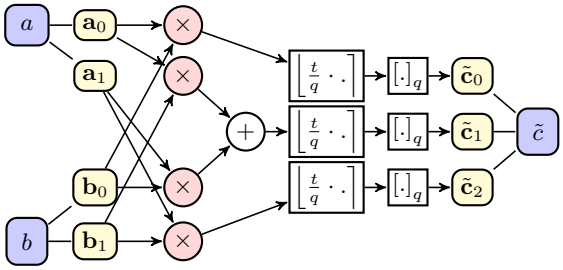
The choice is made to work with the second version of the relinearisation procedure presented in the original work. This version makes the relinearisation

primitive close to the ciphertext multiplication primitive. It is motivated by the intuition that if a hardware platform computes ciphertext multiplications efficiently, it conducts also efficient relinearisations.

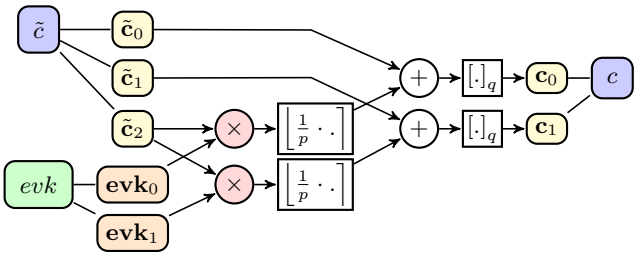
This relinearisation primitive requires the definition of an integer $p > 1$ (usually $p \geq q^3$) and a relinearisation key which is a pair of polynomials in $R_{m,p \cdot q}$. We note $rlk = (\mathbf{rlk}_0, \mathbf{rlk}_1) \in R^2_{m,p \cdot q}$ the relinearisation key of the FV instance.

Let $a = (\mathbf{a}_0, \mathbf{a}_1) \in R^2_{m,q}$ and $b = (\mathbf{b}_0, \mathbf{b}_1) \in R^2_{m,q}$ be two ciphertexts of the same FV instance. We note by $\times$ (resp. $+$) the polynomial multiplication (resp. addition) over R_m. The scalar multiplication is noted $\cdot$ and the scale-and-center-rounding operation is represented using $\lfloor S \cdot . \rceil$ with S the scaling value. Finally we remind that $[.]_q$ reduces all the polynomial coefficients to the interval $[-q/2, q/2)$.

Figure 1 shows the operation flow of the ciphertext multiplication and the ciphertext relinearisation described in the original paper of J. Fan and F. Vercauteren [11] (Sect. 4). In practice the FV multiplication is immediately followed by a relinearisation in order to always handle canonical ciphertexts. It is important to note that polynomial arithmetic takes place in R_m and it is not possible to reduce the coefficients modulo q at will.



(a) Ciphertext multiplication



(b) Ciphertext relinearisation

Fig. 1: FV multiplication and relinearisation primitives

2.3 FV Parameters

In this paper an FV instance is a particular set of FV parameters. Four parameters are considered: the security level λ, the multiplicative depth evaluation capability L, the degree of the cyclotomic polynomial N, and the size of the ciphertext's polynomial coefficients T_q. Other parameters are described in the original work [11].

The particular set of parameters (λ, L, N, T_q) has three degrees of freedom: it requires three of them to be fixed to derive the fourth. A distinction is made between the application level parameters (λ, L), and (N, T_q) which are implementation level parameters.

Both the cyclotomic polynomial degree N and the size of the coefficients T_q have an impact on handling the polynomials. The purpose of this paper is to investigate their impact on the hardware optimization strategies explored in related works.

3 Improving Performances of FV Homomorphic Evaluation

According to Amdahl's law, a dedicated hardware solution should cover the most repetitive and compute intensive operations of an application. The identification of their critical operations is usually done by profiling. When it is possible, hardware optimizations exploit different levels of parallelism and/or mathematical simplifications inherent to the underlying algorithms of these operations.

In this section, profiling results of the FV homomorphic evaluation of Trivium from A. Canteaut et al. [5] are presented. In a second subsection, common approaches for efficient polynomial multiplications are described, highlighting the influence of FV parameters.

