A Comparison of the Homomorphic Encryption Schemes FV and YASHE

Tancrède Lepoint[1,⋆] and Michael Naehrig[2]

[1] CryptoExperts, École Normale Supérieure and University of Luxembourg
tancrede.lepoint@cryptoexperts.com
[2] Microsoft Research
mnaehrig@microsoft.com

Abstract. We conduct a theoretical and practical comparison of two Ring-LWE-based, scale-invariant, leveled homomorphic encryption schemes – Fan and Vercauteren's adaptation of BGV and the YASHE scheme proposed by Bos, Lauter, Loftus and Naehrig. In particular, we explain how to choose parameters to ensure correctness and security against lattice attacks. Our parameter selection improves the approach of van de Pol and Smart to choose parameters for schemes based on the Ring-LWE problem by using the BKZ-2.0 simulation algorithm.

We implemented both encryption schemes in C++, using the arithmetic library FLINT, and compared them in practice to assess their respective strengths and weaknesses. In particular, we performed a homomorphic evaluation of the lightweight block cipher SIMON. Combining block ciphers with homomorphic encryption allows to solve the gargantuan ciphertext expansion in cloud applications.

1 Introduction

In 2009, Gentry proposed the first fully homomorphic encryption scheme [16]. A fully homomorphic encryption (FHE) scheme is an encryption scheme that allows, from ciphertexts $E(a)$ and $E(b)$ encrypting bits a, b, to obtain encryptions of $\neg a$, $a \wedge b$ and $a \vee b$ without using the secret key. Clearly, this allows to publicly evaluate any Boolean circuit given encryptions of the input bits. This powerful primitive has become an active research subject in the last four years. Numerous schemes based on different hardness assumptions have been proposed [16,12,5,4,30,20] and have improved upon previous approaches.

In all of the aforementioned schemes, a ciphertext contains a noise that grows with each homomorphic operation. The noise is minimal when the ciphertext is a fresh encryption of a plaintext bit and has not yet been operated on. Homomorphic operations as those above can be (and are often) expressed as homomorphic addition and multiplication operations, *i.e.* addition and multiplication in the binary field $\mathbb{F}_2$. Both increase the noise in ciphertexts, which means that the noise

⋆ This work was started when the first author was an intern in the Cryptography Research group at Microsoft Research.

in a resulting encryption is larger than the noise in the respective input encryptions. In particular, homomorphic multiplication increases the noise term significantly.

After a certain amount of such homomorphic computations have been carried out, the noise reaches a certain maximal size after which no more homomorphic operations can be done without losing correctness of the encryption scheme. At this point, the ciphertext needs to be publicly refreshed to allow subsequent homomorphic operations. This refreshing procedure is called bootstrapping and is very costly. As a consequence, only few of the FHE schemes have been *fully* implemented [17,11,9] and the resulting performances are rather unsatisfactory.

However, real-world applications do not necessarily need to handle any input circuit. One might avoid using the bootstrapping procedure if the multiplicative depth of the circuit to be evaluated is known in advance and small enough (*cf.* [33,21,24,3] and even [19]). Unfortunately, for the schemes of [17,11,9] the noise grows exponentially with the depth of the circuit being evaluated, severely limiting the circuits that can be evaluated with reasonable parameters. To mitigate this noise growth, Brakerski, Gentry and Vaikuntanathan introduced the notion of *leveled* homomorphic encryption schemes [5]. In such a scheme, the noise grows only linearly with the multiplicative depth of the circuit being evaluated. Therefore for a given circuit of reasonable depth, one can select the parameters of the scheme to homomorphically evaluate the circuit in a reasonable time. They describe a leveled homomorphic encryption scheme called BGV using a modulus switching technique. Furthermore, this scheme and other ring-based homomorphic encryption schemes allow the use of larger plaintext spaces, where bits are replaced by polynomials with coefficients modulo a plaintext modulus possibly different from 2. Such plaintext spaces allow the encryption of more information in a single ciphertext, for example via batching of plaintext bits. Unfortunately, to homomorphically evaluate a circuit of multiplicative depth d using the modulus switching technique, the public key needs to contain d distinct versions of a so-called evaluation key.

At Crypto 2012, Brakerski proposed the new notion of *scale-invariance* [4] for leveled homomorphic encryption schemes. In contrast to a scheme that uses modulus switching, the ciphertexts for a scale-invariant scheme keep the same modulus during the whole homomorphic evaluation and only one copy of the scale-invariant evaluation key has to be stored. This technique has been adapted to the BGV scheme [5] by Fan and Vercauteren [14], and to López-Alt, Tromer and Vaikuntanathan's scheme [30] by Bos, Lauter, Loftus and Naehrig [3].[1] The resulting schemes are called FV and YASHE, respectively. No implementation of the FV scheme is known (except for a proof-of-concept implementation in a computer algebra system that is used in [21]). The YASHE scheme [3] was the first (and only) scale-invariant leveled homomorphic encryption scheme implemented so far. Very satisfactory timings are claimed for a small modulus (then able to

[1] This technique was also adapted to the homomorphic encryption scheme over the integers [12] by Coron, Lepoint and Tibouchi [10].

handle only circuits of multiplicative depth at most 2) on a personal computer. Unfortunately the implementation is not openly available for the community.

Sending Data to the Cloud. In typical real-world scenarios for using FHE with cloud applications, one or more clients communicate with a cloud service. They upload data encrypted with an FHE scheme under the public key of a specific user. The cloud can process this data homomorphically and return an encrypted result. Unfortunately, ciphertext expansion (*i.e.* the ciphertext size divided by the plaintext size) of current FHE schemes is prohibitive (thousands to millions). For example using techniques in [11] (for 72 bits of claimed security), sending 4MB of data on which the cloud is allowed to operate, would require to send more than 73TB of encrypted data over the network.

To solve this issue, it was proposed in [33] to instead send the data encrypted *with a block cipher* (in particular AES). The cloud service then encrypts the ciphertexts with the FHE scheme and the user's public key and *homomorphically decrypts* them before they are processed. Therefore, network communication is lowered to the data size (which is optimal) plus a costly *one-time setup* that consists of sending the FHE public key and an FHE encryption of the block cipher secret key.

The AES circuit was chosen as a standard circuit to evaluate because it is nontrivial (but still reasonably small) and has an algebraic structure that works well with the plaintext space of certain homomorphic encryption schemes [19]. However, there might be other ciphers that are more suitable for being evaluated under homomorphic encryption. In June 2013, the U.S. National Security Agency unveiled a family of lightweight block ciphers called SIMON [2]. These block ciphers were engineered to be extremely small, easy to implement and efficient in hardware. SIMON has a classical Feistel structure and each round only contains one AND. This particularly simple structure is a likely candidate for homomorphic cryptography.

Our Contributions. In this work, we provide a concrete comparison of the supposedly most practical leveled homomorphic encryption schemes FV and YASHE. (To our knowledge, this is the first comparison of leveled homomorphic encryption schemes.) In particular, we revisit and provide precise upper bounds for the norm of the noises in the FV scheme, as done for the YASHE scheme in [3]. It appears from our work that the FV scheme has a theoretical smaller noise growth than YASHE.

