Large FHE Gates from Tensored Homomorphic Accumulator

Guillaume Bonnoron[1], Léo Ducas[2], and Max Fillinger[2]

[1] Chair of Naval Cyber Defense** & Lab-STICC/CID/IRIS, France
[2] CWI, Amsterdam, The Netherlands

Abstract. The main bottleneck of all known Fully Homomorphic Encryption schemes lies in the bootstrapping procedure invented by Gentry (STOC'09). The cost of this procedure can be mitigated either using Homomorphic SIMD techniques, or by performing larger computation per bootstrapping procedure.
In this work, we propose new techniques allowing to perform more operations per bootstrapping in FHEW-type schemes (EUROCRYPT'13). While maintaining the quasi-quadratic $\tilde{O}(n^2)$ complexity of the whole cycle, our new scheme allows to evaluate gates with $\Omega(\log n)$ input bits, which constitutes a quasi-linear speed-up. Our scheme is also very well adapted to large threshold gates, natively admitting up to $\Omega(n)$ inputs. This could be helpful for homomorphic evaluation of neural networks.
Our theoretical contribution is backed by a preliminary prototype implementation, which can perform 6-to-6 bit gates in less than 10 seconds on a single core, as well as threshold gates over 63 input bits even faster.

Keywords: Fully Homomorphic Encryption, Large Gates, Threshold Gates, Ideal lattices.

1 Introduction

Since the first scheme of Gentry [16, 15] a lot of effort has been made to push Fully Homomorphic Encryption (FHE) toward practicality. A first line of research followed the initial approach of Gentry, by bootstrapping FHE from a Somewhat Homomorphic Encryption (SHE) scheme supporting arbitrary circuits of bounded depth. This bootstrapping step consists in homomorphically evaluating the decryption procedure, to refresh ciphertexts. After successive theoretical and practical improvements [31, 8, 21], this bootstrapping procedure has been made feasible in practice, but remains quite expensive, taking several minutes on a single core. Fortunately, this cost can be mitigated thanks to SIMD techniques, allowing to perform the same homomorphic computation on several data sets for the price of one.

A second line of FHE schemes arose from the SHE scheme of Gentry-Sahai-Waters [19]. This SHE scheme supports a different class of functions, including branching programs, and this was also proved sufficient to bootstrap it to FHE via Barrington's theorem [5]. Interestingly, this approach theoretically allows obtaining FHE from a weaker version of the LWE assumption (namely the approximation factor decreases from super-polynomial to polynomial). On the efficiency front, Alperin-Sheriff and Peikert [3] showed how to avoid the costly use of Barrington's transformation by implementing the homomorphic decryption procedure more directly. Then, Ducas and Micciancio [13] adapted the construction to the ring-setting. Providing parameters and implementation,

** Funded and supported by Ecole Navale, IMT Atlantique, Naval Group and Thales

they demonstrated this approach to be feasible with a proof of concept scheme (FHEW): the bootstrapping procedure could be run in under a second on a single core. While their parameters allow one binary gate per bootstrapping, they noted it should be possible in principle to perform slightly larger gates, such as the add-with-carry gate (3-inputs, 2-outputs). This idea was implemented in [6].

Further improvements and generalization were proposed by Chillotti, Gama, Georgieva, and Izabachène [10], leading to a scheme named TFHE. In particular they contributed two improvements of the bootstrapping step, accelerating it by a polylog factor. In practice, this leads to bootstrapping in less than 0.1 second, allowing the same bootstrapped gates as in FHEW [13].

FHE from Homomorphic Accumulator. The core idea in FHE schemes from this second line is to tailor the SHE scheme precisely to the decryption procedure. Namely, the decryption procedure of an LWE ciphertext $\mathbf{c} = (\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$ under key $\mathbf{s} \in \mathbb{Z}_q^n$ for a plaintext space $\mathbb{Z}_t$ is given by:

$$m = \lfloor t(b - \langle \mathbf{a}, \mathbf{s} \rangle)/q \rceil \bmod t \in \mathbb{Z}_t.$$

Given the ciphertext $\mathbf{c}$, this procedure can be split into a $\mathbb{Z}_q$-linear step $L_\mathbf{c} : \mathbf{s} \mapsto b - \langle \mathbf{a}, \mathbf{s} \rangle$, followed by a non-linear function $N : \mathbb{Z}_q \to \mathbb{Z}_t$. Note that one can embed an arbitrary post-decryption transformation $f : \mathbb{Z}_t \mapsto \mathbb{Z}_t$ by setting $N_f : x \mapsto f(\lfloor tx/q \rceil \bmod t)$.

Assume that we have an SHE scheme that precisely supports the class of functions that can be written as $N_f \circ L_\mathbf{c}$ (a notion formalized as a homomorphic accumulator in [13]), and such that the output is again an LWE ciphertext. Then, taking $t = 4$ one can construct an FHE scheme, performing any binary gate g over encryptions of bits (b_1, b_2) for each bootstrap operation. Indeed, using the linearity of LWE ciphertexts, one can compute an encryption of $m = b_1 + 2b_2$, and construct the appropriate function f such that $f(m) = g(b_1, b_2)$.

In more detail, messages m are encoded as powers of a q-th root of unity X^m. With such an encoding, the linear step $L_\mathbf{c}$ is performed by sequential ciphertext multiplications. The non-linear part N_f is performed by computing a subset-sum of the coefficients of the polynomial $E = X^m = \sum e_i X^i$, by exploiting the identity $f(m) = \sum f(i) e_i$.

As the useful computation is provided by the function $f : \mathbb{Z}_t \to \mathbb{Z}_t$, a larger plaintext modulus t allows to perform more computation between each bootstrap operation. Namely, one can build arbitrary k-bit to 1-bit gates if $t \geq 2^k$, and, if we restrict to certain classes of gates, even larger ones (e.g. threshold gates only require $t \geq k + 1$). For most k-to-1 bit functions, this corresponds to a speed up of $\Omega(2^k / \log k) = \tilde{O}(t)$, according to the classical circuit lower-bound of Riordan and Shannon [29]. It is therefore worth increasing the size of the plaintext modulus t in order to perform much more computations per bootstrap operation.

Parameter constraints and efficiency. In the set-up of [13, 6, 10], the constraints for correctness impose asymptotically that $t \leq O(q/n)$.[3] Taking $q = \Theta(n)$, this gives a quasi-quadratic runtime for the whole process, but allows quite small plaintext size: $t \leq O(1)$. In practice, this t cannot be made much larger than 4, maybe up to 6 as done in [6].

Looking more precisely at the complexity of each step, we note an imbalance between the cost of the linear and non-linear steps. Indeed, the linear part requires $\tilde{\Theta}(n)$ operations over $\mathcal{R}_q$, while the non-linear part requires only $\Theta(\log n)$ such operations.

[3] More precisely, $t \leq q/\sqrt{n \cdot \log 1/p_{\text{fail}}}$, where p_{fail} is the failure probability. In this paper, we will always aim for exponentially small failure probability.

This work. We aim to improve the performance of this line of FHE schemes by increasing the plaintext modulus t. Having remarked the imbalance of the costs of the linear and non-linear steps, we proceed to increase the cost of the non-linear step while maintaining the overall quasi-quadratic complexity.

Our approach consists in choosing a ciphertext modulus of the form pq for co-primes p, q, and to perform the linear-step $L_{\mathbf{c}}$ in a CRT fashion. During this linear-step, our SHE scheme only works with the rings $\mathcal{R}_p = \mathbb{Z}[X]/(X^p-1)$ and $\mathcal{R}_q = \mathbb{Z}[Y]/(Y^q-1)$ separately, for a cost of $\tilde{O}(n(p+q))$. Then we proceed to a CRT reconstruction by tensoring the two rings: $\mathcal{R}_p \otimes \mathcal{R}_q \simeq \mathcal{R}_{pq} = \mathbb{Z}[Z]/(Z^{pq}-1)$, noting that $X^a \otimes Y^b = Z^{aq+pb \bmod pq}$. This raises the cost of the non-linear part to $\tilde{\Theta}(pq)$. Setting $p, q = \Theta(n)$ we maintain the quasi-quadratic complexity, but reach a larger plaintext-modulus $t = \Theta(n)$. This is somehow a reminiscence of the approach of [3], adapted to the ring-setting.

One (not so) novel technical aspect is that we choose in this work to use convolution rings $\mathbb{Z}[X]/(X^p-1)$, as in the NTRU schemes [23] rather than cyclotomic ones. The reason is that we need to use some non-power of 2 roots of unity to ensure co-primality of p and q. Indeed, if (say) p is prime, the fact that $X^{p-1} = -1 - X - \cdots - X^{p-2}$ in the p-th cyclotomic ring $\mathbb{Z}(X)/(\Phi_p(X))$ makes the non-linear step described above quite problematic.[4] Yet, we show that the switch to convolution rings can be done without affecting security, by formalizing what we call the *NTRU trick*.[5] More precisely, an appropriately defined version of Ring-LWE over convolution rings is as secure as the usual cyclotomic version of Ring-LWE from [24].

Our work also relies one of the improvements of [10], namely, the use of an "external multiplication" $\mathsf{GSW} \times \mathsf{LWE} \to \mathsf{LWE}$ replacing the $\mathsf{GSW} \times \mathsf{GSW} \to \mathsf{GSW}$ operation used in [19, 3, 13, 6], which saves a log factor on time and memory. It turns out that the second trick of [10], using a mux-gate, is not compatible with our circulant ring set-up, but we instead propose to exploit the Galois action[6] for a similar logarithmic speed-up.

In addition, we propose to use an alternative Gadget matrices based on the Chinese Remainder Theorem, an idea already presented in [20] for different purposes. We show that such gadgets permit a logarithmic speed-up when dealing with gadget inversions of tensored ciphertexts; this contribution may find theoretical and practical applications in other contexts.

To summarize our theoretical construction, we provide schematics in Figure 1, omitting some extra tweaks for practical efficiency that are deferred to Appendix B. We hope this overview may guide the reader through our paper.

Circular security. We recall that all the FHE literature, including our work, relies on (sometimes implicit) circular-security assumptions [15], that may be different from one scheme to the next. Understanding those assumptions is arguably the most important theoretical question in this field.

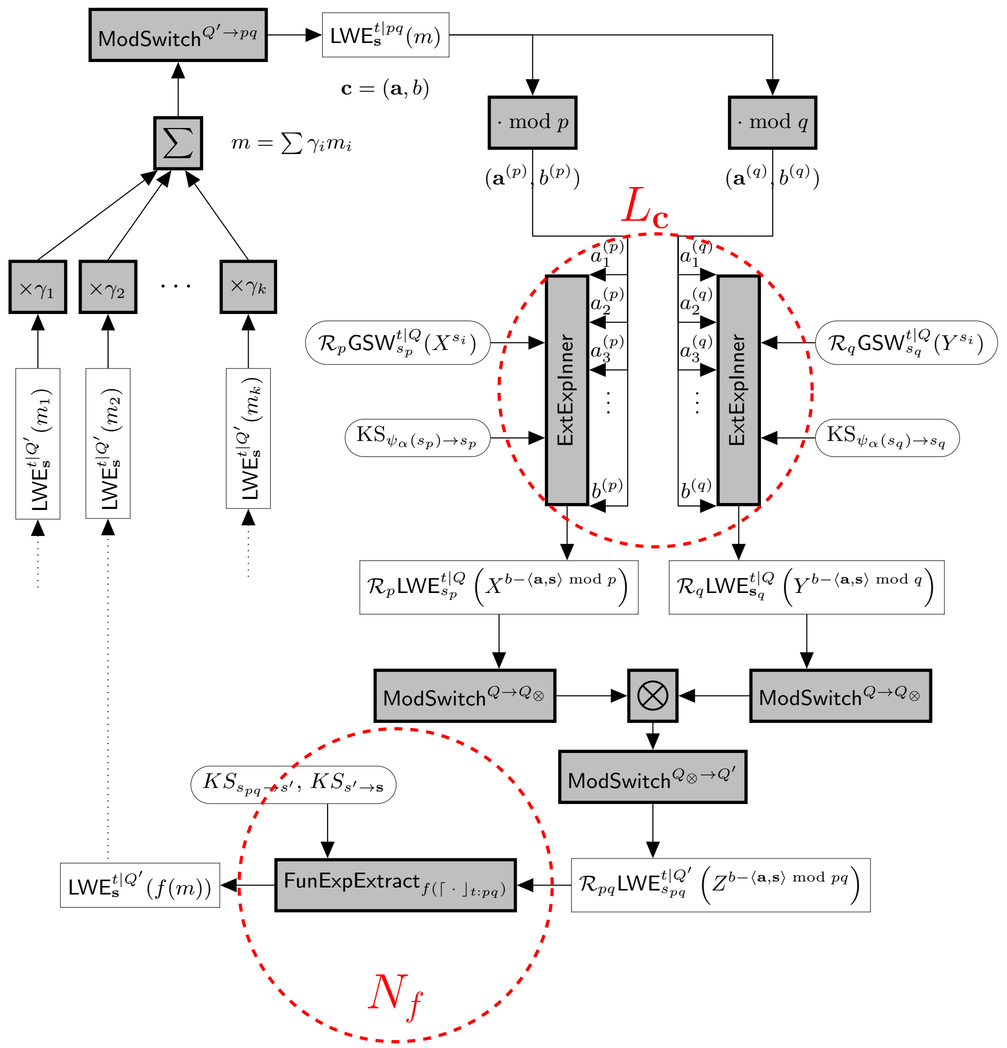
One particular property of our scheme is that this circular security assumption can not be avoided even when relaxing the scheme to a leveled FHE scheme [15]. Indeed, the careful reader may notice that "External Inner-product in the Exponent" step ($\mathsf{ExtExpInner}$, Section 4.3) requires circular encryption.

[4] And maybe even impossible due to dimensionality constraints.

[5] We wish to clarify that our scheme does not require the NTRU assumption, namely the assumption that $f/g \bmod q$ is indistinguishable from random even for small f and g. Up to the usual circular-security assumption, our scheme is based on a ring-LWE type of assumption.

[6] This idea was originally suggested by Daniele Micciancio.

Fig. 1. Scheme overview.



$\mathsf{ModSwitch}^{Q'\to pq}$
$\mathsf{LWE}_{\mathbf{s}}^{t|pq}(m)$
$\mathbf{c} = (\mathbf{a}, b)$
$\cdot \bmod p$
$\cdot \bmod q$
$(\mathbf{a}^{(p)}, b^{(p)})$
$(\mathbf{a}^{(q)}, b^{(q)})$
$\sum$
$m = \sum \gamma_i m_i$
$L_{\mathbf{c}}$
$\times\gamma_1$
$\times\gamma_2$
$\cdots$
$\times\gamma_k$
$a_1^{(p)}$ $a_2^{(p)}$ $a_3^{(p)}$ $b^{(p)}$
$a_1^{(q)}$ $a_2^{(q)}$ $a_3^{(q)}$ $b^{(q)}$
$\mathcal{R}_p\mathsf{GSW}_{s_p}^{t|Q}(X^{s_i})$
$\mathcal{R}_q\mathsf{GSW}_{s_q}^{t|Q}(Y^{s_i})$
$\mathrm{KS}_{\psi_\alpha(s_p)\to s_p}$
$\mathrm{KS}_{\psi_\alpha(s_q)\to s_q}$
ExtExpInner
ExtExpInner
$\mathsf{LWE}_{\mathbf{s}}^{t|Q'}(m_1)$
$\mathsf{LWE}_{\mathbf{s}}^{t|Q'}(m_2)$
$\mathsf{LWE}_{\mathbf{s}}^{t|Q'}(m_k)$
$\mathcal{R}_p\mathsf{LWE}_{s_p}^{t|Q}\left(X^{b-\langle\mathbf{a},\mathbf{s}\rangle \bmod p}\right)$
$\mathcal{R}_q\mathsf{LWE}_{\mathbf{s}_q}^{t|Q}\left(Y^{b-\langle\mathbf{a},\mathbf{s}\rangle \bmod q}\right)$
$\mathsf{ModSwitch}^{Q\to Q_\otimes}$
$\otimes$
$\mathsf{ModSwitch}^{Q\to Q_\otimes}$
$\mathsf{ModSwitch}^{Q_\otimes\to Q'}$
$KS_{s_{pq}\to s'}, KS_{s'\to \mathbf{s}}$
$\mathsf{LWE}_{\mathbf{s}}^{t|Q'}(f(m))$
$\mathsf{FunExpExtract}_{f(\lceil\cdot\rfloor_{t:pq})}$
$\mathcal{R}_{pq}\mathsf{LWE}_{s_{pq}}^{t|Q'}\left(Z^{b-\langle\mathbf{a},\mathbf{s}\rangle \bmod pq}\right)$
N_f

On the left side, there are the k input bits $m_1, \cdots, m_k$ that get combined into $m \in \mathbb{Z}_t$. On the right side we have the two Homomorphic Accumulator $\mathsf{ExtExpInner}$, which perform the linear part $L_{\mathbf{c}}$ of the bootstrapped computation in a CRT fashion. After tensoring it is fed to the non-linear part of the computation $N_f : x \mapsto f(\lfloor tx/q \rceil \bmod t)$, *i.e.* $\mathsf{FunExpExtract}$, where f is the function to be homomorphically evaluated. The computation is intrinsically done with the bootstrapping process, so the final output can directly be used as input.