3.1 FV Homomorphic Evaluation of Trivium

Using a library implementing the FV scheme [6], the experimentation of Canteaut et al. [5] is reproduced by executing an homomorphic evaluation of Trivium. The Valgrind tool suite [18] is used to identify the ciphertext multiplication operation as the performance bottleneck of homomorphic evaluation with more than 99% of the estimated cycles. The results of the profiling are detailed in Table 1.

The CtxtMult operation is, as explained in the previous section, decomposed into the actual ciphertext multiplication (30,6 %) and the immediately following relinearisation (62,8 %). Digging a bit more into these two steps, it appears that they both rely on the same bottleneck operation, the polynomial multiplications, realized through FFT convolutions. During the whole evaluation of Trivium-12, 76,1% of the estimated cycles are spent in these convolutions.

The ciphertext relinearisation is twice the computation workload of the ciphertext multiplication as the relinearisation key is a pair of polynomials with coefficients four times the size of ciphertext polynomials.

Table 1: Profiling results of an homomorphic evaluation of Trivium-12 with the FV implementation of Canteaut et al. [5]. FV parameters: security 80, multiplicative depth 19, polynomial cyclotomic order 4096 (implies polynomials with 2048 coefficients) modulus q size 2658-bits. The experiment uses Valgrind 3.10.

HE operation	Est. Cycles (Million cycles)	% Est.
HE Trivium	**13 337 699**	**100 %**
CtxtMult	**13 272 536**	**99,5 %**
- Relinearise	8 381 331	62,8 %
- Multiply	4 085 675	30,6 %
- Others	805 530	6,1 %
CtxtAdd	**50 923**	**0,4 %**
Others	**14 240**	**0,1 %**

This profiling confirms that the critical part of FV homomorphic evaluation is the ciphertext multiplication and relinearisation. Both rely on polynomial multiplication, which is also found in lattice-based cryptography.

3.2 Improving Polynomial Multiplications

Polynomial operations are conducted over R_m, it implies that multi-precision integer operations handle values that could grow up to $N * q^2$ during ciphertext multiplication and up to $N * p * q^2$ during ciphertext relinearisation. In practice, two integers $Q > N * q^2$ and $K > N * p * q^2$ are selected, and the polynomial multiplications are conducted over $R_{m,Q}$ and $R_{m,K}$. The size of modulus q depends on the others FV parameters, but it grows from hundred of bits up to thousand of bits in some FV instances. To tackle the large integer arithmetic, the use of RNS arithmetic through the Chinese Remainders Theorem (CRT) is quite popular for hardware optimisation approaches [8, 19, 25].

Besides the integer arithmetic, a polynomial multiplication is highly dependent of the degree involved. The naive approach for polynomial multiplication consists in computing the linear convolution product of its coefficients and has a complexity in $O(N^2)$. To reduce this complexity, the NTT based polynomial multiplication is widely used in hardware optimization works [19, 21]. A recent work from Migliore et al. [16] proposes the use of the Karatsuba polynomial multiplication algorithm for small multiplicative depth applications.

Chinese Remainder Theorem: To exploit the parallelism brought by CRT, the different modulus q, Q and K are constructed as fixed size primes' products. The number of primes required for each modulus (l_q, l_Q and l_K) depends on Tq and the desired size of these primes T_{primes} (1). By construction of the modulus Q and K, $T_Q = 2 * T_q + \log_2(N)$ and $T_K = 5 * T_q + \log_2(N)$.

$$l_q = \left\lceil \frac{T_q}{T_{\text{primes}}} \right\rceil, \quad l_Q = \left\lceil \frac{2 * T_q + \log_2(N)}{T_{\text{primes}}} \right\rceil, \quad l_K = \left\lceil \frac{5 * T_q + \log_2(N)}{T_{\text{primes}}} \right\rceil \ . \quad (1)$$

With a direct application of the CRT, the bijections $R_{m,Q} \cong (R_{m,p_0} \times ... \times R_{m,p_{l_Q-1}})$ and $R_{m,K} \cong (R_{m,p_0} \times ... \times R_{m,p_{l_K-1}})$ allow the addition of parallelism in the polynomial multiplication. The computational cost of switching the polynomial representation from $R_{m,Q}$ (resp. $R_{m,K}$) to the residue system representation, and vice versa, is not taken into account. A recent work from J-C. Bajarad et al. [2] proposes a variant of the FV scheme in which polynomials stay all along in Residue Number System representations.