We revisit van de Pol and Smart's approach [35] to derive secure parameters for these schemes. They use the BKZ-2.0 simulation algorithm [7,8] (the most up-to-date lattice basis reduction algorithm) to determine an upper bound on the modulus to ensure a given level of security. We show that their methodology has some small limitations and we describe how to resolve them. The resulting method yields a more conservative but meaningful approach to select parameters for lattice-based cryptosystems.

Finally, we propose proof-of-concept implementations of both FV and YASHE in C++ using the arithmetic library FLINT [23]. This allows us to *practically* compare the noise growth and the performances of the FV and YASHE schemes. The implementations provide insights into the behavior of these schemes for circuits of multiplicative depth larger than 2 (contrary to the implementation described in [3]). For this purpose, we implemented SIMON-32/64 using FV, YASHE and the batch integer-based scheme from [10]. Our implementations are publicly available for the community to reproduce our experiments [26]. Due to the similarity in the design of the FV and YASHE schemes and the common basis of our implementations, we believe that our comparison gives meaningful insights into which scheme to use according to the desired application, and on the achievable performance of leveled homomorphic encryption.

2 Preliminaries

In this section, we provide a succinct background on lattices, the (Ring) Learning With Errors problem and recall the FV [14] and YASHE [3] leveled homomorphic encryption schemes.

2.1 Lattices

A (full-rank) *lattice* of dimension m is a discrete additive subgroup of $\mathbb{R}^m$. For any such lattice $L \neq \{0\}$, there exist linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \mathbb{R}^m$ such that $L = \mathbf{b}_1\mathbb{Z} \oplus \cdots \oplus \mathbf{b}_m\mathbb{Z}$. This set of vectors is called a *basis* of the lattice. Thus a lattice can be represented by its basis matrix $\mathbf{B} \in \mathbb{R}^{m \times m}$, *i.e.* the matrix consisting of the rows $\mathbf{b}_i$ in the canonical basis of $\mathbb{R}^m$. In particular, we have $L = \{\mathbf{z} \cdot \mathbf{B} : \mathbf{z} \in \mathbb{Z}^m\}$. The determinant (or volume) of a lattice is defined as $\det(L) = (\det(\mathbf{B}\mathbf{B}^t))^{1/2} = |\det(\mathbf{B})|$, where $\mathbf{B}$ is any basis of L. This quantity is well-defined since it is independent of the choice of basis.

Among all the bases of a lattice L, some are 'better' than others. The goal of lattice basis reduction is to shorten the basis vectors and thus, since the determinant is invariant, to make them more orthogonal. In particular, any basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_m)$ can be uniquely written as $\mathbf{B} = \mu \cdot \mathbf{D} \cdot \mathbf{Q}$ where $\mu = (\mu_{ij})$ is lower triangular with unit diagonal, $\mathbf{D}$ is diagonal with positive coefficients and $\mathbf{Q}$ has orthogonal row vectors. We call $\mathbf{B}^* = \mathbf{D} \cdot \mathbf{Q}$ the Gram-Schmidt orthogonalization of $\mathbf{B}$, and $\mathbf{D} = \mathrm{diag}(\|\mathbf{b}_1^*\|, \ldots, \|\mathbf{b}_m^*\|)$ is the diagonal matrix formed by the ℓ_2-norms $\|\mathbf{b}_i^*\|$ of the Gram-Schmidt vectors.

Following the approach popularized by Gama and Nguyen [15], we say that a specific basis $\mathbf{B}$ has *root Hermite factor* γ if its element of smallest norm $\mathbf{b}_1$ (*i.e.* we assume that basis vectors are ordered by their norm) satisfies

$$\|\mathbf{b}_1\| = \gamma^m \cdot |\det(\mathbf{B})|^{1/m} .$$

By using lattice basis reduction algorithms, one aims to determine an output lattice basis with guaranteed norm and orthogonality properties. A classical lattice basis reduction algorithm is LLL (due to Lenstra, Lenstra and Lovász [25]), which ensures that for all $i < m$, $\delta_{\mathsf{LLL}}\|\mathbf{b}_\mathbf{i}^*\|^2 \leqslant \|\mathbf{b}_{i+1}^* + \mu_{i+1i}\mathbf{b}_i^*\|^2$ for a given parameter $\delta_{\mathsf{LLL}} \in (1/4, 1]$. The LLL algorithm runs in polynomial-time and provides bases of quite decent quality. For many cryptanalytic applications, Schnorr and Euchner's blockwise algorithm BKZ [36] is the most practical algorithm for lattice basis reduction in high dimensions. It provides bases of higher quality but its running time increases significantly with the blocksize. Now if A denotes a lattice basis reduction algorithm, applying it to $\mathbf{B}$ yields a reduced basis $\mathbf{B}' = \mathsf{A}(\mathbf{B})$. Thus we can define $\gamma_{\mathsf{A}(\mathbf{B})}$ as the value such that

$$\|\mathbf{b}'_1\| = \gamma_{\mathsf{A}(\mathbf{B})}^m \cdot |\det(\mathbf{B}')|^{1/m} = \gamma_{\mathsf{A}(\mathbf{B})}^m \cdot |\det(\mathbf{B})|^{1/m} .$$

It is conjectured [15,7] that the value $\gamma_{\mathsf{A}(\mathbf{B})}$ depends mostly on the lattice basis reduction algorithm, and not on the input basis $\mathbf{B}$ (unless it has a special structure and cannot be considered random). Thus, in this paper, we refer to this value as γ_{A}. For example for LLL and $\mathsf{BKZ\text{-}20}$ (*i.e.* BKZ with a blocksize $\beta = 20$), in the literature one can find the well-known values $\gamma_{\mathsf{LLL}} \approx 1.021$ and $\gamma_{\mathsf{BKZ\text{-}20}} \approx 1.013$.

2.2 Ring-LWE

In this section, we briefly introduce notation for stating the Ring-LWE-based homomorphic encryption schemes FV and YASHE, and formulate the Ring Learning With Errors (RLWE) Problem relating to the security of the two schemes. For further details, we refer to [31], [14], and [3].

Let d be a positive integer and let $\Phi_d(x) \in \mathbb{Z}[x]$ be the d-th cyclotomic polynomial. Let $R = \mathbb{Z}[x]/(\Phi_d(x))$, *i.e.* the ring R is isomorphic to the ring of integers of the d-th cyclotomic number field. The elements of R are polynomials with integer coefficients of degree less than $n = \varphi(d)$. For any polynomial $a = \sum_{i=0}^{n} a_i x^i \in \mathbb{Z}[x]$, let $\|a\|_\infty = \max\{|a_i| : 0 \leqslant i \leqslant n\}$ be the infinity norm of a. When multiplying elements of R, the norm of the product grows at most with a factor $\delta = \sup\{\|ab\|_\infty/\|a\|_\infty\|b\|_\infty : a, b \in R\}$, the so-called expansion factor. For an integer modulus $q > 0$, define $R_q = R/qR$. If t is another positive integer, let $r_t(q)$ be the reduction of q modulo t into the interval $[0, t)$, and let $\Delta = \lfloor q/t \rfloor$, then $q = \Delta t + r_t(q)$. Denote by $[\cdot]_q$ reduction modulo q into the interval $(-q/2, q/2]$ of an integer or integer polynomial (coefficient wise). Fix an integer base w and let $\ell_{w,q} = \lfloor \log_w(q) \rfloor + 1$. Then a polynomial $a \in R_q$ can be written in base w as $\sum_{i=0}^{\ell_{w,q}-1} a_i w^i$, where $a_i \in R$ with coefficients in $(-w/2, w/2]$. Define $\mathsf{WordDecomp}_{w,q}(a) = ([a_i]_w)_{i=0}^{\ell_{w,q}-1} \in R^{\ell_{w,q}}$ and $\mathsf{PowersOf}_{w,q}(a) = ([aw^i]_q)_{i=0}^{\ell_{w,q}-1} \in R^{\ell_{w,q}}$. Note that

$$\langle \mathsf{WordDecomp}_{w,q}(a), \mathsf{PowersOf}_{w,q}(b) \rangle = ab \pmod{q} .$$