Grey boxes represent operations, white square boxes represent ciphertexts, and rounded white boxes represent key material. The linear step $L_{\mathbf{c}}$ and the non-linear step N_f discussed in introduction are highlighted by dashed red circles.

Instantiation and implementation. To attest to the feasibility of our approach, we also provide an instantiation supporting[7] 6-to-6 bit gates, at a security level of about 100 bits. Its current implementation runs this 6-to-6 bits bootstrapped gate in about 10 seconds.

Related work. Recently Chillotti et al. [11] also proposed the construction of large homomorphic gates, using a quite different approach. They claim impressive performances, such as a 16-to-8 bit homomorphic gate running in about 2 seconds. Admittedly, our current implementation is significantly slower. One case for which our approach might be advantageous is the case of threshold gates: for example, our 6-bit scheme can natively support threshold gates over 63 input bits with the same performance.

Impact. Our implementation should certainly not be understood as publicity for the practical efficiency of this overall design. It nevertheless serves the purpose of demonstrating that our new building blocks can be used inside a reasonable scheme. It is therefore plausible that our contributions are not only of theoretical interest, but may as well find some use in future practical FHE designs.

Plan. We begin in Section 2 with preliminary results and notations. Then we introduce the underlying encryption schemes at hand in Section 3. Section 4 presents in detail the building blocks of the gate, leading to the overall description in Section 5. Finally Section 6 reports implementation details and performances.git

In addition, Appendix A provides the hardness proof for Circulant-LWE and the cpa-security proofs of the associated cryptosystems. Appendix B provides several useful optimization of our scheme for its concrete efficiency.

Acknowledgments

The authors are grateful to Ana Costache, Daniele Micciancio and Jessica Sorrell for helpful conversations. Most of this work was done when the first author was visiting at CWI, with the support of the CWI internship program. The second author was supported by a Veni Innovational Research Grant from NWO under project number 639.021.645. The third author was supported by the NWO Free Competition grant 617.001.203.

2 Preliminaries

2.1 Subgaussian Random Variables

Definition 1. *We say that a real random variable X is* subgaussian with parameter δ *(or δ-*subgaussian*) if $\mathbb{E}[X] = 0$, and for all t, $\mathbb{E}[\exp(tX)] \leq \exp\big(t^2\delta^2/2\big)$.*

Subgaussian random variables have the following well known properties (see [32] and [30]):

Theorem 1. *Let X_1 and X_2 be subgaussian random variables with parameters δ_1 and δ_2, respectively.*

- *$X_1 + X_2$ is $(\delta_1 + \delta_2)$-subgaussian.*

[7] `https://github.com/gbonnoron/Borogrove`

– *If X_1 and X_2 are independent, $X_1 + X_2$ is $\sqrt{\delta_1^2 + \delta_2^2}$-subgaussian.*
– *aX_1 is $(|a|\delta_1)$-subgaussian.*
– *Subgaussian tail estimate: $P(|X_1| \geq \sqrt{2\lambda}\delta_1) \leq 2\exp(-\lambda)$.*

Note that [32] also defines non-centered subgaussian variables. However, in this work, we only consider centered ones, ie. with $\mathbb{E}[X] = 0$.

2.2 Rings

Our FHE scheme uses *circulant convolution rings* (or, for short, *circulant rings*). Circulant rings of degree d will be denoted with indeterminate T: $\mathcal{R}_d = \mathbb{Z}[T]/(T^d - 1)$. We fix two distinct odd primes p and q. When speaking specifically of rings $\mathcal{R}_p$, $\mathcal{R}_q$, and $\mathcal{R}_{pq}$ we shall use indeterminates X, Y and Z, respectively. We write $\tilde{\mathcal{R}}_d$ for the cyclotomic ring $\mathbb{Z}[\tilde{T}]/\Phi_d(\tilde{T})$ where $\Phi_d(\tilde{T})$ is the d-th cyclotomic polynomial; if d is prime, $\Phi_d(\tilde{T}) = 1 + \tilde{T} + \tilde{T}^2 + \ldots + \tilde{T}^{d-1}$. We identify a ring element $a \in \mathcal{R}_d$ with its lowest degree representative $a_0 + a_1 T + \ldots + a_{d-1}T^{d-1} \in \mathbb{Z}[T]$ and call $a_0, \ldots, a_{d-1}$ the coefficients of a. We identify $a \in \mathcal{R}_d/Q\mathcal{R}_d$ with its lowest degree representative with coefficients $a_0, \ldots, a_{d-1} \in [-Q/2, Q/2)$. We define the following norms for ring elements:

Definition 2. *Let $a \in \mathcal{R}_d$ (or $\mathcal{R}_d/Q\mathcal{R}_d$). We define the* coefficient norm *of a as $\|a\| = \|(a_0, \ldots, a_{d-1})\| = \sqrt{\sum_{i=0}^{d-1} a_i^2}$.*

Definition 3. *Let $a \in \mathcal{R}_d$ (or $\mathcal{R}_d/Q\mathcal{R}_d$). We define the* operator norm *of a as $|a| = \max_{b \in \mathcal{R}\setminus\{0\}} \|ab\|/\|b\|$. We expand this notion to vectors $\mathbf{x} \in \mathcal{R}_d^n$ by maximizing $\mathbf{y}$ over $\mathcal{R}^n \setminus \{0\}$ and replacing the multiplication with the inner product over $\mathcal{R}_d$.*

Definition 4. *We define the (normalized) trace function[8] as follows: We let $\mathrm{Tr}^*_{\mathcal{R}_d/\mathbb{Z}} : \mathcal{R}_d \to \mathbb{Z}, a \mapsto a_0$. If d is clear from context, we simply write this function as Tr^*. We let $\mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathcal{R}_p} : \mathcal{R}_{pq} \to \mathcal{R}_p$ be the linear function defined by*

$$\mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathcal{R}_p}(Z^k) = \begin{cases} X^{k/q} & \text{if } q|k \\ 0 & \text{otherwise} \end{cases}$$

The following property is easy to see:

Lemma 1. *$\mathrm{Tr}^*_{\mathcal{R}_d/\mathbb{Z}}$ and $\mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathcal{R}_p}$ are linear, and $\mathrm{Tr}^*_{\mathcal{R}_p/\mathbb{Z}} \circ \mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathcal{R}_p} = \mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathbb{Z}}$.*

Definition 5. *A random variable $A \in \mathcal{R}_d$ is δ-subgaussian if, for every $b \in \mathcal{R} \setminus \{0\}$, $\mathrm{Tr}^*(Ab)/\|b\|$ is δ-subgaussian.*

In Section 4, we need to bound the subgaussinity parameter of $e + \langle \mathbf{d}, \mathbf{e} \rangle$ where e and the components of $\mathbf{e}$ are independent γ- and δ-subgaussian variables over $\mathcal{R}_d$, and $\mathbf{d}$ is a random variable over $\mathcal{R}_d^n$, independent of $\mathbf{e}$, such that for some constant k, $|\mathbf{d}_i| \leq k$. Note that we do *not* assume that e and $\mathbf{d}$ are independent. Thus, if we apply Theorem 1 in a straightforward way, we obtain a subgaussinity parameter of $\gamma + k\sqrt{n}\delta$. However, we can do better: The following lemma shows that we can bound the parameter by $\sqrt{\gamma^2 + k^2 n \delta^2}$.

[8] This is simply a special case of the usual definition of the trace function, but we do not need the general definition here.

Lemma 2. *Let e be a γ-subgaussian variable over $\mathcal{R}_d$ and $\mathbf{e} = (\mathbf{e}_1, \ldots, \mathbf{e}_n)$ be a vector of independent δ-subgaussian random variables over $\mathcal{R}_d$. Let $\mathbf{d}$ be a random variable over $\mathcal{R}_d^n$ such that $|\mathbf{d}_i| \leq k$ for all i. If $\mathbf{d}$ and $\mathbf{e}$ are independent and e and $\mathbf{e}$ are independent, then $e + \langle \mathbf{d}, \mathbf{e} \rangle$ is $\sqrt{\gamma^2 + k^2 n \delta^2}$-subgaussian.*

Proof. We first consider the case where $e = 0$ and $\mathbf{d}$ is a fixed vector instead of a random variable. For every $b \in \mathcal{R}_d$ and every i, we have $\operatorname{Tr}(\mathbf{d}_i \mathbf{e}_i b)/\|b\| \leq k \operatorname{Tr}(\mathbf{e}_i(\mathbf{d}_i b))/\|\mathbf{d}_i b\|$ which is $(k\delta)$-subgaussian. From the independence of the $\mathbf{e}_i$, it follows that $\operatorname{Tr}(\langle \mathbf{d}, \mathbf{e} \rangle b)/\|b\|$ is $(\sqrt{n} k \delta)$-subgaussian.

If $\mathbf{d}$ and e are random variables independent of $\mathbf{e}$, it holds for every $b \in \mathcal{R}_d$ that

$$\begin{aligned}
&\mathbb{E}[\exp(t \operatorname{Tr}(eb + \langle \mathbf{d}, \mathbf{e} \rangle b)/\|b\|)] \\
&= \sum_{e^*, \mathbf{d}^*} P[e = e^*, \mathbf{d} = \mathbf{d}^*] \cdot \mathbb{E}[\exp(t \operatorname{Tr}(eb + \langle \mathbf{d}, \mathbf{e} \rangle b)/\|b\|) \mid e = e^*, \mathbf{d} = \mathbf{d}^*] \\
&= \sum_{e^*, \mathbf{d}^*} P[e = e^*, \mathbf{d} = \mathbf{d}^*] \cdot \exp(t \operatorname{Tr}(e^* b)/\|b\|) \cdot \mathbb{E}[\exp(t \operatorname{Tr}(\langle \mathbf{d}^*, \mathbf{e} \rangle b)/\|b\|)] \\
&\leq \sum_{e^*, \mathbf{d}^*} P[e = e^*, \mathbf{d} = \mathbf{d}^*] \cdot \exp(t \operatorname{Tr}(e^* b)/\|b\|) \cdot \exp\bigl(t^2 n k^2 \delta^2/2\bigr) \\
&= \mathbb{E}[t \exp(\operatorname{Tr}(eb)/\|b\|)] \cdot \exp\bigl(t^2 n k^2 \delta^2/2\bigr) \\
&\leq \exp\bigl(t^2 \bigl(\gamma^2 + n k^2 \delta^2\bigr)/2\bigr)
\end{aligned}$$

which concludes the proof.

Finally, we show that if we trace down a subgaussian random variable over $\mathcal{R}_{pq}$ down to $\mathcal{R}_p$, the result is a subgaussian random variable over $\mathcal{R}_p$.

Lemma 3. *Let A be a δ-subgaussian random variable over $\mathcal{R}_{pq}$. Then $\operatorname{Tr}^*_{\mathcal{R}_{pq}/\mathcal{R}_p}(A)$ is δ-subgaussian as well.*

Proof. Let $b \in \mathcal{R}_p$. Then, $\operatorname{Tr}^*_{\mathcal{R}_p/\mathbb{Z}}\bigl(\operatorname{Tr}^*_{\mathcal{R}_{pq}/\mathcal{R}_p}(Ab)\bigr)/\|b\| = \operatorname{Tr}^*(Ab)/\|b\|$ which is δ-subgaussian by assumption.

2.3 Gadgets

Throughout this exposition we use a binary *decomposition* operation on ring elements, and the reverse. For simplicity we adopt the notation of gadget vector and matrix.

Definition 6. *The gadget vector $\mathbf{g}^T$ of size K is set to $\bigl(1\ 2\ 2^2\ \cdots\ 2^{K-1}\bigr) \in \mathcal{R}_d^K$. Reciprocally, we define $\mathbf{g}^{-T}$ as a function such that, for $\mathbf{w} \in \mathcal{R}_d^n$, $\mathbf{V} = \mathbf{g}^{-T}(\mathbf{w})$ is a $(K \times n)$-matrix whose entries are ring elements with coefficients in $\{0, 1\}$ such that $\mathbf{g}^T \mathbf{V} = \mathbf{w}$.*

Definition 7. *For some integer $n \geq 1$, the gadget matrix $\mathbf{G}_n$ is defined by $\mathbf{G}_n = \mathbf{I}_{n+1} \otimes \mathbf{g} \in \mathcal{R}_d^{(n+1)K \times (n+1)}$.*

$$\mathbf{G}_n^T = \begin{pmatrix}
1\ 2 \cdots 2^{K-1} & 0\ 0 \cdots 0 & \cdots & 0\ 0 \cdots 0 \\
0\ 0 \cdots 0 & 1\ 2 \cdots 2^{K-1} & \cdots & 0\ 0 \cdots 0 \\
\vdots & \vdots & \vdots & \vdots \\
0\ 0 \cdots 0 & 0\ 0 \cdots 0 & \cdots & 1\ 2 \cdots 2^{K-1}
\end{pmatrix}$$

We define $\mathbf{G}_n^{-1}$ similarly to $\mathbf{g}^{-T}$: for $\mathbf{a} \in \mathcal{R}_d^{n+1}$, we let $\mathbf{d} = \mathbf{G}_n^{-1}(\mathbf{a}) \in \mathcal{R}_d^{(n+1)K}$ be the vector whose entries have coefficients in $\{0,1\}$ such that $\mathbf{d}^T \cdot \mathbf{G} = \mathbf{a}$. For convenience we write $\mathbf{G}_n = \mathbf{G}$ as n is typically clear from context.

2.4 Circulant LWE and reduction to Ring-LWE

It is well known that the naive decisional version of Ring-LWE is insecure over circulant rings, simply by exploiting the CRT decomposition. Say that d is prime, and note that $\mathcal{R}_d/Q\mathcal{R}_d \simeq \tilde{\mathcal{R}}_d/Q\tilde{\mathcal{R}}_d \times \mathbb{Z}/Q\mathbb{Z}$ if Q is coprime to d, so one may mount an attack on the $\mathbb{Z}/Q\mathbb{Z}$ part (projecting to this part corresponds to evaluate the polynomial at 1, and therefore maintain smallness of the error). However, this does not mean that such rings are inherently insecure: the NTRU cryptosystems [23, 22] use circulant rings, choosing the secret key and errors that evaluate to a fixed known value (say 0) at 1.

This suggests a strategy to construct a variant of Ring-LWE over circulant rings that would be as secure as the cyclotomic Ring-LWE, simply by lifting all elements $\tilde{x} \in \tilde{\mathcal{R}}_d/Q\tilde{\mathcal{R}}_d$ to $x \simeq (\tilde{x}, 0)$, yet this reverse CRT operation may not keep small elements small. In Appendix A.1 we show how to circumvent this obstacle, and discuss error sampling in practice in Appendix A.3.

3 Encryption schemes

3.1 LWE Encryption

We recall the definition of the most basic LWE symmetric encryption scheme (see [7, 28, 4]). LWE symmetric encryption is parametrized by a dimension n, a message modulus $t \geq 2$, a ciphertext modulus $Q = n^{O(1)}$ and an error distribution χ. The message space of the scheme is $\mathbb{Z}_t$. (Typically, $e \leftarrow \chi$ satisfies the condition $|e| < Q/2t$, and $t = 2$ is used to encrypt message bits.) The (secret) key of the encryption scheme is a vector $\mathbf{s} \in \mathbb{Z}_Q^n$, which may be chosen uniformly at random, or as a random short vector. The encryption of a message $m \in \mathbb{Z}_t$ under key $\mathbf{s} \in \mathbb{Z}_Q^n$ is

$$\mathbf{c} = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \lfloor Q/t \rceil\, m \bmod Q) \in \mathbb{Z}_Q^{n+1} \tag{1}$$

where $\mathbf{a} \leftarrow \mathbb{Z}_Q^n$ is chosen uniformly at random. A ciphertext $(\mathbf{a}, b)$ is decrypted by computing

$$m' = \lfloor t(b - \langle \mathbf{a}, \mathbf{s} \rangle)/Q \rceil \bmod t \in \mathbb{Z}_t. \tag{2}$$

We write $\mathbf{c} \in \mathsf{LWE}_{\mathbf{s}}^{t|Q}(m)$ to denote that $\mathbf{c}$ is an LWE-encryption of m, and $\mathbf{c} \in \mathsf{LWE}_{\mathbf{s}}^{t|Q}(m; E)$ if $\mathbf{c}$ is a random LWE-ciphertext such that $\mathbf{c} = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \lfloor Q/t \rceil\, m + e)$ where e is a subgaussian random variable with parameter E. The error of $\mathbf{c} = (\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{t|Q}(m)$ is $\mathsf{err}(\mathbf{c}) = (b - \langle \mathbf{a}, \mathbf{s} \rangle - \lfloor Q/t \rceil\, m) \bmod Q$, reduced modulo Q to the centered interval $[-Q/2, Q/2)$.