The ciphertext polynomial multiplications are decomposed into l_Q (resp. l_K) independent residue polynomial multiplications during ciphertext multiplication (resp. ciphertext relinearisation). The independence of each residue polynomial multiplication implies a thread level parallelism that could be exploited through distributed computation.

Considering a residue polynomial multiplication as a simple hardware block (B_{RPM}), the latency and the hardware cost of a ciphertext polynomial multiplication block is roughly expressed in function of the number of blocks at disposal ($\#B_{\text{RPM}} \in [1; l_Q]$ (resp. $[1; l_K]$)). Equations (2) and (3) express them for a polynomial multiplication during ciphertext multiplications.

$$Lat_{\text{PolyMult}} = \left\lceil \frac{l_Q}{\#B_{\text{RPM}}} \right\rceil * Lat_{B_{\text{RPM}}} \quad (2)$$

$$HCost_{\text{PolyMult}} = \#B_{\text{RPM}} * HCost_{B_{\text{RPM}}} \quad (3)$$

As a residue polynomial has the size of its coefficients fixed by T_{primes}, the characteristics of a block B_{RPM} are independent of the parameter T_q. It can be already pointed out that when exploiting the RNS arithmetic introduced by the CRT, the parameters T_q and T_{primes} determine the thread level parallelism. The impact of the parameters N and T_{primes} on the hardware blocks B_{RPM} are detailed in the next subsection.

Polynomial Multiplication over R_{m,p_i}: For high degree polynomials, the NTT-based polynomial multiplication seems to be the most popular approach for hardware optimizations. It computes the convolution product of the polynomial multiplication through Number Theoretical Transforms (Fourier transform on finite fields), and the Cooley-Tukey algorithm reduces the NTT complexity to $O(N \log(N))$. Furthermore, exploiting the negacyclic convolution theorem, the NTT-based multiplication can directly perform polynomial multiplications modulo $\Phi_m(X) = X^N + 1$, avoiding a non-trivial polynomial modular reduction. Nevertheless this approach reduces the choice of the primes p_i, as the existence of an N-point NTT over the residue space $\mathbb{Z}_{p_i}$ must be guaranteed.

As described in [20], the existence of the N-point NTT over R_{m,p_i} is conditioned by the existence of a primitive N-root of unity ω over $\mathbb{Z}_{p_i}$. If one wants

to use the negacyclic convolution theorem over $\mathbb{Z}_{p_i}$, he has to find a primitive $2N$-root of unity ψ such that $\psi^2 = \omega \bmod p_i$. Furthermore, all the elements of $\mathbb{Z}_{p_i}$ should be invertible, this property is guaranteed by selecting p_i as a prime. According to [20], all the conditions above are satisfied if a prime p_i can be found such that $2N$ divides $(p_i - 1)$. Then it just remains the selection of appropriate ω and ψ. Efficient prime selection is adressed by the NFLlib developpement team C. Aguilar-Melchor et al. [1].

During the computation of a residue polynomial multiplication, two forward and one backward N-point NTT are computed ($O(N\log(N))$ complexity). The other operations consist in point-wise coefficient multiplications between N-point sequences ($O(N)$ complexity). It is then reasonable to focus the hardware optimizations on the NTT. We would like now to observe the impact of N on the NTT latency and hardware cost.

The implementation of an NTT with respect to a radix-2 basic block B_{RX2} is now considered. The latency and hardware cost of this block are noted respectively $Lat_{B_{\text{RX2}}}$ and $HCost_{B_{\text{RX2}}}$. An N-point NTT computation, with N a power of two, is composed of $\log_2(N)$ iterations of $N/2$ radix-2 block computations. In this work, it is assumed that an iteration has to finish before the next one can start, and that the radix-2 blocks are re-used from one iteration to another. Let $\#B_{\text{RX2}} \in [1; N/2]$ be the number of radix-2 blocks available for the computation of one iteration of an N-point NTT.