Let χ_{key} and χ_{err} be two discrete, bounded probability distributions on R. In practical instantiations, the distribution χ_{err} is typically a truncated discrete Gaussian distribution that is statistically close to a discrete Gaussian. The distribution χ_{key} is chosen to be a very narrow distribution, sometimes even such that the coefficients of the sampled elements are in the set $\{-1, 0, 1\}$. We denote the bounds corresponding to these distributions by B_{key} and B_{err}, respectively.

This means that $\|e\|_\infty < B_{\text{err}}$ for $e \leftarrow \chi_{\text{err}}$ and $\|f\|_\infty < B_{\text{key}}$ for $f \leftarrow \chi_{\text{key}}$. With the help of χ_{key} and χ_{err}, we define the Ring-LWE distribution on $R_q \times R_q$ as follows: sample $a \leftarrow R_q$ uniformly at random, $s \leftarrow \chi_{\text{key}}$ and $e \leftarrow \chi_{\text{err}}$, and output $(a, [as + e]_q)$.

Next, we formulate a version of the Ring-LWE problem that applies to the schemes FV and YASHE considered in this paper.

Definition 1 (Ring-LWE problem). *With notation as above, the* Ring-Learning With Errors Problem *is the problem to distinguish with non-negligible probability between independent samples $(a_i, [a_i s + e_i]_q)$ from the Ring-LWE distribution and the same number of independent samples (a_i, b_i) from the uniform distribution on $R_q \times R_q$.*

In order for FV and YASHE to be secure, the RLWE problem as stated above needs to be infeasible. We refer to [14] and [3] for additional assumptions and detailed discussions of the properties of χ_{key} and χ_{err}.

2.3 The Fully Homomorphic Encryption Scheme FV

Fan and Vercauteren [14] port Brakerski's scale-invariant FHE scheme introduced in [4] to the RLWE setting. Using the message encoding as demonstrated in an RLWE encryption scheme presented in an extended version of [31] makes it possible to avoid the modulus switching technique for obtaining a leveled homomorphic scheme. We briefly summarize (a slightly generalized version of) the FV scheme in this subsection.

- $\mathsf{FV.ParamsGen}(\lambda)$: Given the security parameter λ, fix a positive integer d that determines R, moduli q and t with $1 < t < q$, distributions $\chi_{\text{key}}, \chi_{\text{err}}$ on R, and an integer base $w > 1$. Output $(d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$.
- $\mathsf{FV.KeyGen}(d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$: Sample $s \leftarrow \chi_{\text{key}}$, $a \leftarrow R_q$ uniformly at random, and $e \leftarrow \chi_{\text{err}}$ and compute $b = [-(as + e)]_q$. Sample $\mathbf{a} \leftarrow R_q^{\ell_{w,q}}$ uniformly at random, $\mathbf{e} \leftarrow \chi_{err}^{\ell_{w,q}}$, compute $\boldsymbol{\gamma} = ([\mathsf{PowersOf}_{w,q}(s^2) - (\mathbf{e} + \mathbf{a} \cdot s)]_q, \mathbf{a}) \in R^{\ell_{w,q}}$,and output $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) = ((b, a), s, \boldsymbol{\gamma})$.
- $\mathsf{FV.Encrypt}((b, a), m)$: The message space is R/tR. For a message $m + tR$, sample $u \leftarrow \chi_{\text{key}}$, $e_1, e_2 \leftarrow \chi_{\text{err}}$, and output the ciphertext $\mathbf{c} = ([\Delta[m]_t + bu + e_1]_q, [au + e_2]_q) \in R^2$.
- $\mathsf{FV.Decrypt}(s, \mathbf{c} = (c_0, c_2))$: Output $m = [\lfloor t/q \cdot [c_0 + c_1 s]_q \rceil]_t \in R_t$.
- $\mathsf{FV.Add}(\mathbf{c_1}, \mathbf{c_2})$: Given ciphertexts $\mathbf{c_1} = (c_{1,0}, c_{1,1})$ and $\mathbf{c_2} = (c_{2,0}, c_{2,1})$, output $\mathbf{c_{add}} = ([c_{1,0} + c_{2,0}]_q, [c_{1,1} + c_{2,1}]_q)$.
- $\mathsf{FV.ReLin}(\tilde{\mathbf{c}}_{\text{mult}}, \mathsf{evk})$: Let $(\mathbf{b}, \mathbf{a}) = \mathsf{evk}$ and let $\tilde{\mathbf{c}}_{\text{mult}} = (c_0, c_1, c_2)$. Output the ciphertext

$$([c_0 + \langle \mathsf{WordDecomp}_{w,q}(c_2), \mathbf{b} \rangle]_q, [c_1 + \langle \mathsf{WordDecomp}_{w,q}(c_2), \mathbf{a} \rangle]_q).$$

- $\mathsf{FV.Mult}(\mathbf{c_1}, \mathbf{c_2}, \mathsf{evk})$: Output the ciphertext $\mathbf{c}_{\text{mult}} = \mathsf{FV.ReLin}(\tilde{\mathbf{c}}_{\text{mult}}, \mathsf{evk})$, where

$$\tilde{\mathbf{c}}_{\text{mult}} = (c_0, c_1, c_2) = \left(\left[\left\lfloor \frac{t}{q} c_{1,0} c_{2,0} \right\rceil \right]_q, \left[\left\lfloor \frac{t}{q} (c_{1,0} c_{2,1} + c_{1,1} c_{2,0}) \right\rceil \right]_q, \left[\left\lfloor \frac{t}{q} c_{1,1} c_{2,1} \right\rceil \right]_q \right).$$

2.4 The Fully Homomorphic Encryption Scheme YASHE

In [3], a fully homomorphic encryption scheme is introduced that is based on the modified version of NTRU by Stehlé and Steinfeld [38] and the multi-key fully homomorphic encryption scheme presented in [30]. In this subsection, we state the more practical variant of the leveled homomorphic scheme from [3].