Notice that the error $\mathsf{err}(\mathbf{a}, b)$ depends not just on $(\mathbf{a}, b)$, but also on $\mathbf{s}, Q, t$ and m. By the subgaussian tail estimate, if $e = \mathsf{err}(\mathbf{c})$ is subgaussian with parameter E, then $|e| < \sqrt{2\lambda}E$ except with probability at most $2\exp(-\lambda)$. Thus, if t divides Q and $E \leq Q/(2t\sqrt{2\lambda})$, the decryption procedure recovers the encrypted message with high probability:

$$\lfloor t(b - \langle \mathbf{a}, \mathbf{s} \rangle)/Q \rceil \bmod t = \left\lfloor \frac{t}{Q} \cdot \left(\frac{Q}{t} m + e \right) \right\rceil = \left\lfloor m + \frac{t}{Q} e \right\rceil = m \bmod t$$

because $\frac{t}{Q}|e| < 1/2$ except with probability $2\exp(-\lambda)$.

3.2 CLWE and CGSW Encryption Schemes

Below, we describe two encryption schemes, Circulant-LWE and Circulant-GSW[9], which we need for our *homomorphic accumulator* (see Section 4). We do not specify any decryption procedures since these are not needed for the homomorphic accumulator.

CLWE Encryption Scheme.

Definition 8. *We let $\mathcal{R}$, $\tilde{\mathcal{R}}$, d, and Q be as in Section 2.2. Let $t \geq 2$ be the plaintext modulus. The Circulant-LWE scheme over $\mathcal{R}$ consists of the following algorithms:*

- KeyGen*: Output a uniformly random element s of $\tilde{\mathcal{R}}$.*
- $\mathsf{Enc}_s(m)$ *for $m \in \mathcal{R}/t\mathcal{R}$: Let (a, b) be a sample from the Circulant-LWE distribution over $\mathcal{R}$ with secret s and output $(a, b' = b + \lfloor Q/t \rceil \cdot m)$.*

We also define an n-dimensional variant of the scheme where the key is $\mathbf{s} \in \mathcal{R}^n$, $\mathbf{a}$ is a random vector in $\mathcal{R}^n$ and the product $a \cdot s$ is replaced by the inner product over $\mathcal{R}$ $\langle \mathbf{a}, \mathbf{s} \rangle = \sum_{i=1}^{n} \mathbf{a}_i \cdot \mathbf{s}_i$.

Lemma 4. *If the decisional $\tilde{\mathcal{R}}$-* LWE *problem is hard, then the Circulant-LWE over scheme is cpa-secure for messages of the form $m = X^k$.*

We defer this proof to Appendix A.2.

CGSW Encryption Scheme.

Definition 9. *We let $\mathcal{R}$, $\tilde{\mathcal{R}}$, d, and Q be as in Section 2.2 and $\mathbf{G}$ as in Definition 7. Furthermore, let $t \geq 2$ be the plaintext modulus and B an integer ≥ 2, let K be the smallest integer such that $B^K \geq Q$.*

The Circulant-GSW scheme is described by the following algorithms:

- KeyGen*: Sample a uniformly random s from $\tilde{\mathcal{R}}$.*
- $\mathsf{Enc}_s(m)$ *for $m \in \mathcal{R}/t\mathcal{R}$: Generate a matrix $\mathbf{A} \in \mathcal{R}^{2K \times 2}$ where each row is a sample from the Circulant-LWE distribution with secret s. Output $\mathbf{A} + \lfloor Q/t \rceil \cdot m\mathbf{G}$.*

We also define a n-dimensional variant of the scheme where $\mathbf{A} \in \mathcal{R}^{(n+1)K \times (n+1)}$ whose rows are samples from the n-dimensional Circulant-LWE and where $\mathbf{G}_1$ is replaced by $\mathbf{G}_n$.

Lemma 5. *If the decisional $\tilde{\mathcal{R}}$-* LWE *problem is hard, then the Circulant-GSW scheme is cpa-secure.*

We defer this proof to Appendix A.2.

Ciphertext Spaces.

- We write $\mathbf{c} \in \mathcal{R}_d\mathsf{LWE}^{t|Q}_{\mathbf{s}}(m, E)$ if $\mathbf{c} = (\mathbf{a}, \mathbf{a}^t\mathbf{s} + \left\lfloor \frac{Q}{t} \right\rceil m + \mathbf{e})$ for some random error vector $\mathbf{e}$ that is E-subgaussian. We extend the notation to $\mathbf{C} \in \mathcal{R}_d\mathsf{LWE}^{t|Q}_{\mathbf{s}}(\mathbf{m}^T, E)$ for message $\mathbf{m} \in \mathcal{R}_t^k$ that are vectors, meaning that the i-th column $\mathbf{C}_i$ of $\mathbf{C}$ is in $\mathcal{R}_d\mathsf{LWE}^{t|Q}_{\mathbf{s}}(m_i, E)$. Furthermore, we write $\mathsf{err}(\mathbf{c})$ for the error term e in $\mathbf{c}$.
- We write $\mathbf{C} \in \mathcal{R}_d\mathsf{GSW}^{t|Q}_{s}(m, E)$ if $\mathbf{C} = (\mathbf{a}, \mathbf{a}s + \mathbf{e}) + \left\lfloor \frac{Q}{t} \right\rceil \cdot m\mathbf{G}$, and the components of $\mathbf{e}$ are independent E-subgaussian variables. We write $\mathsf{err}(\mathbf{C})$ for the error vector $\mathbf{e}$ in $\mathbf{C}$.

[9] Based on [19].

4 Homomorphic Operations

Most of the operations presented below are meaningful both in the ring/circulant-setting or over the integers. We consider the $\mathcal{R}\mathsf{LWE}$ problem over rings $\mathcal{R}_d = \mathbb{Z}[X]/(X^d - 1)$ with d prime and over $\mathcal{R} = \mathbb{Z}$ (i.e., simply the LWE problem). However, most of the results presented in this section also hold for cyclotomic rings. We assume that coefficients of ring elements in $\mathcal{R}/Q\mathcal{R}$ can be added and multiplied in constant time since, in our implementation, each coefficient fits into a machine word. Thus, adding two ring elements takes time $O(d)$ and multiplying them takes time $O(d \log d)$ using FFT.

4.1 Known Building Blocks

Let us first recall, within our formalism, known building blocks from the literature. The only novelty is in this section concerns the $\mathsf{FunExpExtract}$ function: while this was already constructed in previous work, in our set-up we will need to apply a trick from [18] to improve its efficiency.

Linearity.

Key Material: None
Runtime: $O(nd)$ for addition, $O(nd \log d)$ for multiplication
Signature:

$$\begin{aligned} \mathsf{Add} : \mathcal{R}_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m;\, E) \times \mathcal{R}_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m';\, E') \\ \to \mathcal{R}_d\mathsf{LWE}_{s}^{t|Q}\left(m + m';\, \sqrt{E^2 + E'^2}\right) \\ x \in \mathcal{R}_d, \mathsf{Mult}_x : \mathcal{R}_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m;\, E) \to \mathcal{R}_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}\left(xm;\, |x|E\right) \end{aligned} \tag{3}$$

The error term in the result of Add holds when the error terms in the input ciphertexts are independent. Otherwise, it is $E + E'$.

The Add operations are computed by simply adding the ciphertexts component-wise. The Mult_x operations work by scalar multiplication with x. If e is subgaussian with parameter E, then $x \cdot e$ is subgaussian with parameter $|x|E$ since, for all b, $\mathrm{Tr}^*(xeb)/\|b\| \leq |x| \, \mathrm{Tr}^*(e(xb))/\|xb\|$, and $\mathrm{Tr}^*(e(xb))/\|xb\|$ is E-subgaussian.

Modulus Switching.

Key Material: None
Runtime: $O(d)$
Signature: $\mathsf{ModSwitch}^{Q \to Q'}$:

$$\mathcal{R}_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m;\, E) \to \mathcal{R}_d\mathsf{LWE}_{\mathbf{s}}^{t|Q'}\left(m;\, \sqrt{(kE)^2 + 1 + \sum_i |\mathbf{s}_i|^2}\right) \tag{4}$$

where $\mathbf{s} \in \mathcal{R}_d^n$ and $k = \lfloor Q'/t \rceil \,/\, \lfloor Q/t \rceil \approx Q'/Q$.

The basic idea of modulus switching is to multiply the ciphertext with Q'/Q, or rather $\lfloor Q'/t \rceil \,/\, \lfloor Q/t \rceil$. However, since this factor is not necessarily an integer, we instead use a randomized rounding function $[x] = \lfloor x \rfloor + B_r$ where B_r is a Bernoulli random variable with $\Pr[B_r = 1] = x - \lfloor x \rfloor$. The rounding error $r = [x] - x$ is subgaussian with parameter 1. Let us write $k = \lfloor Q'/t \rceil / \lfloor Q/t \rceil$. Applying the rounding function component-wise to $k \cdot (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \lfloor Q/t \rceil m + e)$, we obtain

$$(k\mathbf{a} + \mathbf{r}, k\langle \mathbf{a}, \mathbf{s} \rangle + \lfloor Q'/t \rceil m + ke + r') = (k\mathbf{a} + \mathbf{r}, \langle k\mathbf{a} + \mathbf{r}, \mathbf{s} \rangle + \lfloor Q'/t \rceil m + ke + r' - \langle \mathbf{r}, \mathbf{s} \rangle)$$

where $\mathbf{r}$ is the vector of rounding errors for $k\mathbf{a}$ and r is the rounding error for b. Thus, the error term of the modulus-switched ciphertext is $ke + r - \langle \mathbf{r}, \mathbf{s} \rangle$. For each i, $\mathbf{r}_i \mathbf{s}_i$ is $|\mathbf{s}_i|$-subgaussian. Since all terms in the sum are independent, the error parameter is $\sqrt{(kE)^2 + 1 + \sum_i |\mathbf{s}_i|^2}$.

Remark 1. We only use modulus switching in the following two cases: when the dimension of the key is $n = 1$, and for short keys in $\mathbb{Z}^n$, i.e., n-dimensional keys where $|\mathbf{s}_i| \leq 1$. In the first case, the error parameter simplifies to $\sqrt{(kE)^2 + 1 + |s|^2}$, in the second case to $\sqrt{(kE)^2 + n + 1}$.

Key switching.

Key Material: $\mathbf{S} = [\mathbf{S}_i]_{i \in [n]}$ where $\mathbf{S}_i \in \mathcal{R}_d\mathsf{LWE}^{Q|Q}_{s'}(s_i \cdot \mathbf{g}^T;\, \sigma)$ (Size: $O(nd \log^2 Q)$)
Runtime: $O(d \log dn \log Q)$
Signature: $\mathsf{KeySwitch}^{\mathbf{s} \to s'}_{\mathbf{S}}$:

$$\mathcal{R}_d\mathsf{LWE}^{t|Q}_{\mathbf{s}}(m;\, E) \to \mathcal{R}_d\mathsf{LWE}^{t|Q}_{s'}\left(m;\, \sqrt{E^2 + \sigma^2 d^2 nK}\right). \tag{5}$$

where $\mathbf{s} \in \mathcal{R}_d^n$, $s' \in \mathcal{R}_d$.

Algorithm 1 $\mathsf{KeySwitch}^{\mathbf{s} \to s'}_{\mathbf{S}}(\mathbf{c})$: Transform an $\mathcal{R}_d\mathsf{LWE}$ ciphertext under key $\mathbf{s}$ into a ciphertext under s'.

```
Require:
    $\mathbf{S} = [\mathbf{S}_i]_{i \in [n]}$ where $\mathbf{S}_i \in \mathcal{R}_d\mathsf{LWE}^{Q|Q}_{s'}(s_i \cdot \mathbf{g}^T;\, \sigma)$.
    A ciphertext $(\mathbf{a}, b) \in \mathcal{R}_d\mathsf{LWE}^{t|Q}_{\mathbf{s}}(m; E)$ for some $m \in \mathcal{R}/t\mathcal{R}$.
Ensure: A ciphertext $c \in \mathcal{R}_d\mathsf{LWE}^{t|Q}_{s'}\left(m;\, \sqrt{E^2 + \sigma^2 d^2 nK}\right)$ if the error terms in $\mathbf{c}$ and $\mathbf{S}$ are independent.

    return $(\mathbf{0}_{n'}, b) - \mathbf{g}^{-T}(\mathbf{a}) \cdot \mathbf{S}$
```

Lemma 6. *Algorithm 1 is correct. Furthermore, if $e = \mathsf{err}(\mathbf{c})$ and $\mathbf{e}_i = \mathsf{err}(\mathbf{S}_i)$, then the error term of the output ciphertext is $e + \sum_{i=1}^n \mathbf{d}_i^T \mathbf{e}_i$, where each $\mathbf{d}_i$ is a vector whose entries have operator norm at most d.*

Proof. By definition of $\mathbf{g}^{-T}$, it is easy to see that the error term is $e - \sum_{i=1}^n \mathbf{g}^{-T}(\mathbf{a}_i)\mathbf{e}_i$ and each component of $g^{-T}(\mathbf{a}_i)$ is in $\mathcal{R}_d/2\mathcal{R}_d$. Thus, the second part of the lemma follows. The first part holds because for every i, $g^{-T}(\mathbf{a}_i)\mathbf{e}_i$ is subgaussian with parameter at most $\sqrt{K} d\sigma$. If the error terms are independent, it follows that the error parameter is as stated in the algorithm.

Remark 2. In practice, the choice of the basis decomposition B for the gadget is important. It allows to trade off key size and running time against error growth. We use

$$S = \left[\mathcal{R}_d\mathsf{LWE}^{1;Q}_{s'}(B^j \mathbf{s}_i; \sigma)\right]_{i=1\ldots n, j=0\ldots K-1}, \textit{with } K = \lceil \log_B Q \rceil$$

as key material. The key size decreases to $O(nn'dK \log Q)$, and the running time decreases to $O(d \log dnn'K)$, while the output error parameter also increases to $\sqrt{E^2 + \sigma^2 d^2 B^2 nK}$.

External Multiplication.

Key Material: None
Runtime: $O(Kd \log d)$
Signature: $\mathsf{ExtMult}$:

$$\mathcal{R}_d\mathsf{LWE}_s^{t|Q}(T^m;\, E) \times \mathcal{R}_d\mathsf{GSW}_s^{t|Q}(T^{m'};\, E') \rightarrow \mathcal{R}_d\mathsf{LWE}_s^{t|Q}\left(T^{m+m'};\, \sqrt{E^2 + 2Kd^2E'^2}\right) \quad (6)$$

for $s \in \mathcal{R}$ if $\lfloor Q/t \rceil$ is invertible modulo Q.

Algorithm 2 $\mathsf{ExtMult}(\mathbf{c}, \mathbf{C})$: Multiply an $\mathcal{R}_d\mathsf{LWE}$ ciphertext and a $\mathcal{R}_d\mathsf{GSW}$ ciphertext into a $\mathcal{R}_d\mathsf{LWE}$ ciphertext.

