Somewhat Practical Fully Homomorphic Encryption [⋆]

Junfeng Fan and Frederik Vercauteren

Katholieke Universiteit Leuven, COSIC & IBBT
Kasteelpark Arenberg 10
B-3001 Leuven-Heverlee, Belgium
`firstname.lastname@esat.kuleuven.be`

Abstract. In this paper we port Brakerski's fully homomorphic scheme based on the Learning With Errors (LWE) problem to the ring-LWE setting. We introduce two optimised versions of relinearisation that not only result in a smaller relinearisation key, but also faster computations. We provide a detailed, but simple analysis of the various homomorphic operations, such as multiplication, relinearisation and bootstrapping, and derive tight worst case bounds on the noise caused by these operations. The analysis of the bootstrapping step is greatly simplified by using a modulus switching trick. Finally, we derive concrete parameters for which the scheme provides a given level of security and becomes fully homomorphic.

1 Introduction

Fully homomorphic encryption (FHE) allows evaluation of arbitrary functions on encrypted data, and as such has a myriad of potential applications such as private cloud computing. Gentry [7, 8] was the first to show that FHE is theoretically possible. His construction consisted of three parts: first, construct an encryption scheme that is somewhat homomorphic, i.e. that can evaluate functions of limited complexity (think low degree), secondly, simplify the decryption function of this scheme as much as possible (so called squashing), thirdly, evaluate this simplified decryption function homomorphically to obtain ciphertexts with a fixed inherent noise size (so called bootstrapping).

The first variants [21, 18, 20, 4, 5] of Gentry's scheme all followed the same structure and as such had to make additional security assumptions to enable the squashing step. More recent schemes [9, 4, 3, 2] avoid squashing all together and can bootstrap by evaluating the real decryption circuit. Another advantage of the more recent schemes is that their security is based on the Learning with Errors (LWE) problem [17] or its ring variant RLWE [15], the hardness of which can be related to classical problems on (ideal) lattices.

[⋆] This work was supported in part by the Research Council K.U.Leuven: GOA TENSE (GOA/11/007), by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

All existing schemes have the common trait that they add a small "noise" component during encryption. Computing homomorphically on ciphertexts will cause these noises to grow up to the point when they become so large that decryption fails. The bootstrapping approach by Gentry can then be used to lower the noise in a ciphertext to a fixed level determined by the complexity of the decryption circuit. Especially the noise growth caused by homomorphic multiplication has been the major obstacle to designing efficient schemes. In the first generation of schemes, the noises themselves multiplied upon each homomorphic multiplication leading to a doubly exponential growth in the depth of the circuit, i.e. evaluating a depth n circuit on clean ciphertexts (with noise E) resulted in a noise E^{2^n}. A first major improvement was proposed in [3] leading to a noise level of only E^n for a depth n circuit. The most recent scheme [2] further improves on this in that the noise for each multiplication level grows with a constant factor independent of the noise present in the ciphertext, i.e. the noise for a depth n circuit is $E \cdot c(\lambda)^n$ where $c(\lambda)$ is a constant that depends on the security parameter.

As to whether any of these proposals is really practical, the answer is simply "no". There have been several attempts [18, 10, 19, 16, 11] at implementing most of the schemes mentioned, but none of them comes even close to being practical. The most recent paper [11] manages to execute one AES encryption homomorphically in eight days using a massive amount (tens of GBs) of RAM memory. Of course, compared to the very first proposals, there has been a major advance in efficiency.

The main contribution of this paper is its simplicity due to our down to earth approach, whereby any excess mathematical machinery (beautiful it may be) has been left out. Other contributions of this paper are as follows: we port the scheme by Brakerski [2] from the LWE setting to the RLWE setting, which in itself is rather trivial. We provide a detailed, but simple analysis of the various homomorphic operations, such as multiplication, relinearisation and bootstrapping, and derive tight worst case bounds on the noise caused by these operations. Using a simple modulus switching trick we simplify the analysis of the bootstrapping step. Combining this with a practical security analysis of the scheme following [14], we finally obtain concrete parameters for a fully homomorphic scheme with a given security level.

Although this paper is not about optimising the various subroutines, we do provide two versions of relinearisation that are more efficient than the approach taken in [2]. Whenever applicable, we will mention existing optimisations that remain valid for the proposed scheme. In a follow up paper we will consider the real practicality of this scheme by implementing it both in software and hardware, which will show that the title is indeed justified.