- $\mathsf{YASHE.ParamsGen}(\lambda)$: Given the security parameter λ, fix a positive integer d that determines R, moduli q and t with $1 < t < q$, distributions $\chi_{\text{key}}, \chi_{\text{err}}$ on R, and an integer base $w > 1$. Output $(d, q, t, \chi_{key}, \chi_{err}, w)$.
- $\mathsf{YASHE.KeyGen}(d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$: Sample $f', g \leftarrow \chi_{\text{key}}$ and let $f = [tf' + 1]_q$. If f is not invertible modulo q, choose a new f'. Compute the inverse $f^{-1} \in R$ of f modulo q and set $h = [tgf^{-1}]_q$. Sample $\mathbf{e}, \mathbf{s} \leftarrow \chi_{err}^{\ell_{w,q}}$, compute $\boldsymbol{\gamma} = [\mathsf{PowersOf}_{w,q}(f) + \mathbf{e} + h \cdot \mathbf{s}]_q \in R^{\ell_{w,q}}$ and output $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) = (h, f, \boldsymbol{\gamma})$.
- $\mathsf{YASHE.Encrypt}(h, m)$: The message space is R/tR. For a message $m + tR$, sample $s, e \leftarrow \chi_{\text{err}}$, and output the ciphertext $c = [\Delta[m]_t + e + hs]_q \in R$.
- $\mathsf{YASHE.Decrypt}(f, c)$: Decrypt a ciphertext c by $m = [\lfloor t/q \cdot [fc]_q \rceil]_t \in R$.
- $\mathsf{YASHE.Add}(c_1, c_2)$: Output $c_{\text{add}} = [c_1 + c_2]_q$.
- $\mathsf{YASHE.KeySwitch}(\tilde{c}_{\text{mult}}, \mathsf{evk})$: Output $[\langle \mathsf{WordDecomp}_{w,q}(\tilde{c}_{\text{mult}}), \mathsf{evk} \rangle]_q$.
- $\mathsf{YASHE.Mult}(c_1, c_2, \mathsf{evk})$: Output the ciphertext

$$c_{\text{mult}} = \mathsf{YASHE.KeySwitch}(\tilde{c}_{\text{mult}}, \mathsf{evk}), \text{ where } \tilde{c}_{\text{mult}} = \left[\lfloor t/q \cdot c_1 c_2 \rceil\right]_q .$$

3 Parameter Derivation

In this section, we explain how to derive parameters for the fully homomorphic encryption schemes FV [14] and YASHE [3]. For security, we follow van de Pol and Smart's approach to derive the maximal size of the modulus achievable in a given dimension [35] and consider the distinguishing attack against RLWE. In particular, we use Chen and Nguyen's simulation algorithm for the state-of-the-art lattice basis reduction algorithm BKZ-2.0 [7,8]. For correctness, we provide a lower bound on the modulus in a given dimension and for a targeted number of levels (depending on the application), for both schemes FV and YASHE. Therefore for a given application, it suffices to combine these upper and lower bounds to select a suitable modulus.

3.1 Revisiting van de Pol and Smart's Approach

We assume the reader to be familiar with Schnorr and Euchner's blockwise algorithm BKZ [36], and Chen and Nguyen's improved version BKZ-2.0 [7,8]. We provide more details in the full version [27] of this paper. In the following $\mathsf{BKZ\text{-}2.0}_{N,\beta}$ means that BKZ-2.0 is run with blocksize β and for a maximal number of N rounds. In [35], van de Pol and Smart use the formula of [7,8],

$$\mathsf{cost}(\mathsf{BKZ\text{-}2.0}_{N,\beta}) \leqslant N \times (m-\beta) \times \mathsf{cost}(\text{Enumeration in dimension } \beta) + \mathcal{O}(1) \quad (1)$$

to estimate the cost of BKZ-2.0$_{N,\beta}$ (in terms of the number of nodes visited) on an m-dimensional basis, and to *generate* secure parameters.[2] Instead of using BKZ-2.0 to verify heuristically selected parameters, they rather propose a rational method to tackle the parameter selection, which we describe below.

For a given security parameter λ and a dimension m, van de Pol and Smart propose to derive the smallest root Hermite factor $\gamma(m)$ on an m-dimensional lattice achievable using BKZ-2.0 by an adversary limited to a computational cost of at most $\mathsf{cost}(\text{BKZ-2.0}) \leqslant 2^{\lambda}$. By Equation (1), this means that for all β and N, we need to have

$$N \times (m - \beta) \times \mathsf{cost}(\text{Enumeration in dimension } \beta) \leqslant 2^{\lambda} \ .$$

Thus, for each β and using the enumeration costs in [7] (or [8]), one obtains an upper bound N_{max} on the number of BKZ-2.0 rounds with blocksize β that an adversary bounded as above can afford to run, *i.e.* such that this latter inequality is still verified. Next, the quality of the resulting basis is estimated by running the BKZ-2.0$_{N_{\mathsf{max}},\beta}$ simulation algorithm on a random lattice with blocksize β and N_{max} rounds. This yields a root Hermite factor $\gamma(m,\beta)$ for this specific blocksize β. By taking the minimum value over all blocksizes, one obtains the minimum root Hermite factor $\gamma(m)$ achievable in dimension m for the security parameter λ using BKZ-2.0.

Van de Pol and Smart show that, for the homomorphic evaluation of the AES circuit of [19], by using their new approach for a given security level, it is possible to work with significantly smaller lattice dimensions than what previous methods recommended, which affects the performance of the underlying lattice-based homomorphic encryption scheme.

Limitations of [35]. However, the approach presented in [35] has some limitations. First of all, van de Pol and Smart only consider dimensions that are a power of two. They use linear interpolation for the missing values and therefore obtain a simplified model which does not reflect the real behavior of the minimal root Hermite factor. Also, the enumeration costs used in [35] are based on the proceedings version [7] of the BKZ-2.0 paper. Recently a full version [8] with smaller enumeration costs has been published, which forces one to revisit van de Pol and Smart's results. Last but not least, they only consider blocksizes that are a multiple of 10 (due to the tables in [7]). This leads to a phenomenon of *plateaus* (*cf.* Fig 1) and might lead to a choice of parameters ensuring less than λ bits of security.

Overcoming the Limitations of [35]. To overcome these issues, we performed the same experiments as van de Pol and Smart but for *all* dimensions from 1000

[2] The term $\mathcal{O}(1)$ occurs due to the fact that in high dimension, the enumeration time is usually dominant compared to the time spent on computing the Gram-Schmidt orthogonalization and LLL reduction [7,8]. Note again that Chen and Nguyen provide an *ideal* simulation algorithm – experimental applications of BKZ-2.0 might yield a basis with a larger root Hermite factor. Therefore, using Equation (1) to estimate parameters is conservative.

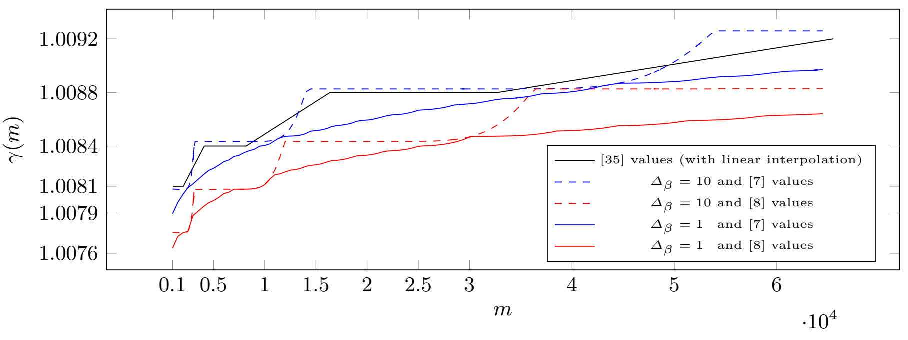

Fig. 1. Minimal root Hermite factor $\gamma(m)$ achievable with a complexity less than 2^{80}, in function of the dimension m

up to 65000. We also considered both the enumeration costs given in Chen and Nguyen's proceedings paper [7] as well as those in the full version [8]. We plotted the results in Fig. 1. As expected, the linear interpolation of [35] does not fully reflect the behavior of the experiments for the other dimensions.