Require: A ciphertext $\mathbf{c} \in \mathcal{R}_d\mathsf{LWE}_s^{t|Q}(T^m;\, E)$, and a ciphertext $\mathbf{C} \in \mathcal{R}_d\mathsf{GSW}_s^{t|Q}(T^{m'};\, E')$ with $\lfloor Q/t \rceil$ invertible modulo Q.
Ensure: A ciphertext $c \in \mathcal{R}_d\mathsf{LWE}_s^{t|Q}\left(T^{m+m'};\, \sqrt{E^2 + 2Kd^2E'^2}\right)$.

```
return $\mathbf{G}^{-1}\big(\lfloor Q/t \rceil^{-1} \cdot \mathbf{c}\big) \cdot \mathbf{C}$
```

Lemma 7. *Algorithm 2 is correct. Furthermore, for $e = \mathsf{err}(\mathbf{c})$ and $\mathbf{e} = \mathsf{err}(\mathbf{C})$, the error term of the output is $X^k \cdot e + \mathbf{d}^T\mathbf{e}$ for some k and a random vector $\mathbf{d} \in \mathcal{R}_d^{2K}$ independent of $\mathbf{e}$ with $\|\mathbf{d}_i\| \leq d$ for every i.*

Proof. Write $u = \lfloor Q/t \rceil$, so $\mathbf{c} = (a, as + e + \lfloor Q/t \rceil T^m)$ and $\mathbf{C} = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \mathbf{e}) + uT^{m'}\mathbf{G}$. Let $\mathbf{d} = \mathbf{G}^{-1}(u^{-1} \cdot \mathbf{c})$. We have:

$$\begin{aligned}
\mathbf{d}^T \cdot \mathbf{C} &= \mathbf{d}^T \cdot (\mathbf{a}, \mathbf{a}s + \mathbf{e}) + uT^{m'}\mathbf{d}^T\mathbf{G} \\
&= (\mathbf{d}^T\mathbf{a}, \mathbf{d}^T\mathbf{a}s + \mathbf{d}^T\mathbf{e}) + uu^{-1}T^{m'}\left(a, as + e + \left\lfloor \frac{Q}{t} \right\rceil T^m\right) \\
&= \left(a', a's + e' + \left\lfloor \frac{Q}{t} \right\rceil T^{m+m'}\right)
\end{aligned}$$

where $a' = \mathbf{d}^T\mathbf{a} + aT^{m'}$ and $e' = \mathbf{d}^T\mathbf{e} + eT^{m'}$. Each component of $\mathbf{e}$ is independent and subgaussian with parameter E', and $\mathbf{d}$ is a vector in $\mathcal{R}_d^{2K}$, where each entry has binary coefficients. Thus, for every i, we have $|\mathbf{d}_i| \leq d$. By Lemma 2, the error parameter follows.

Exponent Function Extraction.

Key Material: A key-switch key $\mathbf{S}$ from $\mathbf{s}^{(pq)} \in \mathcal{R}_{pq}^3$ to $s' = \sum_{i=0}^{p-1} \mathbf{s}_{i+1} X^i \in \mathcal{R}_p \subseteq \mathcal{R}_{pq}$ (Size: $O(p(q+n)K^2)$
Runtime: $O(pq \log(pq) K)$
Signature: $\mathsf{FunExpExtract}_{F,\mathbf{S}}^{\mathbf{s}^{(pq)} \to \mathbf{s}}$:

$$\mathcal{R}_{pq}\mathsf{LWE}_{\mathbf{s}^{(pq)}}^{t|Q'}(Z^m;\, E) \to \mathsf{LWE}_{\mathbf{s}}^{t|Q'}\left(F(m);\, |F|\sqrt{E^2 + 3\sigma^2 p^2 q^2 K}\right) \tag{7}$$

for some function $F : \mathbb{Z}_{pq} \to \mathbb{Z}_t$ where $|F| = \sum_{i \in \mathbb{Z}_{pq}} |F(i)|$ and $\mathbf{s} \in \mathbb{Z}^p$.

Let us first consider the function F_0 that maps $0 \mapsto 1$ and $k \mapsto 0$ for $k \neq 0$. If we can extract this function, we can extract *any* function by first multiplying the ciphertext with an appropriate polynomial.

This extraction is easily provided by the trace function $\mathrm{Tr}^* = \mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathbb{Z}}$ (see Lemma 1). Indeed, if $(a, b) \in \mathcal{R}_{pq}\mathsf{LWE}_s(m)$, then $(\mathbf{a}, \mathrm{Tr}^*(b)) \in \mathsf{LWE}_{\mathbf{s}}(m_0)$, where $\mathbf{a}, \mathbf{s} \in \mathbb{Z}^{pq}$ are the vectors of coefficients of a and s.

However, this leads to an LWE ciphertext with quadratic dimension $pq = \Theta(n^2)$, that must be key-switched to a much smaller dimension $\Theta(n)$. Such a key-switch without any ring structure would require up to $\tilde{\Theta}(n^3)$ running time, and as much key-material.

To circumvent this issue, we exploit the intermediate ring, following one of the tricks of [17]. Namely, we choose a key in $\mathcal{R}_p$, which can also be viewed as an element of $\mathcal{R}_{pq}$. Switching to this key, exploiting the structure of $\mathcal{R}_{pq}$, requires only $\tilde{\Theta}(pq) = \tilde{\Theta}(n^2)$ operations. Then, one can trace a down to $\mathcal{R}_p$, and b down to $\mathbb{Z}$, and obtain the desired result.

Algorithm 3 $\mathsf{FunExpExtract}_{F,\mathbf{S}}^{\mathbf{s}^{(pq)} \to \mathbf{s}}$: Turn an $\mathcal{R}_{pq}\mathsf{LWE}$ encryption of Z^m into an LWE encryption of $F(m)$.

```
Require:
  A ciphertext $\mathbf{c} \in \mathcal{R}_{pq}\mathsf{LWE}_{\mathbf{s}^{(pq)}}^{t|Q'}(Z^m; E)$,
  A function $F : \mathbb{Z}_{pq} \to \mathbb{Z}_t$,
  A key-switch key $\mathbf{S}$ from $\mathbf{s}^{(pq)}$ to $s' \in \mathcal{R}_p \subseteq \mathcal{R}_{pq}$, where $s' = \sum_{i=0}^{p-1} \mathbf{s}_{i+1}^{(pq)} X^i$.
Ensure: A ciphertext $c' \in \mathsf{LWE}_{\mathbf{s}}^{t|Q'}\left(F(m); |F|\sqrt{E^2 + 3\sigma^2 p^2 q^2 K}\right)$.

  $f \leftarrow \sum_{i \in \mathbb{Z}_{pq}} F(i) Z^{-i \bmod pq} \in \mathcal{R}_{pq}$
  $\mathbf{c} \leftarrow \mathsf{KeySwitch}_{\mathbf{S}}^{\mathbf{s}^{(pq)} \to s'}(\mathbf{c})$          $\triangleright \in \mathcal{R}_{pq}\mathsf{LWE}_{s'}^{t|Q'}(Z^m)$
  $\mathbf{c} \leftarrow \mathsf{Mult}_f(\mathbf{c})$          $\triangleright \in \mathcal{R}_{pq}\mathsf{LWE}_{s'}^{t|Q'}(\sum_{i \in \mathbb{Z}_{pq}} F(i) Z^{m-i \bmod pq})$
  $(a, b) \leftarrow \mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathcal{R}_p}(\mathbf{c})$ (component-wise)
          $\triangleright \in \mathcal{R}_p\mathsf{LWE}_{s'}^{t|Q'}(\sum_{i,\ \mathrm{st}\ q|(m-i)} F(i) X^{m-i \bmod pq})$
  $\mathbf{a} \leftarrow (a_0, a_{p-1}, a_{p-2}, \ldots, a_1)$
  $b \leftarrow \mathrm{Tr}^*_{\mathcal{R}_p/\mathbb{Z}}(b)$
  return $(\mathbf{a}, b)$
```

Lemma 8. *Algorithm 3 is correct and runs in time $O(pq \log(pq) \log Q')$.*

Proof. We can compute $\mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathcal{R}_p}(x)$ by examining p coefficients of x, and $\mathrm{Tr}^*_{\mathcal{R}_p/\mathbb{Z}}(x)$ is simply the constant term of x. Thus, the runtime is dominated by the key-switch, which runs in time $O(pq\log(pq)K)$. After the multiplication and key-switch, it holds that

$$c \in \mathcal{R}_{pq}\mathsf{LWE}^{t|Q'}_{\mathbf{s}^{(pq)}}\left(\sum_{i\in\mathbb{Z}_{pq}} Z^{m-i \bmod pq}; |F|\sqrt{E^2+3\sigma^2p^2q^2K}\right)$$

since $|f| \le |F|$. Using Lemma 3, the linearity of the trace function, and the fact that $s' \in \mathcal{R}_p$, we conclude that after the trace,

$$(a,b) \in \mathcal{R}_p\mathsf{LWE}^{t|Q'}_{s'}\left(\mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathcal{R}_p}\left(\sum_{i\in\mathbb{Z}_{pq}} F(i)Z^{m-i \bmod pq}\right); |F|\sqrt{E^2+3\sigma^2p^2q^2K}\right)$$

It holds that

$$\mathrm{Tr}^*(b) = \mathrm{Tr}^*(a\cdot s') + \lfloor Q'/t\rceil\, \mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathbb{Z}}\left(\sum_{i\in\mathbb{Z}_{pq}} F(i)Z^{m-i \bmod pq}\right) + \mathrm{Tr}^*(e)$$

and by Lemma 1, $\mathrm{Tr}^*\left(\sum_{i\in\mathbb{Z}_{pq}} F(i)Z^{m-i \bmod pq}\right) = F(j)$ if $m = j$. Since $\mathrm{Tr}^*(as) = a_0s_0 + \sum_{i=1}^{p-1} a_{p-i}s_i = \langle \mathbf{a}, \mathbf{s}\rangle$ and Tr^* does not increase the error parameter, the correctness of our algorithm follows.

Remark 3. Note that we could reduce the error parameter in Algorithm 3 by performing the multiplication before the key-switch. However, doing the key-switch first allows to amortize the cost of gates with multiple outputs, as we shall describe in section B.3.

4.2 New Building Blocks

Exponent Multiplication by Galois Conjugation.

Key Material: None
Runtime: $O(nd)$
Signature: Galois^α :

$$\mathcal{R}_d\mathsf{LWE}^{t|Q}_s(T^m;\, E) \to \mathcal{R}_d\mathsf{LWE}^{t|Q}_{\psi_\alpha(s)}(T^{\alpha m};\, E)\,. \tag{8}$$

where $\alpha \in \mathbb{Z}_d^*$ and ψ_α is the automorphism of $\mathcal{R}_d$ defined by $T \mapsto T^\alpha$.

Given a $\mathcal{R}_d\mathsf{LWE}$-ciphertext $(a, as + \lfloor Q/t\rceil\, T^m + e)$, by applying ψ_α component-wise, we obtain $(\psi_\alpha(a), \psi_\alpha(a)\cdot\psi_\alpha(s) + \lfloor Q/t\rceil\, T^{\alpha m} + \psi_\alpha(e))$. Applying Galois^α does not change the error parameter because $\mathrm{Tr}^*(\psi_\alpha(e)b) = \mathrm{Tr}^*(\psi_\alpha(e\psi_\alpha^{-1}(b))) = \mathrm{Tr}^*(e\psi_\alpha^{-1}(b))$. The running time is $O(d)$ because for $x \in \mathcal{R}$, $\psi_\alpha(x)$ is computed simply by permuting the coefficients of x. Even if the ciphertext is in FFT representation, the runtime remains $O(d)$, as ψ_α also acts on those representations by permutation.

Exponent CRT by tensoring.

Key Material: None
Runtime: $O(pq)$

Signature: ExpCRT:

$$\mathcal{R}_p\mathsf{LWE}^{t|Q_\otimes}_{s_p}(X^{m_p};\, E_p) \times \mathcal{R}_q\mathsf{LWE}^{t|Q_\otimes}_{s_q}(Y^{m_q};\, E_q) \\ \to \mathcal{R}_{pq}\mathsf{LWE}^{t|Q_\otimes}_{\mathbf{s}}\left(Z^m;\, \sqrt{E_p^2+E_q^2} + t\sqrt{2\lambda}E_pE_q\right) \quad (9)$$

if $t\cdot\lfloor Q_\otimes/t\rceil = 1 \bmod Q_\otimes$, and where $m = \alpha m_p + \beta m_q$ is such that $m_p = m \bmod p$ and $m_q = m \bmod q$ and $\mathbf{s} = (-\psi_\alpha(s_p)\otimes\psi_\beta(s_q), \psi_\alpha(s_p)\otimes 1, 1\otimes\psi_\beta(s_q))$.

Note that the condition $t\cdot\lfloor Q_\otimes/t\rceil = 1$ can be easily satisfied in our bootstrapping scheme because we perform a modulus switch before and after ExpCRT.

Algorithm 4 $\mathsf{ExpCRT}(\mathbf{c}^{(p)}, \mathbf{c}^{(q)})$

Require: Ciphertexts $\mathbf{c}^{(p)} \in \mathcal{R}_p\mathsf{LWE}^{t|Q_\otimes}_{s_p}(X^{m_p}; E_p)$ and $\mathbf{c}^{(q)} \in \mathcal{R}_q\mathsf{LWE}^{t|Q_\otimes}_{s_q}(Y^{m_q}; E_q)$.

Ensure: A ciphertext $\mathbf{c} \in \mathcal{R}_{pq}\mathsf{LWE}^{t|Q_\otimes}_{\mathbf{s}}(Z^m; \sqrt{E_p^2+E_q^2} + t\sqrt{2\lambda}E_pE_q)$ except with probability $2\min(p,q)\exp(-\lambda)$ where $\mathbf{s} = (-\psi_\alpha(s_p)\otimes\psi_\beta(s_q), \psi_\alpha(s_p)\otimes 1, 1\otimes\psi_\beta(s_q))$ for $\alpha = q^{-1} \bmod p$ and $\beta = p^{-1} \bmod q$, and m is such that $m \bmod p = m_p$ and $m \bmod q = m_q$.

```
$(a_p, b_p) \leftarrow \mathsf{Galois}^\alpha(\mathbf{c}^{(p)})$        $\triangleright \in \mathcal{R}_p\mathsf{LWE}^{t|Q_\otimes}_{\psi_\alpha(s_p)}(X^{\alpha m_p}; E_p)$
$(a_q, b_q) \leftarrow \mathsf{Galois}^\beta(\mathbf{c}^{(q)})$        $\triangleright \in \mathcal{R}_q\mathsf{LWE}^{t|Q_\otimes}_{\psi_\beta(s_q)}(Y^{\beta m_q}; E_q)$
$\mathbf{a} \leftarrow (a_p\otimes a_q, a_p\otimes b_q, b_p\otimes a_q)$
return $(t\mathbf{a}, tb_p\otimes b_q)$
```

We will need the following lemma to bound the tensor product of two subgaussian random variables.

Lemma 9. *Let A and B be independent subgaussian random variables on $\mathcal{R}_p$ and $\mathcal{R}_q$, respectively, with parameters γ and δ. Then, for every $\lambda\in\mathbb{R}$, $A\otimes B$ is subgaussian with parameter $\sqrt{2\lambda}\gamma\delta$ except with probability $2\min(p,q)\exp(-\lambda)$.*[10]

Proof. We want to show that for every $y\in\mathcal{R}_{pq}\setminus\{0\}$, $\mathrm{Tr}^*((A\otimes B)y)/\|y\|$ is subgaussian (except with a small probability). Let $y\in\mathcal{R}_{pq}\setminus\{0\}$. We can write $y = \sum_{i=0}^{q-1} y_i\otimes Y^i$. It holds that $\|y\| = \sqrt{\sum_i \|y_i\|^2}$. Thus,

$$\frac{\mathrm{Tr}^*((A\otimes B)y)}{\|y\|} = \sum_i \frac{\mathrm{Tr}^*(Ay_i\otimes BY^i)}{\|y\|} = \sum_i \frac{\mathrm{Tr}^*(Ay_i)\cdot\mathrm{Tr}^*(BY^i)}{\|y\|}$$

Let E_i be the event that $|\mathrm{Tr}^*(Ay_i)| \geq \sqrt{2\lambda}\gamma\|y_i\|$. Applying the subgaussian tail estimate, we conclude that for each i, $p(E_i)\leq 2\exp(-\lambda)$. By the union bound, it follows that, for $E = \bigcup_i E_i$, $p(E)\leq 2q\exp(-\lambda)$. We now proceed similarly to the proof of Lemma 2. For every fixed value

[10] More formally, for some event E with $p(E)\leq 2\min(p,q)\exp(-\lambda)$, when conditioning on $\overline{E}$, $A\otimes B$ is subgaussian with parameter $\sqrt{2\lambda}\gamma\delta$.