$$Lat_{B_{\text{NTT}}} = \left\lceil \frac{N}{2 * \#B_{\text{RX2}}} \right\rceil * \log_2(N) * Lat_{B_{\text{RX2}}}(T_{\text{primes}}) \tag{4}$$

$$HCost_{B_{\text{NTT}}} = \#B_{\text{RX2}} * HCost_{B_{\text{RX2}}}(T_{\text{primes}}) \tag{5}$$

Now that the impact of N over an NTT computation is known ((4) and (5)), its influence on latency and hardware cost of a residue polynomial multiplication must be expressed. It is done by considering that the N extra modular multiplications required for backward NTT are computed as an N-point wise multiplication. It is reminded that in the negacyclic convolution approach there are already four N-point wise modular multiplications to compute [21].

Equations (6) and (7) express the latency and the hardware cost of a residue polynomial multiplication with respect to an NTT block (B_{NTT}) and a modular multiplier block (B_{MM}). Let $\#B_{\text{MM}} \in [1; N]$ be the number of modular multipliers available for an N-point wise multiplication.

$$Lat_{B_{\text{RPM}}} = 3 * Lat_{B_{\text{NTT}}} + 5 * \left\lceil \frac{N}{\#B_{\text{MM}}} \right\rceil * Lat_{B_{\text{MM}}}(T_{\text{primes}}) \tag{6}$$

$$HCost_{B_{\text{RPM}}} = HCost_{B_{\text{NTT}}} + \#B_{\text{MM}} * HCost_{B_{\text{MM}}}(T_{\text{primes}}) \tag{7}$$

Both the radix-2 and the modular multiplier implementations have their efficiency related to the choice of an efficient modular reduction, but also in the choice of the T_{primes} parameter [1]. From an hardware design point of view, working with small T_{primes} could be interesting to have smaller integer arithmetic to perform.

This section has expressed some high level equations that link the FV parameters and the hardware optimization choices together. The next one describes an analysis of the FV parameters and their impact on the hardware optimization opportunities.

4 FV Parameters and Optimization Opportunities

From an applicative point of view, the security level λ and the multiplicative depth L are the parameters that determine the FV instance. But when λ and L are fixed, there is still freedom in the choice of the cyclotomic polynomial degree N and in the coefficient size T_q of the ciphertext polynomials. As seen in the previous section, the parameter T_q has an impact on thread parallelism, and the parameter N has an impact on the residue polynomial multiplication. This section discusses the impacts of the relation $T_q(N)$ over the hardware optimization strategy.

To generate correct sets of FV parameters, a Sage script [6] implementing the derivation rules from J. Fan and S. Vercauteren [11] is used. Some sets of parameters are generated with ranges that one could expect in a stream-cipher trans-ciphering context with Trivium-12 from Canteaut et al. [5]. The security level is selected between 80 and 192 and the multiplicative depth is selected between 16 and 32.

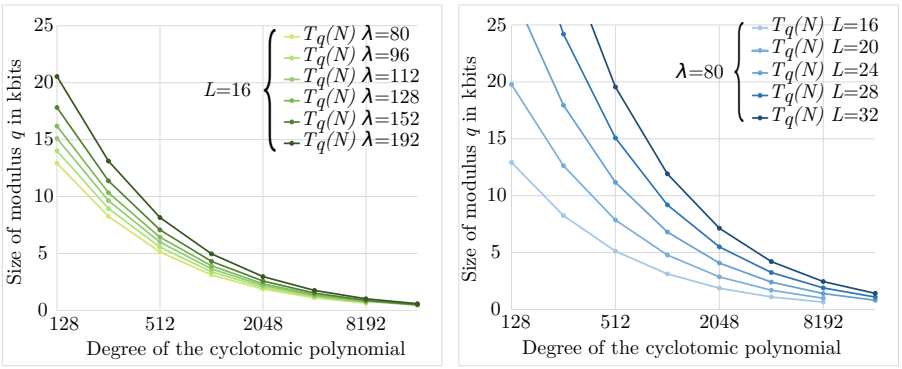
4.1 Scalability over Applicative Level Parameters

Figure 2 shows the impact of the security and the multiplicative depth parameters on the relation $T_q(N)$. The first relation $T_q(N)$ displayed is fixed for security 80 and multiplicative depth 16. Figure 2a represents the influence of the security, and 2b the multiplicative depth's influence over this relation. We observe that high degree cyclotomic polynomials reduce the influences of λ and L over coefficient sizes.