However, when performing the experiments for all dimensions, but only avoiding linear interpolation, we still observe the plateau phenomenon. This can be explained by the fact that the enumeration costs from [7] are only used for blocksizes that are a multiple of $\Delta_\beta = 10$ (which are the only values given in [7]), and only those are considered in [35]. Each plateau consists of the minimal root Hermite factor achievable for a specific blocksize β. Now for the whole plateau, $\mathsf{BKZ\text{-}2.0}_{N_{\mathsf{max},\beta}}$ terminates in less than N_{max} rounds, *i.e.* a fix-point is attained at some round $i < N_{\mathsf{max}}$. The next plateau corresponds to a blocksize $\beta - \Delta_\beta = \beta - 10$. Between plateaus, the number N_{max} is the limiting factor in BKZ-2.0 (*i.e.* $\mathsf{BKZ\text{-}2.0}_{N_{\mathsf{max},\beta}}$ terminates at round N_{max}) and determines the achievable root Hermite factor (therefore this latter value increases until a blocksize of size $\beta - 10$ instead of β is more useful).

To resolve this issue, we used the least squares method to interpolate the enumeration costs for blocksizes β that are not a multiple of 10 (for more details see the full version [27] of this paper). These new costs allowed us to perform the experiments with *all* blocksizes $\beta \in \{100, \ldots, 250\}$ (*i.e.* with steps $\Delta_\beta = 1$) and we obtained the plain lines in Fig. 1.[3] As one can see there, parameters selected from plateaus might yield attacks of complexity smaller than 2^λ if the attacker chooses a blocksize that actually allows to achieve a smaller root Hermite factor.

Therefore, to be more conservative than [35] in our parameter selection, in the rest of the paper, we use the values of $\gamma(m)$ for $\Delta_\beta = 1$ using the enumeration

[3] Note that, without loss of generality, we only considered blocksizes larger than 100. Indeed, for $\beta = 100$ the cost of the enumeration of [8] is 2^{39} and BKZ-2.0 usually reaches a fix point in less than 100 rounds (*cf.* [8, Fig.7]). Therefore for a target security level of 80 bits and dimensions up to $\approx 2^{32}$, one will not be able to obtain a better reduction with a $\beta < 100$.

costs of [8]. Note that there is a significant difference in the achievable values compared to [35].

3.2 Security Requirements for RLWE: The Distinguishing Attack

In this section, we restate and extend the security analysis of [35]. Namely, we consider the distinguishing attack against RLWE (see [32,28]). In the following, we denote by $0 < \epsilon < 1$ the advantage with which we allow the adversary to distinguish an RLWE instance $(a, b = a \cdot s + e) \in R_q^2$ from a uniform random pair $(a, u) \in R_q^2$ (*i.e.* the advantage of the adversary for solving the Decisional-RLWE problem). For any $a \in R_q$, we denote by $\Lambda_q(a)$ the lattice

$$\Lambda_q(a) = \{y \in R_q : \exists\ z \in R, y = a \cdot z \bmod q\}.$$

Recall that, for an n-dimensional lattice Λ, we denote by $\Lambda^\times$ its dual, *i.e.* the lattice defined by $\Lambda^\times = \{v \in \mathbb{R}^n : \forall\ b \in \Lambda, \langle v, b\rangle \in \mathbb{Z}\}$. The distinguishing attack consists in finding a small vector $v \in q \cdot \Lambda_q(a)^\times$. Then, for all $y \in \Lambda_q(a)$, $\langle v, y\rangle = 0 \bmod q$. To distinguish whether a given pair (a, u) was sampled according to the RLWE distribution or the uniform distribution, one tests whether the inner product $\langle v, u\rangle$ is 'close' to 0 modulo q (*i.e.* whether $|\langle v, u\rangle| < q/4$) or not.

Indeed, when u is uniformly distributed in R_q and $n \geqslant 2\lambda + 1$, $\langle v, u\rangle$ is statistically close to the uniform distribution by the leftover hash lemma and the test accepts with probability $1/2 - \mathsf{negl}(\lambda)$. However, when (a, u) is an RLWE sample, *i.e.* there exists $s \in R_q$ and $e \leftarrow \chi_{\text{err}}$ such that $u = a \cdot s + e$, we have $\langle v, u\rangle = \langle v, e\rangle \bmod q$, which is essentially a sample from a Gaussian (reduced modulo q) with standard deviation $\|v\| \cdot \sigma_{\text{err}}$. Now when this parameter is not much larger than q, $\langle v, e\rangle$ can be distinguished from uniform with advantage $\exp(-\pi\tau^2)$ with $\tau = \|v\| \cdot \sigma_{\text{err}}/q$, for details see [32,28].

The distinguishing attack against LWE is more efficient when working with a $m \times n$ matrix with $m > n$ [32,28,35]. Moreover, it is unknown how to exploit the ring structure of RLWE to improve lattice reduction [15,7]. Therefore, we will embed our RLWE instance into an LWE lattice. Next we apply the distinguishing attack against LWE and the result can be used to distinguish the RLWE instance from uniform. Define an LWE matrix $A \in \mathbb{Z}_q^{m\times n}$ associated to a as the matrix whose first n lines are the coefficient vectors of $x^i \cdot a$ for $i = 0, \ldots, n-1$ and the $m - n$ last lines are small linear combinations of the first n lines. Denote the LWE lattice

$$\Lambda_q(A) = \{y \in \mathbb{Z}^m : \exists\ z \in \mathbb{Z}^n, y = Az \bmod q\}.$$

Now, we use lattice basis reduction in order to find such a short vector $v \in q \cdot \Lambda_q(A)^\times$. An optimal use of BKZ-2.0 would allow us to recover a vector v such that $\|v\| = \gamma(m)^m \cdot q^{n/m}$ (because $\det(q\Lambda_q(A)^\times) = q^n$, *cf.* [35]). Therefore, to keep the advantage of the BKZ-2.0-adversary small enough, we need to have $\exp(-\pi\tau^2) \leqslant \epsilon$, *i.e.*

$$\gamma(m)^m \cdot q^{(n/m)-1} \cdot \sigma_{\text{err}} \geqslant \sqrt{-\log(\epsilon)/\pi}\ .$$

Table 1. Maximal values of $\log_2(q)$ to ensure $\lambda = 80$ bits of security, with distinguishing advantage $\epsilon = 2^{-80}$ and standard deviation $\sigma_{\text{err}} = 8$

n	1024	2048	4096	8192	16384
Maximal $\log_2(q)$ (method of [28])	40.6	79.4	157.0	312.2	622.7
Maximal $\log_2(q)$ (our method)	**47.5**	**95.4**	**192.0**	**392.1**	**799.6**

Define $\alpha = \sqrt{-\log(\epsilon)/\pi}$. To ensure security for all $m > n$, we obtain the condition

$$\log_2(q) \leqslant \min_{m>n} \frac{m^2 \cdot \log_2(\gamma(m)) + m \cdot \log_2(\sigma/\alpha)}{m-n} \ . \tag{2}$$