The remainder of the paper is organised as follows: Section 2 briefly recalls notation and some background on probability. Section 3 reviews a very elegant encryption scheme based on RLWE, which will be used as the basis for the somewhat homomorphic scheme described in Section 4. Section 5 analyses the bootstrapping step and determines the minimal depth at which the somewhat homomorphic scheme can be made fully homomorphic. Section 6 uses the analysis of Lindner and Peikert [14] to derive parameters for a fully homomorphic scheme with a given security level. Finally, Section 7 concludes the paper and highlights work in progress.

2 Preliminaries

2.1 Basic Notation

The basic object we will work with is the polynomial ring $R = \mathbb{Z}[x]/(f(x))$ where $f(x) \in \mathbb{Z}[x]$ is a monic irreducible polynomial of degree d. In practice one would typically restrict to using a cyclotomic polynomial $\Phi_m(x)$, i.e. the minimal polynomial of the primitive m-th roots of unity. The most popular choice for expository purposes is to take $f(x) = x^d + 1$ with $d = 2^n$.

Elements of the ring R will be denoted in lowercase bold, e.g. $\mathbf{a} \in R$. The coefficients of an element $\mathbf{a} \in R$ will be denoted by a_i, i.e. $\mathbf{a} = \sum_{i=0}^{d-1} a_i \cdot x^i$. The infinity norm $||\mathbf{a}||$ is defined as $\max_i |a_i|$ and the expansion factor of R is defined as $\delta_R = \max\{||\mathbf{a} \cdot \mathbf{b}||/(||\mathbf{a}|| \cdot ||\mathbf{b}||) : \mathbf{a}, \mathbf{b} \in R\}$.

Let $q > 1$ be an integer, then by $\mathbb{Z}_q$ we denote the *set* of integers $(-q/2, q/2]$. Note that we really simply consider $\mathbb{Z}_q$ to be a set, and as such should not be confused with the ring $\mathbb{Z}/q\mathbb{Z}$. Similarly, we denote with R_q the set of polynomials in R with coefficients in $\mathbb{Z}_q$. For $a \in \mathbb{Z}$ we denote by $[a]_q$ the unique integer in $\mathbb{Z}_q$ with $[a]_q = a \bmod q$. In very few places we will need reduction in the interval $[0, q)$, which will be denoted as $r_q(a)$ (remainder modulo q).

Similarly, when $\mathbf{a} \in R$, we denote by $[\mathbf{a}]_q$ the element in R obtained by applying $[\cdot]_q$ to all its coefficients. For $x \in \mathbb{R}$, we use $\lfloor x \rceil$ to denote rounding to the nearest integer and $\lfloor x \rfloor, \lceil x \rceil$ to indicate rounding up or down. Given an integer n, we denote by size(n) its bit size, i.e. size$(n) = \lceil \log_2(n + 0.5) \rceil$. With $n[i]$ we denote the i-th bit (counting from 0) in the bit-expansion of $|n|$.

Note that all arithmetic takes place in R itself and in many cases even (temporarily) in $\mathbb{Q}[x]/(f(x))$. When implementing the scheme presented in this paper one therefore has to take care precisely when the result of a computation can be reduced modulo some integer q.

2.2 Probability

Given a probability distribution $\mathcal{D}$, we use $x \leftarrow D$ to denote that x is sampled from $\mathcal{D}$. For a set S, $x \leftarrow S$ denotes that x is sampled uniformly from S. A distribution χ over the integers is called B-bounded if it is supported on $[-B, B]$.

The discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ over the integers is the probability distribution that assigns a probability proportional to $\exp(-\pi|x|^2/\sigma^2)$ to each $x \in \mathbb{Z}$. We note that $D_{\mathbb{Z},\sigma}$ is statistically indistinguishable from a B-bounded distribution for large enough B, e.g. in practice one could take $B = 10 \cdot \sigma$.

The discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ is then used to define a distribution χ on R. The distribution χ is in general not as simple as just sampling the coefficients according to $D_{\mathbb{Z},\sigma}$. However, for the polynomial $f(x) = x^d + 1$ with d a power of 2, we can indeed define χ as $D^d_{\mathbb{Z},\sigma}$. For more general cyclotomic polynomials, sampling from χ is only slightly more involved. Recall that for the normal distribution $\mathcal{N}(0, \sigma^2)$, we have that $\mathrm{Prob}_{x \leftarrow \mathcal{N}(0,\sigma^2)}[|x| > k \cdot \sigma] = \mathrm{erf}(k/\sqrt{2})$. As such define the function $\beta(\epsilon) := \min\{\beta \mid \mathrm{erf}(\beta/\sqrt{2}) < \epsilon\}$, then with probability $1 - \epsilon$ the samples are bounded by $\beta \cdot \sigma$. For instance, if we set $\epsilon = 2^{-64}$, then it suffices to take $\beta(\epsilon) > 9.2$.

3 RLWE-based Encryption

In this section we recall a very simple and elegant encryption scheme based on the RLWE problem introduced by Lyubashevsky, Peikert and Regev [15].

3.1 RLWE Problem

The RLWE problem is simply a ring based version of the LWE problem [17] and is formulated as follows.

Definition 1 (Decision-RLWE). *For security parameter λ, let $f(x)$ be a cyclotomic polynomial $\Phi_m(x)$ with $\deg(f) = \varphi(m)$ depending on λ and set $R = \mathbb{Z}[x]/(f(x))$. Let $q = q(\lambda) \geq 2$ be an integer. For a random element $\mathbf{s} \in R_q$ and a distribution $\chi = \chi(\lambda)$ over R, denote with $A^{(q)}_{\mathbf{s},\chi}$ the distribution obtained by choosing a uniformly random element $\mathbf{a} \leftarrow R_q$ and a noise term $\mathbf{e} \leftarrow \chi$ and outputting $(\mathbf{a}, [\mathbf{a} \cdot \mathbf{s} + \mathbf{e}]_q)$. The Decision-RLWE$_{d,q,\chi}$ problem is to distinguish between the distribution $A^{(q)}_{\mathbf{s},\chi}$ and the uniform distribution $U(R_q^2)$.*

The RLWE problem can be reduced (using a quantum algorithm) to the shortest vector problem over ideal lattices [15]. Furthermore, one can restrict $\mathbf{s}$ to be sampled from χ instead of taken uniformly in R_q without any security implications [1, 15]. Finally, we note that the hardness of the problem is independent of the precise shape of q [13] and as such q does not has to be prime and can be taken simply as a power of 2.

3.2 Encryption Scheme

The above decision problem immediately leads to the following encryption scheme as described in the extended version of [15]. The plaintext space is taken as R_t for some integer $t > 1$. Let $\Delta = \lfloor q/t \rfloor$ and denote with $r_t(q) = q \mod t$ then we clearly have $q = \Delta \cdot t + r_t(q)$. We remark that q nor t have to prime, nor that t and q are coprime. The encryption scheme `LPR.ES` is then defined as follows:

- `LPR.ES.SecretKeyGen`(1^λ): sample $\mathbf{s} \leftarrow \chi$ and output $\mathtt{sk} = \mathbf{s}$
- `LPR.ES.PublicKeyGen`$(\mathtt{sk})$: set $\mathbf{s} = \mathtt{sk}$, sample $\mathbf{a} \leftarrow R_q$, $\mathbf{e} \leftarrow \chi$ and output

$$\mathtt{pk} = \left([-(\mathbf{a} \cdot \mathbf{s} + \mathbf{e})]_q, \mathbf{a}\right).$$

- `LPR.ES.Encrypt`$(\mathtt{pk}, \mathbf{m})$: to encrypt a message $\mathbf{m} \in R_t$, let $\mathbf{p}_0 = \mathtt{pk}[0]$, $\mathbf{p}_1 = \mathtt{pk}[1]$, sample $\mathbf{u}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi$ and return

$$\mathtt{ct} = \left([\mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \Delta \cdot \mathbf{m}]_q, [\mathbf{p}_1 \cdot \mathbf{u} + \mathbf{e}_2]_q\right)$$

- `LPR.ES.Decrypt`$(\mathtt{sk}, \mathtt{ct})$: set $\mathbf{s} = \mathtt{sk}$, $\mathbf{c}_0 = \mathtt{ct}[0]$, $\mathbf{c}_1 = \mathtt{ct}[1]$ and compute

$$\left[\left\lfloor \frac{t \cdot [\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s}]_q}{q} \right\rceil\right]_t$$

The above scheme can be shown to be semantically secure assuming the hardness of RLWE given 3 samples [15].