A direct relation between the latency of a polynomial multiplication during ciphertext multiplication and the size of the modulus q is expressed in equations (1), (2) and (3).

$$Lat_{\text{PolyMult}} = \left\lceil \frac{2 * T_q + \log_2(N)}{T_{\text{primes}} * \#B_{\text{RPM}}} \right\rceil * Lat_{B_{\text{RPM}}} \tag{8}$$

As explained in Sect. 3.2, the latency and the hardware cost of a residue polynomial multiplication block are only dependent of T_{primes} and N. Thus when T_{primes}, N and the hardware target ($\#B_{\text{RPM}}$) are fixed, it seems that small variations of T_q imply small variations of Lat_{PolyMult}. Naturally, it depends also on the constant $Lat_{B_{\text{RPM}}}$. An implementation that chooses to handle large N, has its latency Lat_{PolyMult} less impacted by T_q compare to one that handle smaller N. However, it is necessary to assess the influence of N over $Lat_{B_{\text{RPM}}}$ before concluding that large N reduces application parameter's influences over a given implementation.



(a) Influence of security (b) Influence of multiplicative depth

Fig. 2: Influence of λ and L on the relation $T_q(N)$. The size of the modulus q is expressed in kbits.

4.2 Smaller Ciphertexts

The ciphertext size is a high level indicator for memory requirements (storage capacity, access latency...) that directly impact performances. Without considering any parallelism, implementation details or optimized data accesses, a coarse grain relation could be expressed: the smaller the ciphertexts, the better the performances.

Figure 3 shows the impact of the security and multiplicative depth on the ciphertext size, which is directly related to T_q and N, CtxtSize $= 2 * (T_q * N)$. The observation shows that a large degree N increases the influence of security and multiplicative depth over the size of the ciphertexts. This is mildly counterintuitive with the influence of N on the size of the modulus q. It seems now more interesting to select small N, and further discussions confirms it.

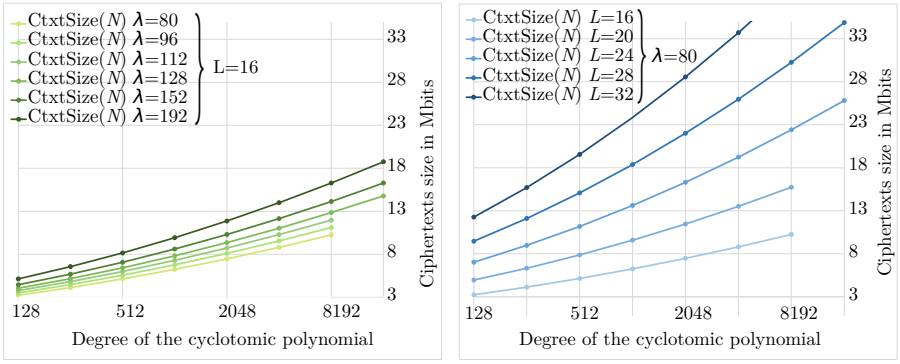
4.3 Influence of N on Residue Polynomial Multiplications

The description of the NTT-based polynomial multiplication presents the bottleneck operation as the computation of forward and backward N-point NTT. Equations (4), (6) and equations (5), (7) have expressed the latency and the hardware cost of a residue polynomial multiplication function of the FV parameter N and the hardware blocks availability $\#B_{\text{RX2}}$, $\#B_{\text{MM}}$.

$$Lat_{B_{\text{RPM}}} = 3 * \left\lceil \frac{N}{2 * \#B_{\text{RX2}}} \right\rceil * \log_2(N) * Lat_{B_{\text{RX2}}} + 5 * \left\lceil \frac{N}{\#B_{\text{MM}}} \right\rceil * Lat_{B_{\text{MM}}} \quad (9)$$

$$HCost_{B_{\text{RPM}}} = \#B_{\text{RX2}} * HCost_{B_{\text{RX2}}} + \#B_{\text{MM}} * HCost_{B_{\text{MM}}} \quad (10)$$