Let us fix the security parameter λ. Following the experiment described in Section 3.1, one can recover the minimal root Hermite factor $\gamma(m)$ for all $m > n$. Therefore, given a target distinguishing advantage ϵ, a dimension n and an error distribution χ_{err}, one can derive the maximal possible value for $\log_2(q)$ using Equation (2). Some interesting values are presented in Table 1. As in [35], it seems that the parameters obtained by using Lindner and Peikert's method [28] are more conservative than those obtained with the BKZ-2.0 simulation.[4]

3.3 Correctness and Noise Growth of YASHE

Any YASHE ciphertext c carries an inherent noise term, which is an element $v \in R$ of minimal norm $\|v\|_\infty$ such that $fc = \Delta[m]_t + v \pmod{q}$. If $\|v\|_\infty$ is small enough, decryption works correctly, which means that it returns the message m modulo t. More precisely, [3, Lemma 1] shows that this is the case if $\|v\|_\infty < (\Delta - r_t(q))/2$. A freshly encrypted ciphertext output by YASHE.Encrypt has an inherent noise term v that can be bounded by $\|v\|_\infty < V = \delta t B_{\text{key}}(2B_{\text{err}} + r_t(q)/2)$, see [3, Lemma 2].

During a homomorphic addition, the inherent noise terms roughly add up such that the resulting noise term is bounded by $\|v_{\text{add}}\|_\infty \leqslant \|v_1\|_\infty + \|v_2\|_\infty + r_t(q)$, where v_1 and v_2 are the respective noise terms in c_1 and c_2. For a multiplication operation, noise growth is much larger. It is shown in [3, Theorem 4 and Lemma 4] that, when $\|v_1\|_\infty, \|v_2\|_\infty < V$ the noise term after multiplication can be bounded by

$$\|v_{\text{mult}}\|_\infty < \delta t(4 + \delta t B_{\text{key}})V + \delta^2 t^2 B_{\text{key}}(B_{\text{key}} + t) + \delta^2 t \ell_{w,q} w B_{\text{err}} B_{\text{key}}.$$

For a homomorphic computation with L levels of multiplications (and considering only the noise growth from multiplications), [3, Corollary 1 and Lemma 9]

[4] In [29], Lin and Nguyen obtained significant improvements upon Lindner-Peikert's decoding attack [28] using only pruned enumeration. However, there is no detail on how to compute the success probability, nor on how to estimate the number of nodes to enumerate, nor on how long an enumeration takes. It remains an interesting open problem to adapt van de Pol and Smart's approach to Lin and Nguyen's attack for parameter selection, as it is currently unclear how to compute the above values.

give an upper bound on the inherent noise in the resulting ciphertext as $\|v\|_\infty < C_1^L V + L C_1^{L-1} C_2$, where

$$C_1 = (1 + 4(\delta t B_{\text{key}})^{-1})\delta^2 t^2 B_{\text{key}}, C_2 = \delta^2 t B_{\text{key}} \left(t(B_{\text{key}} + t) + \ell_{w,q} w B_{\text{err}}\right).$$

In order to choose parameters for YASHE so that the scheme can correctly evaluate such a computation with L multiplicative levels, the parameters need to satisfy $C_1^L V + L C_1^{L-1} C_2 < (\Delta - r_t(q))/2$. In Table 2(a), we provide some values for power-of-two dimensions n and levels $L = 0, 1, 10, 50$.

3.4 Correctness and Noise Growth of FV

We can treat FV and YASHE ciphertexts similarly by simply interchanging $c_0 + c_1 s$ and fc. Thus, for an FV ciphertext (c_0, c_1) the inherent noise term is an element $v \in R$ of minimal norm such that $c_0 + c_1 s = \Delta[m]_t + v \pmod q$. Since decryption is the same once $[c_0 + c_1 s]_q$ or $[fc]_q$ are computed, respectively, this also means that correctness of decryption is given under the same condition $\|v\|_\infty < (\Delta - r_t(q))/2$ in both schemes. In an FV ciphertext, the value $v = e_1 + e_2 s - eu$ satisfies $c_0 + c_1 s = \Delta[m]_t + v \pmod q$ and therefore, we can bound the noise term in a freshly encrypted FV ciphertext by $\|v\|_\infty < V = B_{\text{err}}(1 + 2\delta B_{\text{key}})$.

The same reasoning shows that noise growth during homomorphic addition can be bounded in the same way by $\|v_{\text{add}}\|_\infty \leqslant \|v_1\|_\infty + \|v_2\|_\infty + r_t(q)$. Following the exact same proofs as for YASHE as in [3] (see the proofs for the more practical variant YASHE', which we use here), one can show that the noise growth during a homomorphic multiplication is bounded by

$$\|v_{\text{mult}}\|_\infty < \delta t(4 + \delta B_{\text{key}})V + \delta^2 B_{\text{key}}(B_{\text{key}} + t^2) + \delta \ell_{w,q} w B_{\text{err}},$$

where as before, it is assumed that $\|v_1\|_\infty, \|v_2\|_\infty < V$. Note that the bound on the multiplication noise growth is smaller than the respective bound for YASHE by roughly a factor t. This means that FV is more robust against an increase of the parameter t. Similarly as above, when doing a computation in L levels of multiplications (carried out in a binary tree without taking into account the noise growth for homomorphic additions), the noise growth can be bounded by $\|v\|_\infty < C_1^L V + L C_1^{L-1} C_2$, where

$$C_1 = (1 + \epsilon_2)\delta^2 t B_{\text{key}}, \quad C_2 = \delta^2 B_{\text{key}}(B_{\text{key}} + t^2) + \delta \ell_{w,q} w B_{\text{err}}, \quad \epsilon_2 = 4(\delta B_{\text{key}})^{-1},$$

and the correctness condition for choosing FV parameters for an L-leveled multiplication is $C_1^L V + L C_1^{L-1} C_2 < (\Delta - r_t(q))/2$ as above. In Table 2(b), we provide some values for power-of-two dimensions n and levels $L = 0, 1, 10, 50$; these values illustrate the smaller theoretical noise growth for FV in comparison to YASHE.

4 Practical Implementations

In order to assess the relative practical efficiency of FV and YASHE, we implemented these leveled homomorphic encryption schemes in C++ using the arithmetic library FLINT [23]; our implementations are publicly available at [26].

Table 2. Minimal value of $\log_2(q)$ to ensure correctness of YASHE and FV, with overwhelming probability, using standard deviation $\sigma_{\text{err}} = 8$, plaintext modulus $t = 2$, integer base $w = 2^{32}$, and $B_{\text{key}} = 1$

(a) YASHE

n	1024	2048	4096	8192	16384
$L = 0$	20	21	22	23	24
$L = 1$	62	64	66	68	70
$L = 10$	265	286	306	326	346
$L = 50$	1150	1250	1350	1450	1550

(b) FV

n	1024	2048	4096	8192	16384
$L = 0$	19	20	21	22	23
$L = 1$	40	43	46	49	52
$L = 10$	229	250	271	292	313
$L = 50$	1069	1170	1271	1372	1473

Table 3. Timings of YASHE and FV using the same parameters as in [3]: $R = \mathbb{Z}[x]/(x^{4096} + 1)$, $q = 2^{127} - 1$, $w = 2^{32}$, $t = 2^{10}$ on an Intel Core i7-2600 at 3.4 GHz with hyper-threading turned off and over-clocking ('turbo boost') disabled

Scheme	KeyGen	Encrypt	Add	Mult	KeySwitch or ReLin	Decrypt
YASHE	3.4s	16ms	0.7ms	18ms	31ms	15ms
FV	0.2s	34ms	1.4ms	59ms	89ms	16ms
YASHE [3] (estimation)	–	23ms	0.020ms	27ms		4.3ms

Timings. In Table 3, we provide timings using the same parameters as in [3]. As expected from the structure of the ciphertexts, it takes twice more time to Encrypt or Add using FV compared to YASHE and three times longer to multiply two ciphertexts. These parameters also allow us to provide *estimated* timings for the implementation of [3] on the same architecture as an illustration of a possible overhead in performances due to the arithmetic libraries (namely, FLINT) and the C++ wrappers.[5] This corroborates the significant performance gains obtained in recent works in lattice-based cryptography [22,13] using home-made implementations, instead of relying on arithmetic libraries [19,1].