$a \in \mathcal{R}_p$ such that $\mathrm{Tr}^*(ay_i) \ < \sqrt{2\lambda}\gamma\|y_i\|$ for all i, we have

$$\sum_i \frac{\mathrm{Tr}^*(ay_i)\cdot \mathrm{Tr}^*(BY^i)}{\|y\|} = \frac{\mathrm{Tr}^*\left(\sum_i B\,\mathrm{Tr}^*(ay_i)Y^i\right)}{\|y\|}$$

which is subgaussian with parameter

$$\frac{\left\|\sum_i \mathrm{Tr}^*(ay_i)Y^i\right\|\delta}{\|y\|} = \frac{\sqrt{\sum_i \mathrm{Tr}^*(ay_i)^2}\delta}{\sqrt{\sum_j \|y_j\|^2}} < \frac{\sqrt{2\lambda}\gamma\delta\sqrt{\sum_i \|y_i\|^2}}{\sqrt{\sum_j \|y_j\|^2}} = \sqrt{2\lambda}\gamma\delta$$

We can then use the independence of A and B to conclude that, conditioned on $\overline{E}$, $\mathrm{Tr}^*((A\otimes B)y)/\|y\|$ is $(\sqrt{2\lambda}\gamma\delta)$-subgaussian, as claimed.

Using a similar argument, this time writing $y = \sum_{i=0}^{p-1} X^i \otimes y_i$, it also follows that $\mathrm{Tr}^*((A\otimes B)y)/\|y\|$ is $(\sqrt{2\lambda}\gamma\delta)$-subgaussian except with probability $2p\exp(-\lambda)$. This proves our claim.

Lemma 10. *Algorithm 4 is correct and runs in time $\Theta(pq)$.*

Proof. Let $m' = \alpha q m_p + \beta p m_q \bmod pq$. It holds that $m' \bmod p = m_p$ and $m' \bmod q = m_q$. Thus, by the Chinese Remainder Theorem, $m' = m$. Let $s'_p = \psi_\alpha(s_p)$ and $s'_q = \psi_\beta(s_q)$. Let us write $b'_p = uX^{\alpha m_p} + e_p$ and $b'_q = Q_\otimes Y^{\beta m_q}/t + e_q$. We have

$$\begin{aligned} tb_p \otimes b_q &= ta_p s_p \otimes b_q + tb'_p \otimes a_q s_q + tb'_p \otimes b'_q \\ &= -ta_p s'_p \otimes a_q s'_q + ta_p s'_p \otimes b'_q + tb_p \otimes a_q s'_q + tb'_p \otimes b'_q \end{aligned}$$

and $tb'_p \otimes b'_q = \lfloor Q_\otimes/t \rceil X^{\alpha m_p} \otimes Y^{\beta m_q} + X^{\alpha m_p} \otimes e_q + e_p \otimes Y^{\beta m_q} + te_p \otimes e_q$. Since $X^{\alpha m_p} \otimes Y^{\alpha m_q} = Z^m$, the error term is

$$e_{pq} = X^{\alpha m_p} \otimes e_q + e_p \otimes Y^{\beta m_q} + te_p \otimes e_q.$$

Since e_p and e_q are independent, the sum of the first two terms is subgaussian with parameter $\sqrt{E_p^2 + E_q^2}$. The third term is subgaussian with parameter $t\sqrt{2\lambda}E_pE_q$, except with probability $2\min(p,q)\exp(-\lambda)$ by Lemma 9. In total, e_{pq} is subgaussian with parameter $\sqrt{E_p^2 + E_q^2} + t\sqrt{2\lambda}E_pE_q$ except with probability $2\min(p,q)\exp(-\lambda)$.

Thus, with $\mathbf{a} = (a_p \otimes a_q, a_p \otimes b_q, b_p \otimes a_q)$ and $\mathbf{s} = (-\psi_\alpha(s_p) \otimes \psi_\beta(s_q), \psi_\alpha(s_p) \otimes 1, 1 \otimes \psi_\beta(s_q)) = (-s'_p \otimes s'_q, s'_p \otimes 1, 1 \otimes s'_q)$, an easy computation shows that $tb_p \otimes b_q - t\langle \mathbf{a}, \mathbf{s}\rangle = tb'_p \otimes b'_q = \lfloor Q_\otimes/t \rceil Z^m + e_{pq}$. The algorithm is correct.

The running time is dominated by the cost of tensoring the ring elements, which takes time $\Theta(pq)$.

4.3 Evaluating Inner Products in Exponents

This procedure allows evaluation of inner products in exponents with $\log d$ times less homomorphic additions in exponents than in FHEW, also less key material.

As a subroutine, we construct an (External) Multiply-and-Add operation in the exponent, for a public coefficient $\alpha \in \mathbb{Z}_d^*$. We defer the error analysis of this step to the next algorithm (Theorem 2) with $\ell = 1$.

External Multiply-and-Add in the Exponent.

Key Material: Key-switch keys $\mathbf{S}^\alpha$ (from $\psi_\alpha(s)$ to s) and $\mathbf{S}^\beta$ (from $\psi_\beta(s)$ to s), where $\beta = \alpha^{-1} \mod d$ (Size: $O(Kd \log Q)$)
Runtime: $O(Kd \log d)$
Signature: $\mathsf{ExtExpMultAdd}^{\alpha}_{\mathbf{S}^\alpha, S^\beta}$:

$$\mathcal{R}_d\mathsf{LWE}^{t|Q}_s(T^{m'};\, E) \times \mathcal{R}_d\mathsf{GSW}^{t|Q}_s(T^{m};\, E') \to \mathcal{R}_d\mathsf{LWE}^{t|Q}_s\left(T^{\alpha m + m'};\, E''\right).$$

where $E'' = \sqrt{E^2 + d^2K(4\sigma^2 + E'^2)}$.

Algorithm 5 $\mathsf{ExtExpMultAdd}^{\alpha}_{\mathbf{S}^\alpha,\mathbf{S}^\beta}(\mathbf{c}, \mathbf{C})$

```
Require: $\alpha \in \mathbb{Z}_d^*$, with inverse $\beta = \alpha^{-1} \in \mathbb{Z}_d^*$
  A $\psi_\alpha(s) \to s$ Key-Switching key $\mathbf{S}^\alpha \in \mathcal{R}_d\mathsf{LWE}^{Q|Q}_s\left(\psi_\alpha(s)\cdot \mathbf{g}^T;\, \sigma\right)$
  A $\psi_\beta(s) \to s$ Key-Switching key $\mathbf{S}^\beta \in \mathcal{R}_d\mathsf{LWE}^{Q|Q}_s\left(\psi_\beta(s)\cdot \mathbf{g}^T;\, \sigma\right)$
  A ciphertext $\mathbf{c} \in \mathcal{R}_d\mathsf{LWE}^{t|Q}_s(T^{m'};\, E)$. A ciphertext $\mathbf{C} \in \mathcal{R}_d\mathsf{GSW}^{t|Q}_s(T^{m};\, E')$
Ensure: A ciphertext $\mathbf{c}' \in \mathcal{R}_d\mathsf{LWE}^{t|Q}_s(T^{\alpha m + m'};\, E'')$.

  $\mathbf{c}_1 \leftarrow \mathsf{Galois}^\beta(\mathbf{c})$                                   ▷ $\in \mathcal{R}_d\mathsf{LWE}^{t|Q}_{\psi_\beta(s)}(T^{\beta m'})$
  $\mathbf{c}_2 \leftarrow \mathsf{KeySwitch}^{\psi_\beta(s)\to s}_{\mathbf{S}^\beta}(\mathbf{c}_1)$           ▷ $\in \mathcal{R}_d\mathsf{LWE}^{t|Q}_s(T^{\beta m'})$
  $\mathbf{c}_3 \leftarrow \mathsf{ExtMult}(\mathbf{C}, \mathbf{c}_2)$                             ▷ $\in \mathcal{R}_d\mathsf{LWE}^{t|Q}_s(T^{m+\beta m'})$
  $\mathbf{c}_4 \leftarrow \mathsf{Galois}^\alpha(\mathbf{c}_3)$                                 ▷ $\in \mathcal{R}_d\mathsf{LWE}^{t|Q}_{\psi_\alpha(s)}(T^{\alpha m+m'})$
  $\mathbf{c}_5 \leftarrow \mathsf{KeySwitch}^{\psi_\alpha(s)\to s}_{\mathbf{S}^\alpha}(\mathbf{c}_4)$           ▷ $\in \mathcal{R}_d\mathsf{LWE}^{t|Q}_s(T^{\alpha m+m'})$
  return $\mathbf{c}_5$.
```

Remark 4. A similar speed-up was obtained in [10] using a different technique, namely a *Mux* operation. We are unfortunately unable to use it in our circulant set-up, essentially because encryptions of 0 are not allowed: our ind-cpa-security guarantee (Lemma 4) only applies to encryptions of X^m for some $m \in \mathbb{Z}_d$. Yet our technique is more general, precisely, we do not restrict the secret input vector to have binary coefficients.

By chaining, this allows us to evaluate inner products $\langle \mathbf{x}, \mathbf{y} \rangle$ over $\mathbb{Z}_d$ in the exponent, given GSW encryptions $\mathcal{R}_d\mathsf{GSW}^{t|Q}_s(T^{x_i})$ and a public vector of coefficients $\mathbf{y} \in \mathbb{Z}_d^\ell$.

External Inner-product in the Exponent.

Key Material: Key-switch keys $\mathbf{S}^\alpha$ from $\psi_\alpha(s)$ to s, for every $\alpha \in \mathbb{Z}_d^*$. (Size: $O(d^2 \log^2 Q)$)
Runtime: $O(lKd \log d)$
Signature: $\mathsf{ExtExpInner}^{\mathbf{y}}_{[\mathbf{S}^\alpha]_\alpha}$:

$$\bigoplus_{i=1}^{\ell} \mathcal{R}_d\mathsf{GSW}^{t|Q}_s(T^{x_i};\, E') \to \mathcal{R}_d\mathsf{LWE}^{t|Q}_s\left(T^{\langle \mathbf{x},\mathbf{y}\rangle};\, \sqrt{2K\ell^2 d^2\sigma^2 + 2K\ell d^2 E'^2}\right). \qquad (10)$$

Algorithm 6 $\mathsf{ExtExpInner}^{\mathbf{y}}_{[\mathbf{S}^\alpha]_\alpha}([\mathbf{C}_i]_{i\in[l]})$

```
Require: A public vector $\mathbf{y} \in \mathbb{Z}_d^\ell$
  A $\psi_\alpha(s) \to s$ Key-Switching key $\mathbf{S}^\alpha \in \mathcal{R}_d\mathsf{LWE}_s^{Q|Q}\left(\psi_\alpha(s)\cdot \mathbf{g}^T;\, \sigma\right)$ for each $\alpha \in \mathbb{Z}_d^*$
  A ciphertext $\mathbf{C}_i \in \mathcal{R}_d\mathsf{GSW}_s^{t|Q}(T^{x_i};\, E')$ for each $i \in [\ell]$
Ensure: A ciphertext $\mathbf{c} \in \mathcal{R}_d\mathsf{LWE}_s^{t|Q}\left(T^{\langle \mathbf{x},\mathbf{y}\rangle};\, \sqrt{4K\ell^2d^2\sigma^2 + 2K\ell d^2E'^2}\right)$ if the error terms in the $\mathbf{C}_i$ are
  independent.

  $\mathbf{c} \leftarrow (0, T^0)$                                        ▷ $\in \mathcal{R}_d\mathsf{LWE}_s^{t|Q}(T^0; 0)$
  for $i$ from 1 to $\ell$ where $\mathbf{y}_i \neq 0$ do
      $\alpha = \mathbf{y}_i$; $\beta = \alpha^{-1} \bmod d$
      $\mathbf{c} \leftarrow \mathsf{ExtExpMultAdd}^{\alpha}_{\mathbf{S}^\alpha,\mathbf{S}^\beta}(\mathbf{c}, \mathbf{C}_i)$
                          ▷ $\in \mathcal{R}_d\mathsf{LWE}_s^{t|Q}\left(T^{\sum_{j=1}^i \mathbf{x}_j\mathbf{y}_j}, \sqrt{4Ki^2d^2\sigma^2 + 2Kid^2E'^2}\right)$
  end for
  return $\mathbf{c}$
```

Theorem 2. *Algorithm 6 is correct and runs in time $\Theta(lKd\log d)$.*

Proof. By induction, we prove that the error term of $\mathbf{c}$ in the i-th iteration of the for-loop is of the form $e_1 + e_2$ where e_1 is $(2id\sqrt{K}\sigma)$-subgaussian, and $e_2 = \sum_{j=1}^i \left\langle \mathbf{d}^{(j)}, \psi_{\mathbf{y}_j}(\mathbf{e}^{(j)})\right\rangle$ with the following properties: $\mathbf{e}^{(j)}$ is the error vector of $\mathbf{C}_j$, and $\mathbf{d}^{(j)} \in \mathcal{R}_d^{2K}$ is a random vector with $|\mathbf{d}_n^{(j)}| \le d$ that is independent of $\mathbf{e}^{(k)}$ for all $k \ge j$.

Clearly, our claim holds prior to the loop (with $i = 0$) since $\mathbf{c}$ has no error term at this point. Suppose now that the claim holds for $i-1$. Let $\alpha = \mathbf{y}_i$ and $\beta = \alpha^{-1} \bmod d$. During the $\mathsf{ExtExpMultAdd}$ operation, we first apply a Galois operation, which results in an error term of $\psi_\beta(e_1) + \psi_\beta(e_2)$. This is followed by a key-switch, which, by Lemma 6, changes the error to $\psi_\beta(e_1) + \psi_\beta(e_2) + e_{ks,1}$ where $e_{ks,1}$ is independent of $\mathbf{e}_j$ for all j, and subgaussian with parameter $\sqrt{K}d\sigma$. Next comes an $\mathsf{ExtMult}$ operation which changes it to $X^k\psi_\beta(e_1) + X^k\psi_\beta(e_2) + X^k e_{ks,1} + \left\langle \mathbf{d}, \mathbf{e}^{(i)}\right\rangle$ for some k, where $\mathbf{e}$ is the error in $\mathbf{C}_i$, and $\mathbf{d} \in \mathcal{R}_d^{2K}$ is a random vector independent of $\mathbf{e}^{(j)}$ for $j \ge i$ which satisfies $|\mathbf{d}_n| \le d$ for every n, by Lemma 7. After the second Galois and key-switch, the error term becomes $X^{\alpha k}e_1 + X^{\alpha k}e_2 + X^{\alpha k}\psi_\alpha(e_{ks,1}) + \psi_\alpha\left(\left\langle \mathbf{d}, \mathbf{e}^{(i)}\right\rangle\right) + \psi_\alpha(e_{ks,2})$ where $e_{ks,2}$ is again subgaussian with parameter $\sqrt{K}d\sigma$. We can reorder the error terms and write

$$\underbrace{X^{\alpha k}e_1 + X^{\alpha k}\psi_\alpha(e_{ks,1}) + \psi_\alpha(e_{ks,2})}_{e_1'} + \underbrace{X^{\alpha k}e_2 + \psi_\alpha(\langle \mathbf{d}, \mathbf{e}^{(i)}\rangle)}_{e_2'}$$

By the induction hypothesis and since $e_{ks,1}$ and $e_{ks,2}$ are subgaussian with parameter $d\sqrt{K}\sigma$, it follows that e_1' is $(2id\sqrt{K}\sigma)$-subgaussian (because we do not assume that e_1, $e_{ks,1}$ and $e_{ks,2}$ are independent). Finally, it holds that

$$X^{\alpha k}e_2 = \sum_{j=1}^{i-1}\left\langle X^{\alpha k}\mathbf{d}^{(j)}, \psi_{\mathbf{y}_j}\left(\mathbf{e}^{(j)}\right)\right\rangle$$

and thus, setting $\mathbf{d}'^{(j)} = X^{\alpha k}\mathbf{d}^{(j)}$ for $j < i$ and $\mathbf{d}'^{(i)} = \psi_{\mathbf{y}_i}(\mathbf{d})$, we have $e_2' = \sum_{j=1}^i \left\langle \mathbf{d}'^{(j)}, \psi_{\mathbf{y}_j}(\mathbf{e}^{(j))}\right\rangle$ which completes the induction step. Finally, by repeated applications of Lemma 2, we conclude that the error term in the output is subgaussian with parameter $\sqrt{4K\ell^2d^2\sigma^2 + 2K\ell d^2E'^2}$.

It is easy to see that the algorithm has the claimed runtime by adding up the runtimes of the algorithms used in ExtExpMultAdd.

Remark 5. The asymmetry in the error parameter $\sqrt{4K\ell^2 d^2\sigma^2 + 2K\ell d^2 E'^2}$ with ℓ^2 on the left-hand side and ℓ on the right is due to the fact that key-switch keys can be reused in multiple loop iterations. Thus, the error parameter that we state in Algorithm 6 represents the worst case where we have the same α in every loop iteration, and $\alpha = \alpha^{-1} \bmod d$. In practice, this will happen very rarely, so we can expect an error parameter close to $\sqrt{K\ell d^2\big(4\sigma^2 + 2E'^2\big)}$.