Because N is a power of two, selecting larger N has a major impact on the latency and/or on the hardware implementation cost. In Fig. 4, the theoretical



(a) Influence of security (b) Influence of multiplicative depth

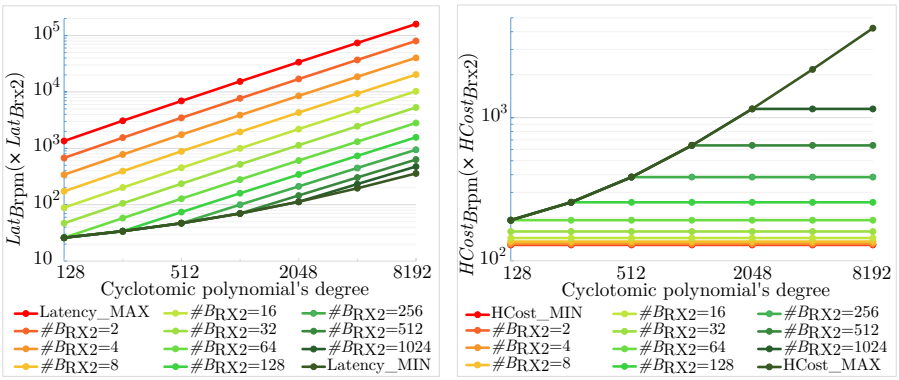
Fig. 3: Influence of λ and L on the ciphertext's size, expressed in Mbits.

latency and hardware cost function of N is represented, and this for different levels of parallelism introduced in a NTT computation. In this representation, the latency and the hardware cost of the N-point wise multiplications are fixed by choosing $\#B_{\text{MM}} = 128$. The latency (resp. the hardware cost) is expressed as multiples of $Lat_{B_{\text{RX2}}}$ (resp. $HCost_{B_{\text{RX2}}}$). For more simplicity $Lat_{B_{\text{RX2}}}$ is considered equivalent to $Lat_{B_{\text{MM}}}$ (resp. $HCost_{B_{\text{RX2}}} \sim HCost_{B_{\text{MM}}}$).

This theoretical experimentation is considered as a best case scenario due to the approximations made in (4), (5), (6) and (7). Indeed the data dependencies in the NTT computations, the storage cost of pre-evaluated factors, and the memory access latencies are not taken into account. In each residue space R_{m,p_i}, $2N$ factors have to be pre-computed for an NTT based negacyclic convolution. Thus doubling N roughly doubles the number of twiddle factors and memory accesses.

As displayed in figures 4a and 4b, and accordingly to equations (9) and (10), choosing larger N linearly increases the latency of a residue polynomial multiplication. Similarly, to guarantee a low latency residue polynomial multiplication for any N, one has to pay an extra hardware cost which linearly increases with N. Nevertheless, N exponentially increases as its value is restricted to be power of two (to have both batching property [24] and nega-cyclic convolution) and naturally, the latency and the hardware cost of a residue polynomial multiplication suffer from this exponentiation.

Considering now that N is fixed, the equations (9) and (10) show that the latency decreases with larger $\#B_{\text{RX2}}$, but the hardware cost similarly increases. This linear behaviour is also reminiscent with $\#B_{\text{RMP}}$ in equations (2) and (3).



(a) B_{RPM} latency (b) B_{RPM} hardware cost

Fig. 4: Influence of the FV parameter N over the latency and the hardware cost of a residue polynomial multiplication, with $\#B_{\text{MM}} = 128$ and for different $\#B_{\text{RX2}}$.

4.4 Influence of T_q on Parallelism from the CRT

As described in subsection 3.2, the CRT parallelism capability is theoretically enforced by large T_q that increases the number of residue spaces for a fixed prime size T_{primes}. In our context, the considered choices of N, and its implication on T_q, always enable a valuable acceleration when exploiting the CRT parallelism.

Two limitations to the hardware optimization at CRT level are identified. The first one is not considered in this paper and is fixed by the extra computation added by the switches between the CRT representation of a polynomial and its standard representation with coefficients over $R_{m,q}$. The second is the availability of residue spaces for a given size T_{primes}. Indeed, the use of the CRT is conditioned by the existence of l_K primes of size T_{primes}. Moreover, the NTT-based polynomial multiplication brings an additional condition over the choice of those primes to guarantee the existence of the $2N$-NTT in the residue spaces.