Practical Noise Growth. In Sections 3.3 and 3.4, we provide strict theoretical upper bounds on the noise growth during homomorphic operations in FV and YASHE to ensure correctness with *overwhelming probability.* In practice however, one expects a smaller noise growth on average and one could choose smaller bounds ensuring correctness with high probability only. This yields a huge gain in performance (allowing to reduce q, and thus n) while still ensuring correctness most of the time. In Figure 2, we depict an average noise growth for levels 0 to 10 for FV and YASHE. For example, this figure shows that the real noise growth allows one to reduce the bit size of q by nearly 33% to handle more than 10 levels. Therefore, for optimal performances in practice, one should select a modulus q as small as possible while still ensuring correctness with high probability.

[5] Both Intel processors have hyper-threading turned off and over-clocking ('turbo boost') disabled; thus timings were estimated proportionally to the processor speeds of the computers (3.4 GHz versus 2.9 GHz).

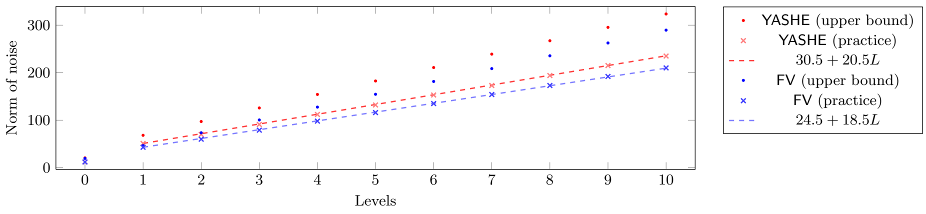

Fig. 2. Evolution of the norm of the noise using a standard deviation $\sigma = 8$, a plaintext modulus $t = 2$, a word $w = 2^{32}$, and $B_{key} = 1$, $R = \mathbb{Z}[x]/(x^{8192} + 1)$, and q a 392-bit prime

4.1 Homomorphic Evaluation of SIMON

The SIMON Feistel Cipher. In June 2013, the U.S. National Security Agency (NSA) unveiled SIMON, a family of lightweight block ciphers. These block ciphers were designed to provide an optimal hardware performance. SIMON has a classical Feistel structure with the round block size of $2n$ bits. For performance reasons, in what follows we focus on SIMON-32/64 having a block size of $2n = 32$ bits, a 64-bit secret key and $N_r = 32$ rounds. At round i, SIMON operates on the left n-bit half $\mathbf{x}_i$ of the block $(\mathbf{x}_i, \mathbf{y}_i)$ and applies a non-linear, non-bijective function $F \colon \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ to it. The output of F is XORed with the right half along with a round key $\mathbf{k}_i$ and the two halves are swapped. The function F is defined as $F(\mathbf{x}) = ((\mathbf{x} \lll 8) \otimes (\mathbf{x} \lll 1)) \oplus (\mathbf{x} \lll 2)$ where $(\mathbf{x} \lll j)$ denotes left rotation of $\mathbf{x}$ by j positions and $\otimes$ is binary AND. The round keys $\mathbf{k}_i$ are very easily derived from a master key k with shifts and XORs. Details on how these subkeys are generated can be found in [2].

Homomorphic Representation of the State. As in [9,10] for AES, we encrypt the SIMON state state-wise. More precisely, the left half $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$ and the right half $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{F}_2^n$ of the SIMON state are encrypted as a set of $2n$ ciphertexts $c_1, \ldots, c_n, c_{n+1}, \ldots, c_{2n}$. For each $1 \leqslant j \leqslant n$, c_j encrypts $x_j \in \mathbb{F}_2$ and c_{n+j} encrypts y_j. In other words, the $2n$ bits of the SIMON state are represented in $2n$ different ciphertexts. Note that the use of batching[6] with ℓ slots allows one to perform ℓ SIMON evaluations in parallel by encoding the corresponding bit of the state of the i-th SIMON plaintext into the i-th slot.

Homomorphic Operations. This state-wise encrypted representation of the steps allows to do the SIMON evaluation easily. Swapping the halves consists in modifying the index of the encrypted state $c_j \leftrightarrow c_{n+j}$. Define encryptions e_{ij} of the bits

[6] To evaluate a Boolean circuit, one can select $t = 2$ and encode each plaintext bit as the constant coefficient of a plaintext polynomial. However, if one uses batching with ℓ slots, where each ciphertext can represent a number of ℓ independent plaintexts, one obtains a significant gain in the use of both space and computational resources. Batching was adapted to the BGV scheme in [18], and can be made compatible with both FV and YASHE.

Table 4. Homomorphic Evaluations of SIMON-32/64 on a 4-core computer (Intel Core i7-2600 at 3.4 GHz)

Scheme	Parameter set	λ	$\ell = \#$ of slots	Keygen	Encrypt State	SIMON Evaluation	Relative time	Norm of Noise Final	Norm of Noise Maximal
FV	Ib	64	2	4 s	7 s	526 s	263 s	509	516
YASHE	Ia	64	1	64 s	4 s	**200 s**	200 s	561	569
YASHE (1 core)	Ia	64	1	64 s	14 s	**747 s**	747 s	557	569
FV	II	> 80	1800	24 s	209 s	3062 s	1.70 s	918	1024
YASHE	II	> 80	1800	1300 s	104 s	1029 s	**0.57 s**	949	1024
SIBDGHV [10]	–	64	199	1032 s	1 s	628 s	3.15 s	650	704

	λ	d	$n = \phi(d)$	# of slots	$\log_2(q)$	$\log_2(w)$	σ	B_{key}
Set-Ia	64	10501	10500	1	570	70	8	1
Set-Ib	64	9551	9550	2	517	65	8	1
Set-II	> 80	32767	27000	1800	1025	257	8	1

k_{ij} of the round keys $\mathbf{k}_i$, for all i, j. (When using batching, one encrypts the vector $(k_{ij}, \ldots, k_{ij}) \in \{0,1\}^\ell$.) This simple representation allows to XOR the right half of the state with the key via n homomorphic additions $c_{n+j} \leftarrow c_{n+j} + e_{ij}$. A shift of a positions as used in the function F is obtained by some index swapping $c_{(i+a) \bmod n}$. Finally, the only AND operation in the function F is obtained by n homomorphic multiplications. Therefore to obtain an encrypted state $c'_1, \ldots, c'_{2n}$ from an encrypted state $c_1, \ldots, c_{2n}$, one can perform:

$$c'_{n+j} \leftarrow c_j; \quad c'_j = (c_{(j+8) \bmod n} \cdot c_{(j+1) \bmod n}) + c_{(j+2) \bmod n} + e_{ij} \ .$$

Practical Results. We homomorphically evaluated SIMON-32/64 using our C++ implementations of FV and YASHE (and also the implementation of [10] for the leveled homomorphic encryption scheme over the integers).[7]

Results are provided in Table 4. Note that we selected parameters ensuring as many bits of security for the homomorphic encryption schemes as the number of bits of the SIMON key.[8]

4.2 Some Thoughts about Homomorphic Evaluations

Let us define the two notions latency and throughput associated to a homomorphic evaluation. We say that the *latency* of a homomorphic evaluation is the time required to perform the entire homomorphic evaluation. Its *throughput* is the number of blocks processed per unit of time.