5 Joining the building blocks

In this section, we explain how the building blocks we described in Section 4 fit together to form the homomorphic evaluation and bootstrapping procedure EvalBootstrap. See Fig. 1 for a schematic overview. We build an algorithm that, given ciphertexts $\mathbf{c}_i \in \mathsf{LWE}_{\mathbf{s}}(m_i; E_{\text{in}})$, $i \in \{1, \ldots, k\}$ with $\mathbf{s} \in \mathbb{Z}_{Q'}^p$ a short vector (i.e., $\mathbf{s}_i \in \{-1, 0, 1\}$ for all i), a function $f : \mathbb{Z}_t \to \mathbb{Z}_t$, and coefficients $\gamma_1, \ldots, \gamma_k \in \mathbb{Z}_t$ such that $\sum_i |\gamma_i| \le t$, produces $\mathbf{c} \in \mathsf{LWE}_{\mathbf{s}}(f(m); E_{\text{out}})$ where $m = \sum_{i=1}^k \gamma_i m_i$. We do *not* assume that the error terms in the $\mathbf{c}_i$ are independent of each other, or independent of the key material used by EvalBootstrap: if an input $\mathbf{c}_i$ is the result of a previous application of EvalBootstrap, then its error term is not independent of the error terms in the bootstrapping/evaluation key material. We use the following parameters for the building blocks:

- n as the security parameter,
- $p, q = \Theta(n)$, $Q = \text{poly}(n)$, $K = \lceil \log Q \rceil = O(\log n)$, $t = \Theta(n)$ such that $t \le \sqrt{pq}/4$,
- $\lambda = \Theta(n)$ such that $\lambda \le q$ as the failure parameter; the decryption and homomorphic evaluation procedures should only fail with probability exponentially small in λ,
- σ as the error parameter used in the key material,
- $Q', Q_\otimes = O(Q/\sqrt{n\sigma})$, with $t \cdot \lfloor Q_\otimes / t \rceil = 1 \bmod Q_\otimes$.

For $m_i \in \{0, 1\}$, the algorithm can evaluate arbitrary k-bit gates if $t \ge 2^k$, using $\gamma_i = 2^{i-1}$ and an appropriately chosen f. We can compute a threshold gate if $t > k$ by setting $\gamma_i = 1$ for all i.

Theorem 3. *Algorithm 7 is correct and runs in time $\tilde{O}(n^2)$. Moreover, there exists $Q = O(\gamma'|f|n^{6.5}\sigma^{1.5})$ such that the output of* EvalBootstrap *can be used as input for another execution of* EvalBootstrap *with coefficients $\gamma_1', \ldots, \gamma_k'$ such that $\gamma' = \sum_i |\gamma_i'|$ (with failure probability exponentially small in n).*

Proof. It is straighforward to verify the error parameters for each step in the comments of the algorithm. There are two steps where failures might occur: the ExpCRT step, and the FunExpExtract step. The failure probability for ExpCRT is $2\exp(-\lambda)$. FunExpExtract will not fail to extract the value of F, but if the error term in $\mathbf{c}$ is too large, the output might not be an encryption of $f(m)$. The subgaussian tail estimate guarantees that the failure probability is at most $2\exp(-\lambda)$ if $\sqrt{r^2\gamma^2 E_{\text{in}}^2 + p + 1} \le pq/(2t\sqrt{2\lambda})$ where $r = \lfloor pq/t \rceil \,/\, \lfloor Q'/t \rceil$. Since $t \le \sqrt{pq}/4$ and $\lambda \le q$, this condition is satisfied if $\sqrt{r^2\gamma^2 E_{\text{in}}^2 + p + 1} \le \sqrt{2p}$, or equivalently,

$$\gamma E_{\text{in}} \le \underbrace{\sqrt{\frac{p-1}{r^2}}}_{T} = \Theta\left(\sqrt{\frac{Q'^2}{pq^2}}\right) = \Theta\left(\frac{Q}{n^2\sqrt{\sigma}}\right)$$

Algorithm 7 $\mathsf{EvalBootstrap}_{\mathcal{S}}^{f,\gamma_1,\ldots,\gamma_k}(\mathbf{c}_1,\ldots,\mathbf{c}_k)$: Homomorphically evaluate a function and produce a bootstrapped encryption of the result.

Require: $\mathbf{c}_i \in \mathsf{LWE}_{\mathbf{s}}^{t|Q'}(m; E_{\text{in}})$, $f : \mathbb{Z}_t \to \mathbb{Z}_t$, $\gamma_i \in \mathbb{Z}_t$ where $\gamma E_{\text{in}} \le T$ for a certain $T = \Theta(Q/(n^2\sqrt{\sigma}))$, and $\mathcal{S}$ is the required public key material consisting of:

- Bootstrapping keys $\mathbf{BK}_i^{(d)} \in \mathcal{R}_d\mathsf{LWE}_{s^{(d)}}^{t|Q}(\mathbf{s}_i \bmod d; \sigma)$ for $i = 1,\ldots,n$ and $d = p, q$, where $|s^{(d)}|+1 \le \mu$
- Key-switch keys $\mathbf{S}^{d,\alpha}$ from $\psi_\alpha(s)$ to s for $d \in \{p,q\}$ and $\alpha \in \mathbb{Z}_d^*$
- A key-switch key $\mathbf{S}$ from $\mathbf{s}^{(pq)}$ to s' where $\mathbf{s}^{(pq)} = \big(-\psi_\alpha(s^{(p)}) \otimes \psi_\beta(s^{(q)}), \psi_\alpha(s^{(p)}) \otimes 1, 1 \otimes \psi_\beta(s^{(q)})\big)$ for $\alpha = q^{-1} \bmod p$, and $\beta = p^{-1} \bmod q$, and $s' = \sum_{i=0}^{p-1} \mathbf{s}_{i+1} X^i$

Ensure: $\mathbf{c} \in \mathsf{LWE}_{\mathbf{s}}^{t|Q'}\Big(f\big(\sum_{i=1}^k \gamma_i m_i\big); E_{\text{out}}\Big)$ where $E_{\text{out}} = O\big(|f| n^{4.5}\sigma\big)$, except with probability exponentially small in n.

```
$\mathbf{c} \leftarrow \sum_{i=1}^k \mathbf{c}_i$    $\triangleright \in \mathsf{LWE}_{\mathbf{s}}^{t|Q'}(m; \gamma E)$
$\mathbf{c} \leftarrow \mathsf{ModSwitch}^{Q'\to pq}(\mathbf{c})$    $\triangleright \in \mathsf{LWE}_{\mathbf{s}}^{t|pq}(m; \sqrt{r^2\gamma^2E^2 + (p+1)^2})$ where $r = \lfloor pq/t \rceil \,/\, \lfloor Q'/t \rceil$
$\Big(\mathbf{a}^{(p)}, b^{(p)}\Big) \leftarrow \mathbf{c} \bmod p$
$\Big(\mathbf{a}^{(q)}, b^{(q)}\Big) \leftarrow \mathbf{c} \bmod q$
$\mathbf{c}^{(p)} \leftarrow X^{b^{(p)}} \cdot \mathsf{ExtExpInner}_{[\mathbf{S}^{p,\alpha}]_\alpha}^{-\mathbf{a}^{(p)}}\Big(\big[\mathbf{BK}_i^{(p)}\big]_i\Big)$
    $\triangleright \in \mathcal{R}_p\mathsf{LWE}_s^{t|Q}\big(X^{b-\langle \mathbf{a},\mathbf{s}\rangle \bmod p}; O\big(n^{2.5}\sigma\big)\big)$
$\mathbf{c}^{(q)} \leftarrow Y^{b^{(q)}} \cdot \mathsf{ExtExpInner}_{[\mathbf{S}^{q,\alpha}]_\alpha}^{-\mathbf{a}^{(q)}}\Big(\big[\mathbf{BK}_i^{(q)}\big]_i\Big)$
    $\triangleright \in \mathcal{R}_q\mathsf{LWE}_s^{t|Q}\big(Y^{b-\langle \mathbf{a},\mathbf{s}\rangle \bmod q}; O\big(n^{2.5}\sigma\big)\big)$
$\mathbf{c}^{(p)} \leftarrow \mathsf{ModSwitch}^{Q\to Q_\otimes}\big(\mathbf{c}^{(p)}\big)$    $\triangleright \in \mathcal{R}_p\mathsf{LWE}_{s^{(p)}}^{t|Q_\otimes}\Big(X^{b-\langle \mathbf{a},\mathbf{s}\rangle \bmod p}; O\big(n\sqrt{\sigma}\big)\Big)$
$\mathbf{c}^{(q)} \leftarrow \mathsf{ModSwitch}^{Q\to Q_\otimes}\big(\mathbf{c}^{(q)}\big)$    $\triangleright \in \mathcal{R}_q\mathsf{LWE}_{s^{(q)}}^{t|Q_\otimes}\Big(Y^{b-\langle \mathbf{a},\mathbf{s}\rangle \bmod q}; O\big(n\sqrt{\sigma}\big)\Big)$
$\mathbf{c}^{(pq)} \leftarrow \mathsf{ExpCRT}\big(\mathbf{c}^{(p)}, \mathbf{c}^{(q)}\big)$    $\triangleright \in \mathcal{R}_{pq}\mathsf{LWE}_{\mathbf{s}^{(pq)}}^{t|Q_\otimes}\big(Z^{b-\langle \mathbf{a},\mathbf{s}\rangle}; O\big(n^{3.5}\sigma\big)\big)$
$\mathbf{c}^{(pq)} \leftarrow \mathsf{ModSwitch}^{Q_\otimes\to Q'}\big(\mathbf{c}^{(pq)}\big)$    $\triangleright \in \mathcal{R}_{pq}\mathsf{LWE}_{\mathbf{s}^{(pq)}}^{t|Q'}\big(Z^{b-\langle \mathbf{a},\mathbf{s}\rangle}; O\big(n^{3.5}\sigma\big)\big)$
$F \leftarrow (x \mapsto f(\lfloor tx/q \rceil \bmod t))$    $\triangleright F : \mathbb{Z}_{pq} \to \mathbb{Z}_t$, $|F| = |f|pq/t = O(|f|n)$

$\mathbf{c} \leftarrow \mathsf{FunExpExtract}_{F,\mathbf{S}}^{\mathbf{s}^{(pq)}\to\mathbf{s}}\big(\mathbf{c}^{(pq)}\big)$    $\triangleright \in \mathsf{LWE}_{\mathbf{s}}^{Q'|t}\big(f(m); O\big(|f|n^{4.5}\sigma\big)\big)$
**return** $\mathbf{c}$
```

The runtime is dominated by ExtExpInner and FunExpExtract, which run in time $O(nKd\log d)$ and $O(Kpq\log(pq))$, respectively. Given our asymptotic parameter choices, both of those are $\tilde{O}(n^2)$.

If we want to use outputs of EvalBootstrap as inputs for another execution of EvalBootstrap, where the absolute values of the coefficients sum up to γ', we require that $\gamma' E_{\text{out}} \leq T$. From the asymptotic formulas for E_{out} and T, it is easy to see that this inequality can be satisfied by a Q in $O(|f|\gamma' n^{6.5}\sigma^{1.5})$.

6 Implementation

In addition to the formal analysis, we developed a complete implementation of the scheme. Our objective was to make it efficient and usable. We present below the key techniques that enable us to evaluate a 6-bit gate in roughly 6.4 seconds on a laptop.

6.1 Implementation details

FFT The most intensive computations throughout the scheme are the multiplications of ring elements. For efficiency this is classically done in the *frequency domain*. The cost for a multiplication decreases from $\Theta(n^c)$ down to $\Theta(n\log n)$, where $c = \log(3)$ in the case of Karatsuba algorithm for example. Since we are dealing with circulant ring elements, we may wish to run the FFT operation in the ring dimension exactly. But our ring dimensions are prime, which is the worst case for FFT efficiency. We ran some benchmarks and it turned out that it was much faster to use a bigger dimension (with small prime factors), and do the polynomial reduction afterwards. Also we do not meet the conditions to apply NTT (our moduli are not primes), so our choice was to stick with FFT computations and we use the FFTW library [14] for the forward and backward transforms.

More challenges arose with FFT computations since our biggest modulus is $Q = 2^{56}$ and the FFT works with *double precision* numbers (i.e. 53 bits mantissa). So we have to split the ring coefficients into two halves of 28 bits each and apply the FFT transformation on each to prevent rounding errors. We perform this splitting trick only when needed, ie when the ring element is not small. For example, in ExtMult products of ring element are computed where one of the operands is the output of a Gadget decomposition. This operand needs not be split before FFT forward transform because it is very small.

Pre-computations In order to minimize the evaluation time of the gate, a maximum of heavy computations are done in the setup phase. Consequently all keys materials: bootstrapping keys, key-switching keys, among others, are computed ahead of time and in FFT domain. Our `CirculantRing` class allows to transparently manipulate ring element in FFT or coefficient representation which greatly contribute to both performance and code readability.

Further optimization The implementation has been done in C++11, using its most convenient and efficient features. For example, all classes are extensively defined with template parameters (dimension, moduli, basis decomposition...). This trick allows the computer to know, at compile time, the values of many variables (eg. loop ranges). The compiler then produces dedicated and highly optimized binaries.

On our benchmark laptop with `gcc` 7.1.1, we also tweaked the optimization flags to save 5% more time than the usual `-O2`, which already saves 75%.

Open-source Many efforts have also been made for general availability and usability. The whole code is documented with Doxygen and many unitary tests are provided. With under 4,000 lines of code, it remains accessible to whoever wants to tweak or improve it. The implementation will be made publicly available and open-source. It is attached as *auxiliary material* of this submission.

At the first start (and only then), heavy computations are performed by the FFTW components, in order to optimize the FFT for the current computer.

6.2 Parameters

For our first implementation, we targeted a 6-bit input gate. The parameters of the scheme are as follows:

– For 6 input bits, the plaintext modulus $t = 2^6$.
– The ring dimensions p and q are 1439 and 1447, so $pq = 2,077,892$.
  Hence the FFT dimensions are $d_1^{FFT} = 3072 = 3 \cdot 2^{10}$ for $\mathcal{R}_p$,$\mathcal{R}_q$ and $d_2^{FFT} = 4,194,304 = 2^{22}$ for $\mathcal{R}_{pq}$.
– The modulus in $\mathsf{ExtExpInner}$ and the LWE are $Q, Q' = 2^{56}$.
– Errors and secrets are sampled according to Section A.3. Secrets are ternary, one third of the coefficients are set to -1, another to 1 and the rest to 0. Errors have variance 4.

For ExpCRT we want a small inverse to $\lfloor Q_\otimes/t \rceil \mod Q_\otimes$. Hence we choose $Q_\otimes = (2^{19} - t + 1)^2$. Finally, for the gadget decomposition we use $B = 2^8$ and $K = 7$ for $\mathsf{ExtExpInner}$ and $\mathsf{FunExpExtract}$ and their key material.

We also have extra parameters related an to optimization presented in Appendix B.1. Namely, we apply an extra $\mathsf{KeySwitch}$ over LWE ciphertext to decrease the length l of the decryption inner-product from $l = p = 1439$ down to $l = 600$. This key-switch happens with modulus $Q = 2^{56}$, error standard deviation 2^{33}, and gadget parameters $B = 2^6$, $K = 10$.

Error growth and Correctness To choose the parameters, we simulated the error growth throughout the gate, using heuristic error propagation assumption, described in Appendix B.2. This simulation script is provided with the code as file `scripts/parameters.sage`. We compared the predicted variance of each step to the experimental one, and found them to be corroborated. From the final variance, and according to a central limit heuristic, we predict a failure probability of only 2^{-74} for the above parameter set. In practice we have tested our scheme hundreds of time on different inputs, and never observed failure.

Security To estimate the concrete security of our parameter set, we use the `lwe-estimator` from Albrecht [2]. All the LWE instances behind our LWE, $\mathcal{R}_p\mathsf{LWE}$, $\mathcal{R}_q\mathsf{LWE}$ ciphertexts given as part of the evaluation key offers at least 100 bits of security, according to the estimator as of commit `cc5f6e8`, which includes the latest result of [1] for small secrets. Therefore we feel safe to claim at least 80 bits of security.