The number of required primes l_K is directly dependent of the relation $T_q(N)$ and the size T_{primes} (cf. equation 1). The number of primes required to represent a polynomial in $R_{m,q}$ using a residue representation is presented in fig. 5, and this for different N and size of primes. To conduct the polynomial multiplications over $R_{m,Q}$ (resp. $R_{m,K}$) with the CRT approach, one has to find roughly $l_Q \sim 2 * l_q$ (resp. $l_K \sim 5 * l_q$) primes.

As introduced in section 3.2, the choice of a small T_{primes} is interesting to reduce the latency and the hardware cost of a modular multiplier block, and has a direct impact on the efficiency of a residue polynomial multiplication. But according to the Prime Number Theorem, the number of prime smaller than n is roughly $\pi(n) \sim n/\log(n)$. Finding enough primes with $T_{\text{primes}} \geq 32$, is not an issue, but considering smaller prime sizes, the number of candidates quickly drops down. The GMP library is used to find a maximum number of primes that

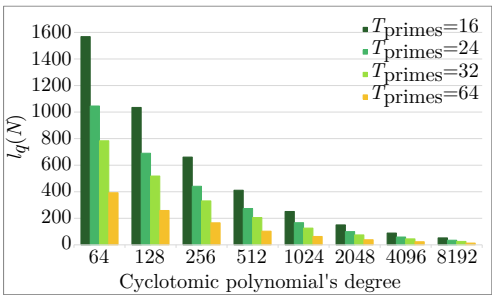


Fig. 5: Influence of N over l_q for $T_{\text{primes}} \in (16, 24, 32, 64)$-bits. Security is fixed at 80 and multiplicative depth at 18.

satisfy the NTT requirement when T_{primes} is 16-bits and 24-bits, the results are shown in Table 2.

The comparison of the number of primes found over 16-bits and 24-bits sizes with the requirements in Fig. 5, shows a limitation in the choice of small T_{primes} to improve the efficiency of basic arithmetic blocks. Nevertheless, as observed in Fig. 5, and even with large N, l_q (and by extension l_Q and l_K) is still large enough to exploit the parallelism bring by CRT.

Table 2: Number of primes allowing N-point NTT transform over $\mathbb{Z}_{p_i}$.

Degree N	128	256	512	1024	2048	4096	8192
16-bits **primes**	47	24	14	5	3	1	0
24-bits **primes**	8430	4230	2134	1047	536	260	130

5 Conclusion

In this paper, insights over the choices of FV parameters are provided for efficient hardware resources exploitation for its evaluation primitives. The analysis was conducted in the context of stream-cipher based trans-ciphering using the Trivium cipher. This analysis was also based both on CRT and NTT approaches to conduct the polynomial multiplications over $R_{m,Q}$ (resp. $R_{m,K}$). This choice was motivated by the existing works on hardware optimization of lattice-based cryptography.

The distinction between application level parameters (λ,L) and implementation level parameters (N, T_q) has been expressed. During the analysis some observations have been brought to help in the choice of appropriate FV implementation parameters for a flexible and efficient hardware implementation, regarding the application parameters.

Larger N reduces the impact of security and multiplicative depth over coefficient's size but still increases ciphertext's size. The cyclotomic polynomial degree N, being a power of two, makes the design of efficient NTT-based polynomial multiplications difficult with increasing N. Furthermore, the CRT parallelism is easier to exploit than large polynomial multiplication as explained by J-C. Bajard et al. [2]. It implies that all (N, T_q) are not quite the same for a fixed (λ, L), and small degree N should be preferred over small coefficient size T_q.

This analysis was a preliminary study for adequacy of hardware architecture and algorithms underlying the FV evaluations primitives. Despite the motivation of choosing small N, concrete choice of (N, T_q) still depends on the practical limitations of the targeted hardware, and in particular on the memory access bandwidth and the available computing resources.