The results presented in Table 4 emphasize an important point: different parameter sets can be selected, either to minimize the latency (Set-Ia and Set-Ib), or to maximize the throughput (Set-II). In [10] and [19,9], the parameters were

[7] Since each round of SIMON consists of one homomorphic multiplication, the leveled homomorphic encryption schemes need to handle at least as many levels as the number of rounds.

[8] Parameter Set-II ensures more than 80 bits of security (more likely around 120 bits) but the smaller the modulus q, the faster is the computation.

selected to maximize the throughput using batching, and therefore claim a small *relative time per block* – the latency however is several dozens of hours. However, 'real world' homomorphic evaluations (likely to be used in the cloud) should be implemented in a transparent and user-friendly way. It is therefore questionable whether the batching technique (to achieve larger throughput in treating blocks) is suitable for further processing of data. In particular, it might only be suitable when this processing is identical over each block (which is likely *not* to be the case in real world scenarios). Overall, one should rather select parameters to have the latency as small as possible. The throughput can be increased by running the homomorphic evaluations in a cluster.

5 Conclusion

In this work, we revisited van de Pol and Smart's approach to tackle parameter selection for lattice-based cryptosystems. We also conducted both a theoretical and practical comparison of FV and YASHE. We obtained that the noise growth is smaller in FV than in YASHE (both theoretically and practically). Conversely, we obtained that YASHE is, as expected, faster than FV. As a side result, for high performances, it seems interesting to implement all building blocks of the schemes rather than to rely on external arithmetic libraries.

Next, we homomorphically evaluated the lightweight block cipher SIMON, and discussed the notions of throughput and latency. We obtain that SIMON-32/64 can be evaluated *completely* in about 12 minutes on a single core and in about 3 minutes on 4 cores using OpenMP (when optimizing latency). If several blocks are processed in parallel, SIMON-32/64 can be evaluated in about 500ms per block (and less than 20 minutes total); and these timings can be lowered by using additional cores.

Finally, note that our results can certainly be improved further by other optimizations. One could incorporate dynamic scaling during the computation as discussed in [14] such that it is ensured that ciphertexts maintain their minimal size. Another possible variant is to use the Chinese Remainder Theorem to pack each half of the SIMON state into one single ciphertext instead of spreading it out over n ciphertexts. Operations that need to move data between different plaintext slots can be realized by Galois automorphisms as explained in [18]. This can possibly be further combined with batching of several SIMON states into one ciphertext. To explore the application of these to both schemes and possibly further optimizations for realizing a fully home-made and fully optimized implementation of a homomorphic SIMON evaluation is left as future work.

Acknowledgments. We thank the Africacrypt 2013 referees for their interesting reviews, and Frederik Vercauteren for insightful comments on batching.

References

1. Bansarkhani, R.E., Buchmann, J.: Improvement and efficient implementation of a lattice-based signature scheme. Cryptology ePrint Archive, Report 2013/297 (2013), http://eprint.iacr.org/
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013), http://eprint.iacr.org/
3. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Stam (ed.) [37], pp. 45–64
4. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012)
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS, pp. 309–325. ACM (2012)
6. Canetti, R., Garay, J.A. (eds.): CRYPTO 2013, Part I. LNCS, vol. 8042. Springer, Heidelberg (2013)
7. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
8. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates (2013), http://www.di.ens.fr/~ychen/research/Full_BKZ.pdf
9. Cheon, J.H., Coron, J.-S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 315–335. Springer, Heidelberg (2013)
10. Coron, J.-S., Lepoint, T., Tibouchi, M.: Scale-invariant fully homomorphic encryption over the integers. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 311–328. Springer, Heidelberg (2014)
11. Coron, J.S., Naccache, D., Tibouchi, M.: Public key compression and modulus switching for fully homomorphic encryption over the integers. In: Pointcheval, Johansson (eds.) [34], pp. 446–464
12. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
13. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Canetti, Garay (eds.) [6], pp. 40–56
14. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive 2012, 144 (2012)
15. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)
16. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) STOC, pp. 169–178. ACM (2009)
17. Gentry, C., Halevi, S.: Implementing gentry's fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)
18. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, Johansson (eds.) [34] , pp. 465–482

19. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
20. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, Garay (eds.) [6], pp. 75–92
21. Graepel, T., Lauter, K., Naehrig, M.: ML confidential: Machine learning on encrypted data. In: Kwon, T., Lee, M.-K., Kwon, D. (eds.) ICISC 2012. LNCS, vol. 7839, pp. 1–21. Springer, Heidelberg (2013)
22. Güneysu, T., Oder, T., Pöppelmann, T., Schwabe, P.: Software speed records for lattice-based signatures. In: Gaborit, P. (ed.) PQCrypto 2013. LNCS, vol. 7932, pp. 67–82. Springer, Heidelberg (2013)
23. Hart, W.: et al.: Fast Library for Number Theory, Version 2.4 (2013), `http://www.flintlib.org`
24. Lauter, K.E.: Practical applications of homomorphic encryption. In: Yu, T., Capkun, S., Kamara, S. (eds.) CCSW, pp. 57–58. ACM (2012)
25. Lenstra, A.K., Jr Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. 261(4), 515–534 (1982)
26. Lepoint, T.: A proof-of-concept implementation of the homomorphic evaluation of SIMON using FV and YASHE leveled homomorphic cryptosystems (2014), `https://github.com/tlepoint/homomorphic-simon`
27. Lepoint, T., Naehrig, M.: A comparison of the homomorphic encryption schemes FV and YASHE (full version). Cryptology ePrint Archive, Report 2014/062 (2014), `http://eprint.iacr.org/`
28. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
29. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 293–309. Springer, Heidelberg (2013)
30. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC, pp. 1219–1234 (2012)
31. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)
32. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer, Heidelberg (2009)
33. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Cachin, C., Ristenpart, T. (eds.) CCSW, pp. 113–124. ACM (2011)
34. Pointcheval, D., Johansson, T. (eds.): EUROCRYPT 2012. LNCS, vol. 7237. Springer, Heidelberg (2012)
35. van de Pol, J., Smart, N.P.: Estimating key sizes for high dimensional lattice-based systems. In: Stam (ed.) [37], pp. 290–303
36. Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Math. Program. 66, 181–199 (1994)
37. Stam, M. (ed.): IMACC 2013. LNCS, vol. 8308. Springer, Heidelberg (2013)
38. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 27–47. Springer, Heidelberg (2011)