6.3 Performances

We run our test on a punchy laptop: Core i7-6500U (2.50 GHz, 4MB L2 cache), 16 GB RAM with a GNU/Linux Fedora 26 installed on a SSD. The computation is single-threaded and we got the following timings:

– FFTW *wisdom* computation (only once per computer): 68 minutes
– Key pre-processing (once per user key pair): 38 seconds
– 6-bit input, 1-bit output gate evaluation: 6.4 seconds

The gate time breaks down into: 0.60 s per ExtExpInner (the two could be run in parallel), 4.0 s for the KeySwitch in FunExpExtract and only 0.55 s for the output bit related operations. Consequently, computing another function (1 more output bit) on the same 6 input bits would add only 0.55 s, and so on. For 6-to-6 bit gate it yields just above 10 seconds. On the memory front, we need 9.2 GB of RAM to store all key materials for the computation.

Optimisations This first implementation includes only those on ExtExpInner described in Section B.1. Over the total gate evaluation time, 60% (3.8 s) are spent on FFT forward and backward transforms. The 3.8 seconds break down into 0.9 sec for more than 350k FFT in dimension d_1^{FFT} ($\mathcal{R}_p$ and $\mathcal{R}_q$), and 2.9 sec for only around 250 FFT in dimension d_2^{FFT} for $\mathcal{R}_{pq}$. We estimate that the optimisations of Appendix B.4 will bring these 2.9 seconds down to 1 or 1.5 second at most. This rough estimate is based on partial implementation, soon to be confirmed after complete integration. The overall gate time should drop below 6 seconds and the cost of additional output bits become negligible.

References

1. Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany.
2. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015. `http://eprint.iacr.org/2015/046`.
3. Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.
4. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany.
5. David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *18th ACM STOC*, pages 1–5, Berkeley, CA, USA, May 28–30, 1986. ACM Press.
6. Jean-François Biasse and Luis Ruiz. FHEW with efficient multibit bootstrapping. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 119–135, Guadalajara, Mexico, August 23–26, 2015. Springer, Heidelberg, Germany.
7. Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 278–291, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Heidelberg, Germany.
8. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. Cryptology ePrint Archive, Report 2011/344, 2011. `http://eprint.iacr.org/2011/344`.
9. Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Provably weak instances of ring-LWE revisited. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 147–167, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
10. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

11. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Improving TFHE: faster packed homomorphic operations and efficient circuit bootstrapping. Cryptology ePrint Archive, Report 2017/430, 2017. `http://eprint.iacr.org/2017/430`.
12. Léo Ducas and Alain Durmus. Ring-LWE in polynomial rings. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 34–51, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany.
13. Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
14. Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".
15. Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.
16. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.
17. Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P. Smart. Field switching in BGV-Style homomorphic encryption. Cryptology ePrint Archive, Report 2012/240, 2012. `http://eprint.iacr.org/2012/240`.
18. Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P. Smart. Ring switching in BGV-style homomorphic encryption. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 19–37, Amalfi, Italy, September 5–7, 2012. Springer, Heidelberg, Germany.
19. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.
20. Shai Halevi, Tzipora Halevi, Victor Shoup, and Noah Stephens-Davidowitz. Implementing BP-obfuscation using graph-induced encoding. Cryptology ePrint Archive, Report 2017/104, 2017. `http://eprint.iacr.org/2017/104`.
21. Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 641–670, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
22. Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140, San Francisco, CA, USA, April 13–17, 2003. Springer, Heidelberg, Germany.
23. Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pages 267–288. Springer, 1998.
24. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.
25. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.
26. Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.
27. Chris Peikert. How (not) to instantiate ring-LWE. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 411–430, Amalfi, Italy, August 31 – September 2, 2016. Springer, Heidelberg, Germany.

28. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.
29. John Riordan and Claude E Shannon. The number of two-terminal series-parallel networks. *Studies in Applied Mathematics*, 21(1-4):83–93, 1942.
30. Omar Rivasplata. Subgaussian random variables: An expository note, 2012. `https://sites.ualberta.ca/~omarr/publications/subgaussians.pdf`.
31. Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 420–443, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany.
32. Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. In Yonina Eldar and Gitta Kutyniok, editors, *Compressed Sensing, Theory and Applications.*, pages 210–268. Cambridge University Press, 2012.

A More details on Circulant LWE

A.1 Circulant LWE and reduction to Ring-LWE

In all this subsection, we assume d to be prime. It is well known that the naive decisional version of Ring-LWE is insecure over circulant rings, simply by exploiting the CRT decomposition $\mathcal{R}_d/Q\mathcal{R}_d \simeq \tilde{\mathcal{R}}_d/Q\tilde{\mathcal{R}}_d \times \mathbb{Z}/Q\mathbb{Z}$ when Q is coprime to d, and mounting an attack on the $\mathbb{Z}/Q\mathbb{Z}$ part (projecting to this part corresponds to evaluating the polynomial at 1, and therefore maintain smallness of the error). However, this does not mean that such rings are inherently insecure: The NTRU cryptosystems [23, 22] use circulant rings, choosing the secret key and errors that evaluate to a fixed known value (say 0) at 1.

This suggests a strategy to construct a variant of Ring-LWE over circulant rings that would be as secure as the cyclotomic Ring-LWE, simply by lifting all elements $\tilde{x} \in \tilde{\mathcal{R}}_d/Q\tilde{\mathcal{R}}_d$ to $x \simeq (\tilde{x}, 0)$, yet this reverse CRT operation may not keep small elements small.

Instead, one can construct such a lift without working modulo Q, in order to preserve smallness of coefficients (up to some reasonable distortion). We also note that such a lift should actually start from the co-different ideal $\tilde{\mathcal{R}}_d^\vee$, so as to match the Ring-LWE instances admitting worst-case hardness proofs [24], yet a reduction (with some loss on the error parameter) to Ring-LWE without the co-different was given in [12].

Because $1 - X$ and $\Phi_d(X)$ are *not coprime* over $\mathbb{Z}[X]$ (their gcd is d, not 1), we *do not* have a CRT decomposition of $\mathcal{R}_d$ as $\tilde{\mathcal{R}}_d \times \mathbb{Z}$. Yet, those polynomials are coprime over $\mathbb{Q}[X]$ which allows to write

$$K_d = \tilde{K}_d \times \mathbb{Q}$$

where $K_d = \mathbb{Q}[X]/(X^d - 1)$ and $\tilde{K}_d = \mathbb{Q}[X]/\Phi_d(X)$.[11] We write L the canonical inclusion map $L : \tilde{K}_d \to K_d$, which is explicitly given by

$$L : \sum_{i=0}^{d-1} a_i X^i \mapsto \sum_{i=0}^{d-1} a_i X^i - \frac{1}{d}(\sum_{i=0}^{d-1} a_i)(\sum_{i=0}^{d-1} X^i).$$

Note that the above formula can be extended to a $\mathbb{Q}$-linear map $K_d \to K_d$, viewing $\tilde{K}_d$ as a subspace of K_d according to the above isomorphism $K_d = \tilde{K}_d \times \mathbb{Q}$. This extension of L is the

[11] While $\tilde{K}_d$ is a field, K_d is only a ring, but we keep this notation for coherence.

projection orthogonal to the all-1 vector in coefficient representation. Unfortunately the image $L(\tilde{\mathcal{R}}_d)$ is not included in $\mathcal{R}_d$: the projection does not maintain integrality of coefficients. Yet, one notes that a small ideal $\mathfrak{I} \subset \tilde{\mathcal{R}}_d$ does have an integer lift: namely, the ideal $\tilde{\mathfrak{I}} = (1-X)\mathcal{R}_d$ satisfies $L(\tilde{\mathfrak{I}}) \subset \mathcal{R}_d$. Moreover, for $a \in \tilde{\mathfrak{I}}$, it holds that $\sum a_i = 0$, in particular L preserves sizes of elements of $\tilde{\mathfrak{I}}$.

Also consider the lift L taken modulo Q (assuming Q is coprime to d), simply replacing $\frac{1}{d} \in \mathbb{Q}$ by the inverse of d in $\mathbb{Z}/Q\mathbb{Z}$, denoted by L_Q. Consider a Ring-LWE sample as defined in [12]: $(\tilde{a}, \tilde{b} = \tilde{a}\tilde{s} + \tilde{e}) \in (\tilde{\mathcal{R}}/Q\tilde{\mathcal{R}})^2$ for small $\tilde{s}, \tilde{e} \in \mathcal{R}$. We lift this sample to $\mathcal{R}/Q\mathcal{R}$:

$$a = L_Q(\tilde{a}), b = L_Q((1-X)\tilde{b}). \tag{11}$$

We define $s = L((1-X)\tilde{s})$ and $e = L((1-X)\tilde{s})$, and it holds that $s = L_Q((1-X)\tilde{s}) \mod Q$ and $e = L_Q((1-X)\tilde{e}) \mod Q$ since s and e are integral. Therefore,

$$\begin{aligned} b &= L_Q((1-X)\tilde{a} \cdot \tilde{s} + (1-X)\tilde{e}) \\ &= L_Q(\tilde{a}) \cdot L_Q((1-X)\tilde{s}) + L_Q((1-X)\tilde{e}) \\ &= L_Q(\tilde{a})s + e \bmod Q \\ &= as + e \bmod Q \end{aligned}$$

We also note that s, e are still small since the operator norm of $1-X$ is less than 2: these Circulant-LWE samples are useful.

It remains to explain what this transformation does to uniform samples $(\tilde{a}, \tilde{b}) \in (\tilde{\mathcal{R}}/Q\tilde{\mathcal{R}})^2$. Assume that Q is coprime to d, it then holds that Q and $(1-X)$ are coprimes over the integral ring $\tilde{\mathcal{R}}_d$. Therefore, the multiplication by $1-X$ over $(\tilde{\mathcal{R}}/Q\tilde{\mathcal{R}})$ is a bijection, so the sample $(\tilde{a}, (1-X)\tilde{b}) \in (\tilde{\mathcal{R}}/Q\tilde{\mathcal{R}})^2$ is also uniform in $(\tilde{\mathcal{R}}/Q\tilde{\mathcal{R}})^2$. Finally, the lift L_Q is injective, so the final sample $(a, b) \in (\mathcal{R}/Q\mathcal{R})^2$ is uniform over $(L_Q(\tilde{\mathcal{R}}/Q\tilde{\mathcal{R}}))^2$. One easily characterizes the image $L_Q(\tilde{\mathcal{R}}/Q\tilde{\mathcal{R}})$ of L_Q as the set $\mathcal{S}_{d,Q} = \{\sum_{i=0}^{d-1} a_i X^i \mid \sum a_i = 0 \bmod Q\}$ of elements of $\mathcal{R}/Q\mathcal{R}$ whose coefficients sums to 0 modulo Q.

Lemma 11 (Hardness of Circulant-LWE). *Assume that d is prime, and Q is coprime to d. If it is hard to distinguish samples $(\tilde{a}_i, \tilde{b}_i = \tilde{a}_i\tilde{s} + \tilde{e}_i) \in (\tilde{\mathcal{R}}/Q\tilde{\mathcal{R}})^2$ from uniform where $\tilde{e}_i$ are independent random variables drawn from a distribution ψ, then the samples $(a_i = L_Q(\tilde{a}_i), b_i = L_Q((1-X)\tilde{b}_i) \in \mathcal{S}_{d,Q}^2 \subset (\mathcal{R}/Q\mathcal{R})^2$ are also hard to distinguish from uniform samples in $\mathcal{S}_{d,Q}^2$.*

A.2 Security proofs for CLWE and CGSW Encryption schemes

Lemma 12 (Restatement of Lemma 4). *If the decisional $\tilde{\mathcal{R}}$-* LWE *problem is hard, then the Circulant-LWE scheme is cpa-secure for messages of the form $m = X^k$.*

Proof. If $\tilde{\mathcal{R}}$- LWE is hard, then by Lemma 11, the Circulant-LWE distribution is indistinguishable from the uniform distribution over $\mathcal{S}_{d,Q}^2$. To prove cpa-security, it suffices to show that, for any $k \in \mathbb{Z}/d\mathbb{Z}$ and $u = \lfloor Q/t \rceil$, we have $\mathcal{S}_{d,Q} + uX^k = \mathcal{S}_{d,Q} + u$. This then shows that a Circulant-LWE encryption of $m = X^k$ is indistinguishable from a uniformly random sample from $\mathcal{S}_{d,Q} \times (\mathcal{S}_{d,Q} + u)$. Indeed, $\mathcal{S}_{d,Q} + u = \{\sum_{i=0}^{d-1} a_i X^i \mid \sum a_i = u \bmod Q\} = \mathcal{S}_{d,Q} + uX^k$.

Lemma 13 (Restatement of Lemma 5). *If the decisional $\tilde{\mathcal{R}}$-* LWE *problem is hard, then the Circulant-GSW scheme is cpa-secure.*

Proof. Let $\mathbf{C}$ be a Circulant-GSW ciphertext. Each row of $\mathbf{C}$ is of the form $(a,b)+(0,uB^i m)$ or $(a,b)+(uB^i m,0)$ where $m=X^k$ and (a,b) is a Circulant-LWE sample, and thus indistinguishable from a random element of $\mathcal{S}_{d,Q}^2$. By the same argument as in the previous proof, each row of $\mathbf{C}$ is indistinguishable from a uniformly random samble from either $(\mathcal{S}_{d,Q}+uB^i)\times\mathcal{S}_{d,Q}$, or $\mathcal{S}_{d,Q}\times(\mathcal{S}_{d,Q}+uB^i)$ where i only depends on the row number, not on m.

A.3 Simpler error distribution in CLWE for practice

In practice, most FHE schemes do not follow precisely the Ring-LWE problem definition admitting reduction to worst-case problem [24, 25]. For example, HElib [21] uses Ring-LWE with spherical errors in the coefficient embedding, and very sparse ternary secrets, and ignoring the co-different ideal $\mathcal{R}^\vee$. The TFHE scheme [10] also relies on Ring-LWE with ternary secrets, which is not know to reduce to the regular Ring-LWE. Cutting such corners appears quite crucial to error growth management and therefore efficiency. We will follow this approach, and define adjust the distributions as follows.

- we proceed to sample secrets and error isotropically in $S_{d,Q}$, while the above reduction leads to errors with a distortion factor $(1-X)$. This distortion seems to be an artefact of the proof, as it breaks symmetries: one could choose a different way of breaking those symmetries by replacing $1-X$ by $1-X^e$ for any e coprime to d. Respecting the symmetries seems a better idea in the light of recent analysis [9, 27].
  This variant could also be proved secure (with a loss of a constant factor about $\sqrt{2}$ on the size of the error), simply by adding more noise to make it spherical again, using the convolution lemma of [26], but this would drag us away from the topic of this paper.
- we choose to use ternary secrets s, which, as in previous schemes leads to serious performance improvements due to smaller error growth. It has recently been showed that such choices make lattice attacks somewhat faster [1], especially when s is very sparse: we will account for this refined analysis when measuring the concrete security of our proposed parameters.

Sampling of a. We sample a uniform in $\mathcal{R}_d/(Q\mathcal{R}_d)$ under the constraint $a(1) \mod Q = 0$ by choosing all the coefficients a_i at random for $i \geq 1$, and setting $a_0 = -\sum_{i>0} a_i \mod Q$.

Sampling of s. When d is prime, we sample a a ternary s of density $\delta = 2/3$ by choosing exactly $\lfloor \delta d/2 \rfloor$ coefficients set to 1 and $\lfloor \delta d/2 \rfloor$ coefficients set to -1. This implies that $s(1) = 0$, and $\|s\|^2 = 2\lfloor \delta d/2 \rfloor$. Indeed, we find it preferable to fix its length to avoid sampling sparse keys that would be subtentially weaker.

Sampling of e. We wish to sample errors e with variance σ in a way that ensures $e(1) = 0$. We set:

$$e = \sum_{i=0}^{\sigma^2 d/2} T^{a_i} - T^{b_i},$$

where the a_i's and b_i's' are independant uniform exponents modulo d. One note that this distribution is invariant by permutation over $\{1, T, \dots, T^{d-1}\}$: we have preserved the symmetries of the ring. Note that this procedure would get rather slow for large σ, yet we won't exceed $\sigma \leq 8$ in our parameter choices.

Remark 6. The above procedure would not be adapted for composite degree d, as more care is required to construct a lift as done in section A.1. Yet, while we will make use of circulant ring $\mathcal{R}_d$ with composite degree $d = pq$, we will never directly construct ciphertexts over that ring. Indeed, the ciphertext in $\mathcal{R}_d$ will be publicly constructed by tensoring two ciphertexts from $\mathcal{R}_p$ and $\mathcal{R}_q$, and are therefore no easier to decrypt than the original ciphertexts over $\mathcal{R}_p$ and $\mathcal{R}_q$.