References

1. Aguilar-Melchor, C., Barrier, J., Guelton, S., Guinet, A., Killijian, M.O., Lepoint, T.: NFLlib: NTT-based fast lattice library. In: Topics in Cryptology - CT-RSA 2016, pp. 341–356. Springer Nature (2016)
2. Bajard, J.C., Eynard, J., Hasan, A., Zucca, V.: A Full RNS Variant of FV like Somewhat Homomorphic Encryption Schemes. In: Selected Areas in Cryptography - SAC. St. John's, Newfoundland and Labrador, Canada (Aug 2016), `http://hal.upmc.fr/hal-01371941`
3. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Advances in Cryptology–CRYPTO 2012, pp. 868–886. Springer (2012)
4. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. IEEE ASFC 2011 (2) (2011)
5. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In: Fast Software Encryption, pp. 313–333. Springer Nature (2016)
6. Carpov, S., Dubrulle, P., Sirdey, R.: Armadillo: A compilation chain for privacy preserving applications. In: Proceedings of the 3rd International Workshop on Security in Cloud Computing. Association for Computing Machinery (ACM) (2015)
7. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: Advances in Cryptology – ASIACRYPT 2016, pp. 3–33. Springer Nature (2016)
8. Dai, W., Doroz, Y., Sunar, B.: Accelerating ntru based homomorphic encryption using gpus. In: High Performance Extreme Computing Conference (HPEC), 2014 IEEE. pp. 1–6. IEEE (2014)
9. Dai, W., Sunar, B.: cuhe: A homomorphic encryption accelerator library. In: Cryptography and Information Security in the Balkans, pp. 169–186. Springer (2015)
10. Ducas, L., Micciancio, D.: Fhew: Bootstrapping homomorphic encryption in less than a second. In: Advances in Cryptology–EUROCRYPT 2015, pp. 617–640. Springer (2015)
11. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive 2012, 144 (2012)

12. Gentry, C., Halevi, S.: Implementing gentrys fully-homomorphic encryption scheme. In: Advances in Cryptology–EUROCRYPT 2011, pp. 129–148. Springer (February 2011)
13. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the aes circuit. In: Advances in Cryptology–CRYPTO 2012, pp. 850–867. Springer (2012)
14. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013, pp. 75–92. Springer (2013)
15. Gentry, C., et al.: Fully homomorphic encryption using ideal lattices. In: STOC. vol. 9, pp. 169–178 (May 2009)
16. Migliore, V., Real, M.M., Lapotre, V., Tisserand, A., Fontaine, C., Gogniat, G.: Hardware/software co-design of an accelerator for FV homomorphic encryption scheme using karatsuba algorithm. IEEE Transactions on Computers pp. 1–1 (2016)
17. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM workshop on Cloud computing security workshop. pp. 113–124. ACM (2011)
18. Nethercote, N., Walsh, R., Fitzhardinge, J.: "building workload characterization tools with valgrind". In: 2006 IEEE International Symposium on Workload Characterization. Institute of Electrical and Electronics Engineers (IEEE) (oct 2006)
19. Öztürk, E., Doröz, Y., Sunar, B., Savas, E.: Accelerating somewhat homomorphic evaluation using fpgas. IACR Cryptology ePrint Archive 2015, 294 (2015)
20. Pollard, J.M.: The fast fourier transform in a finite field. Mathematics of computation 25(114), 365–374 (1971)
21. Pöppelmann, T., Güneysu, T.: Towards practical lattice-based public-key encryption on reconfigurable hardware. In: International Conference on Selected Areas in Cryptography. pp. 68–85. Springer (2013)
22. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations of secure computation 4(11), 169–180 (1978)
23. Roy, S.S., Järvinen, K., Vercauteren, F., Dimitrov, V., Verbauwhede, I.: Modular hardware architecture for somewhat homomorphic function evaluation. In: Cryptographic Hardware and Embedded Systems–CHES 2015, pp. 164–184. Springer (2015)
24. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Public Key Cryptography–PKC 2010, pp. 420–443. Springer (Jan 2010)
25. Wang, W., Chen, Z., Huang, X.: Accelerating leveled fully homomorphic encryption using gpu. In: Circuits and Systems (ISCAS), 2014 IEEE International Symposium on. pp. 2800–2803. IEEE (2014)