B Optimizations

In this section, we present some optimization of the scheme for practice. Our implementation does include the optimizations from Sections B.1, B.2 and B.3. We left out the optimization from Section B.4, which requires substential modifications to our code base.

B.1 Accelerating ExtExpInner

Factoring Galois-KeySwitch *sequences* We note that it is possible to factor some operations when chaining $\mathsf{ExtExpMultAdd}^{\alpha}$ and $\mathsf{ExtExpMultAdd}^{\alpha'}$, by applying $\mathsf{Galois}^{\alpha\beta'}$ rather than Galois^{α} followed by Galois^{β} (together with the appropriate Key Switches), cf Fig. 2.

Furthermore, if $\mathbf{y} \in \mathbb{Z}_d^{\ell}$ contains repeated values, it is possible to re-index the inner product to make equal values contiguous, and skip useless Galois^1 operations. Those tricks also decrease the final error E by constant factors.

Pushing this trick to its limits, if ℓ is large enough, one could re-index the inner product so that the $\alpha\beta'$ all belong to a small[12] subset $\mathbb{Z}_d^*$, allowing to decrease the size of the key material. In combination with the following optimization, this should lead to reduce the overall key-size by a significant factor.

Decreasing LWE *dimension* In our theoretical scheme, the homomorphic inner product in exponent operation is done over vectors of length $\ell = p + 1$ where p is the dimension of of the secret in the LWE scheme.

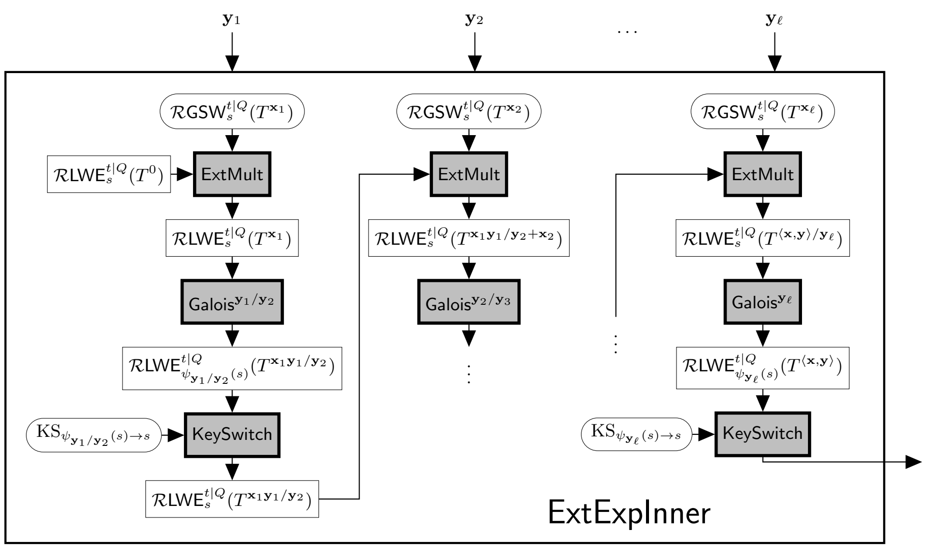
In practice, we remark that this dimension is quite larger than needed for security, given the amount of noise and the modulus pq of those ciphertexts. We therefore proceed with an extra LWE key-switch just the combination of the LWE ciphertexts. In practice it allows to decrease the dimension by a factor between 2 and 3, which accelerates the ExtExpInner operations by the same factor. As a small added bonus, it also slightly decreases the error in the ciphertexts outputted by this function.

B.2 Heuristic error propagation

Our theoretical analysis of the scheme used sub-gaussian analysis [32] to provide bounds on error propagation that are already significantly better than worst-case bounds. Yet those bounds are asymptotic, without explicit constants, and for some operations may not be perfectly tight. As in previous work [13, 10], when it comes to choose practical parameters, we rely on a tighter but heuristic analysis of error propagation, essentially treating all random variables as independent gaussians. More precisely, considering that the critical random variable for correctness is obtained as the sum of many random variables, we only compute its variance as the sum of the variance of its terms, and treat this final result as Gaussian in accordance with the central limit Theorem (which is formally not applicable due to potential dependencies).

[12] of size roughly $d/\ell + 2$ assuming the public vector $\mathbf{y} \in \mathbb{Z}_d^{\ell}$ is uniformly random.

Fig. 2. Optimized ExtExpInner (External Inner Product in Exponent) overview



$\mathbf{y}_1$
$\mathbf{y}_2$
$\mathbf{y}_\ell$
$\mathcal{R}\mathsf{GSW}_s^{t|Q}(T^{\mathbf{x}_1})$
$\mathcal{R}\mathsf{GSW}_s^{t|Q}(T^{\mathbf{x}_2})$
$\mathcal{R}\mathsf{GSW}_s^{t|Q}(T^{\mathbf{x}_\ell})$
$\mathcal{R}\mathsf{LWE}_s^{t|Q}(T^0)$
ExtMult
$\mathcal{R}\mathsf{LWE}_s^{t|Q}(T^{\mathbf{x}_1})$
$\mathcal{R}\mathsf{LWE}_s^{t|Q}(T^{\mathbf{x}_1\mathbf{y}_1/\mathbf{y}_2+\mathbf{x}_2})$
$\mathcal{R}\mathsf{LWE}_s^{t|Q}(T^{\langle\mathbf{x},\mathbf{y}\rangle/\mathbf{y}_\ell})$
$\mathsf{Galois}^{\mathbf{y}_1/\mathbf{y}_2}$
$\mathsf{Galois}^{\mathbf{y}_2/\mathbf{y}_3}$
$\mathsf{Galois}^{\mathbf{y}_\ell}$
$\mathcal{R}\mathsf{LWE}_{\psi_{\mathbf{y}_1/\mathbf{y}_2}(s)}^{t|Q}(T^{\mathbf{x}_1\mathbf{y}_1/\mathbf{y}_2})$
$\mathcal{R}\mathsf{LWE}_{\psi_{\mathbf{y}_\ell}(s)}^{t|Q}(T^{\langle\mathbf{x},\mathbf{y}\rangle})$
$\mathrm{KS}_{\psi_{\mathbf{y}_1/\mathbf{y}_2}(s)\to s}$
$\mathrm{KS}_{\psi_{\mathbf{y}_\ell}(s)\to s}$
KeySwitch
$\mathcal{R}\mathsf{LWE}_s^{t|Q}(T^{\mathbf{x}_1\mathbf{y}_1/\mathbf{y}_2})$
ExtExpInner

Linear Operations. For the linear operations Add, Mult and Galois operations, we use the same equations (3), (8) as in our sub-gaussian analysis, since it is tight in this case, but apply it to the standard deviation of each variable rather than the sub-gaussianity parameter.

Modulus Switching. For our analysis, we needed to randomize the rounding step to ensure sub-gaussianity without resorting to the randomness of the input ciphertext. Instead, in practice we use deterministic rounding and account for the randomness of the input ciphertext. Treating the rounding errors as independent uniform random variables in the interval $[-1/2, 1/2]$ allows to heuristically improve the error bound (4) down to

$$\mathsf{ModSwitch} : \mathcal{R}_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m;\, E) \to \mathcal{R}_d\mathsf{LWE}_{\mathbf{s}}^{t|Q'}\left(m;\, \sqrt{\frac{Q'^2}{Q^2}E^2 + \frac{\|\mathbf{s}\|^2}{12}}\right) \tag{12}$$

Key Switching, External Multiplication and Inner Product in the Exponent. We first note that, according to Remark 2, the bounds given by (5) and (6) must be amended to account for the use of a Gadget matrix in base B rather than in base 2. Additionally, we note that this bound accounts for the worst output of $\mathbf{G}^{-1}$. Instead, we treat the output of $\mathbf{G}^{-1}$ as a uniform random vectors with coordinates uniform in the integer interval $I_B = \{-\left\lfloor \frac{B-1}{2} \right\rfloor, \ldots, \left\lceil \frac{B-1}{2} \right\rceil\}$. Each such coordinate has variance $V_B = \frac{1}{B}\sum_{i\in I_B} i^2 \approx B^2/12$.

For our heuristic analysis, we therefore amend (5) to

$$\mathsf{KeySwitch} : \mathcal{R}_d\mathsf{LWE}_{\mathbf{s}}^{t|Q}(m;\, E) \to \mathcal{R}_d\mathsf{LWE}_{s'}^{t|Q}\left(m;\, \sqrt{E^2 + \sigma^2 dnKV_B}\right). \tag{13}$$

Similarly, (6) is heuristically changed to

$$\mathsf{ExtMult} : \mathcal{R}_d\mathsf{LWE}_s^{t|Q}(T^m;\, E) \times \mathcal{R}_d\mathsf{GSW}_s^{t|Q}(T^{m'};\, E') \\ \rightarrow \mathcal{R}_d\mathsf{LWE}_s^{t|Q}\left(T^{m+m'};\, \sqrt{E^2 + E'^2 dKV_B}\right). \quad (14)$$

Note that assuming independence decreased the factor d^2 to a factor d. Similarly, a factor $4\ell^2$ can be decreased to 2ℓ, ignoring the potential dependences discussed in Remark 5. The trick described in section B.1 further decreases this 2ℓ factor to ℓ.

In conclusion, the accumulated error in the error propagation of the whole $\mathsf{ExtExpInner}$ operation (10) is now heuristically given by:

$$\mathsf{ExtExpInner} : \bigoplus_{i=1}^{\ell} \mathcal{R}_d\mathsf{GSW}_s^{t|Q}(T^{x_i};\, E') \rightarrow \mathcal{R}_d\mathsf{LWE}_s^{t|Q}\left(T^{\langle \mathbf{x},\mathbf{y}\rangle};\, \sqrt{dK\ell V(\sigma^2 + E'^2)}\right). \quad (15)$$

Tensoring Looking only at the variance of individual coefficient, one may save the factor $\sqrt{2\lambda}$ in the error propagation of ExpCRT, namely, (9) becomes:

$$\mathsf{ExpCRT} : \mathcal{R}_p\mathsf{LWE}_{s_p}^{t|Q'}(X^{m_p};\, E_p) \times \mathcal{R}_q\mathsf{LWE}_{s_q}^{t|Q'}(Y^{m_q};\, E_q) \\ \rightarrow \mathcal{R}_{pq}\mathsf{LWE}_{\mathbf{s}}^{t|Q'}\left(Z^m;\, \sqrt{E_p^2 + E_q^2 + t^2E_p^2E_q^2}\right). \quad (16)$$

We could successfully confirm all these heuristic equations by measuring the actual errors in our implementation.

B.3 Amortising $\mathsf{FunExpExtract}$

The costly steps of the $\mathsf{FunExpExtract}$ algorithm consist in computing

$$c^{(pq)} \mapsto \mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathcal{R}_p}(f \cdot \mathbf{G}^{-T}(c^{(pq)}) \cdot \mathbf{S})$$

where f represent the function F to extract, $\mathbf{S}$ is a Key-Switching Key (See Figure 1 and Algorithm 7). We note here that the most expensive part of the computation $\mathbf{G}^{-T}(c^{(pq)}) \cdot \mathbf{S}$ can be re-used for up to several different f's.

This amortization allows to extend our technique so that not only the input of the function is large, but also its output.

B.4 Accelerating $\mathsf{FunExpExtract}$

As mentioned above, the practical cost of the $\mathsf{FunExpExtract}$ step as described in Section 4 is prohibitive. The costly steps consist in the computation of

$$\mathrm{Tr}^*_{\mathcal{R}_{pq}/\mathcal{R}_p}(f \cdot \mathbf{G}^{-T}(\mathbf{x} \otimes \mathbf{y}) \cdot \mathbf{S})$$

where f represent the function F to extract, $\mathbf{x}$, $\mathbf{y}$ are the ciphertexts outputted by $\mathsf{ExtExpInner}$, and $\mathbf{S}$ is a Key-Switching Key. Naively, even using precomputations of f and $\mathbf{S}$, this operation would require $4K + 1$ FFT's in dimension pq: one forward FFT for each component of $\mathbf{G}^{-1}(\mathbf{c})$, and one FFT backward.[13] We here show how to get completly rid of those large FFT's, requiring only small FFT's (dimension p and q) and a few additions of vectors of dimension pq.

[13] This is assuming the FFT can handle numbers of bit-size $\Theta(\log(n))$. In practice more FFT at `double` precision will be needed to avoid numerical errors.

FFT of pure tensors. To tackle these costly FFT operations, one should first note that FFT and $\otimes$ can be commuted. Indeed, one may first rewrite $x \otimes y = (x \otimes 1) \cdot (1 \otimes y)$, and note that the FFT coefficients of $x \otimes 1 \in \mathcal{R}_{pq}$ are easily derived from the FFT coefficients of $x \in \mathcal{R}_p$ by simply repeating the coefficients q times (and similarly for $1 \otimes y$). This remark allows us to decrease the naive cost of the FFT operation over pure tensors from $\Theta(pq \log pq)$ to $\Theta(pq + p \log p + q \log q)$.

The CRT-Gadget. To provide an asymptotic improvement for gadget inversion of pure tensors, we need to rely on a different Gadget matrix construction, based on the Chinese-Remainder Theorem. We describe it over the integers $\mathbb{Z}$, yet it naturally extends coefficient-wise to any ring $\mathcal{R}_d$.

Consider a modulus Q such that we can write $Q = \prod_{i=1}^{K} q_i$ where the q_i are small coprime integers. Consider the CRT isomorphism $\mu : r \in \mathbb{Z}_Q \mapsto (r \bmod q_1, \ldots, r \bmod q_K)$, and let $\mathbf{g} \in \mathbb{Z}_Q^K$ be the vector of the *Bezout coefficients*, i.e., the coefficients such that $\mu^{-1}(\mathbf{x}) = \mathbf{x}^T\mathbf{g} \bmod Q$. This gadget also permits to efficiently find small pre-images. Indeed, define: $\mathbf{g}^{-T}(x) = (x_1, \ldots x_K) \in \mathbb{Z}^K$ where x_i is the representative of $x \bmod q_i$ in the range $(-q_i/2, q_i/2]$.

Gadget inversion of pure tensors (in FFT format). This new gadget has the advantage that gadget inversion is somewhat homomorphic. Let us write $\odot$ for the coefficient-wise product of vectors. While in general we have $\mathbf{g}^{-T}(xy) \neq \mathbf{g}^{-T}(x) \odot \mathbf{g}^{-T}(y)$, it nevertheless holds that

$$(\mathbf{g}^{-T}(x) \odot \mathbf{g}^{-T}(y))\mathbf{g} = xy \bmod Q.$$

It also hold that $\mathbf{g}^{-T}(x) \odot \mathbf{g}^{-T}(y)$ is rather small, namely, its i-th coefficient has absolute value less than $q_i^2/4$. This will allow us, at the cost of increased error propagation, to swap the gadget-inversion and the tensoring.

More precisely, we define

$$\mathbf{g}_{\otimes}^{-T}(x, y) = (\mathbf{g}^{-T}(x)_i \otimes \mathbf{g}^{-T}(y)_i)_{i=1\ldots k},$$

and note that it is a proper gadget inversion: $\mathbf{g}_{\otimes}^{-T}(x, y)\mathbf{g} = x \otimes y \bmod Q$, and the coefficients of $\mathbf{g}_{\otimes}^{-T}(x, y)_i$ are less than $q_i^2/4$.

For inputs $(x, y) \in \mathcal{R}_p \times \mathcal{R}_q$ One may compute $g_{\otimes}^{-T}(x, y)$ in FFT format in time $\Theta(Kpq + Kp \log p + Kq \log q)$, that is in time linear in the size of the output. Indeed, one may compute each $(\mathbf{g}^{-T}(x)_i, \mathbf{g}^{-T}(y)_i)$, convert them to FFT format, and then only perform the tensoring step using the remark above. In comparison, the naive algorithm would have cost $\Theta(Kpq \log pq)$: asymptotically, our new trick improves the complexity by a logarithmic factor $\Theta(\log pq)$. The impact in practice may quite substantial also considering the large hidden constants in FFT operations.

Tracing down in the FFT domain. At last, we note that the trace operation $\mathrm{Tr}^{*}_{\mathcal{R}_{pq}/\mathcal{R}_p}$ can also be performed directly in the FFT domain in time $\Theta(pq)$ by summing the appropriate FFT coefficients. The allows to replace the final large backward FFT (in dimension pq) by a cheap backward FFT in dimension p. The cost of this step decreases form $\Theta(pq \log pq)$ down to $\Theta(pq + p \log p)$.