To show that decryption is correct for properly encrypted ciphertexts, we prove the following lemma.

Lemma 1. *Using the notation of the above encryption scheme* `LPR.ES` *and assuming that* $||\chi|| < B$*, we have that*

$$[\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s}]_q = \Delta \cdot \mathbf{m} + \mathbf{v} \tag{1}$$

with $||\mathbf{v}|| \leq 2 \cdot \delta_R \cdot B^2 + B$*. This implies that for* $2 \cdot \delta_R \cdot B^2 + B < \Delta/2$*, decryption works correctly.*

Proof: Simply writing out the definition modulo q gives:

$$\begin{aligned} \mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} &= \mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \Delta \cdot \mathbf{m} + \mathbf{p}_1 \cdot \mathbf{u} \cdot \mathbf{s} + \mathbf{e}_2 \cdot \mathbf{s} \bmod q \\ &= \Delta \cdot \mathbf{m} + \mathbf{e} \cdot \mathbf{u} + \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{s} \bmod q \,. \end{aligned}$$

Since $\Delta \cdot \mathbf{m} + \mathbf{e} \cdot \mathbf{u} + \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{s}$ is already in R_q for small enough error terms, we conclude that $\mathbf{v} = \mathbf{e} \cdot \mathbf{u} + \mathbf{e}_1 + \mathbf{e}_2 \cdot \mathbf{s}$. Since $\mathbf{e}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{u}, \mathbf{s} \leftarrow \chi$, we recover the given bound $||\mathbf{v}|| \leq 2 \cdot \delta_R \cdot B^2 + B$. Write $\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} = \Delta \cdot \mathbf{m} + \mathbf{v} + q \cdot \mathbf{r}$, then if we divide by q and multiply by t, we obtain $\mathbf{m} + (t/q) \cdot (\mathbf{v} - \epsilon \cdot \mathbf{m}) + t \cdot \mathbf{r}$, where $\epsilon = q/t - \Delta = r_t(q)/t < 1$. For the rounding to be correct, we need $(t/q) \cdot ||\mathbf{v} - \epsilon \cdot \mathbf{m}|| < 1/2$ and since $\mathbf{m} \in R_t$, the given bound follows. □

The term $\mathbf{v}$ is called the *noise* contained in the ciphertext and if not clear from the context which ciphertext `ct` it belongs to, it will be denoted as $\mathbf{v}(\mathtt{ct})$. Looking at the exact expression of the noise term, we see that taking $\mathbf{u}$ and $\mathbf{s}$ as small as possible will make the noise smaller. This remark leads to the following optimisation and corresponding assumption.

Optimisation/Assumption 1: Instead of sampling $\mathbf{s}, \mathbf{u} \leftarrow \chi$, we will sample $\mathbf{s}, \mathbf{u}$ from R_2, i.e. the norm $||\mathbf{s}|| = ||\mathbf{u}|| = 1$. We note that the noise terms $\mathbf{e}_1, \mathbf{e}_2$ remain sampled from χ. The bound in Lemma 1 is then replaced by $B \cdot (2 \cdot \delta_R + 1)$. The security implications of this optimisation seem to be minor, at least when we make the assumption that the results for the LWE setting carry over to the RLWE setting. In [12], the authors show that for the standard LWE one can take the secret $\mathbf{s}$ from any distribution, as long as the distribution has sufficient entropy. Assuming that the LWE analysis also holds for the RLWE setting, we can even use a secret with a given low Hamming weight h, as long as $\binom{d}{h}$ is large enough.

4 Somewhat Homomorphic Encryption

In this section we will derive a simple somewhat homomorphic encryption scheme `FV.SH` based on RLWE by using `LPR.ES` as a basis. In fact, `FV.SH` is mostly a simple port to the RLWE stetting of the fully homomorphic scheme by Brakerski [2] based on standard LWE.

The scheme consists of an augmented version of the scheme $\texttt{LPR.ES}$ introduced in the previous section. The generation of the secret key, public key and the encryption/decryption procedures will remain exactly the same bar the fact that we use the optimisation $\mathbf{u}, \mathbf{s} \leftarrow \mathbb{R}_2$. The main addition to $\texttt{LPR.ES}$ is a so called relinearisation key $\texttt{rlk}$ that will be used to compute a homomorphic multiplication.

The main invariant of the scheme $\texttt{LPR.ES}$ is given by Equation (1), namely when we interpret the elements of a ciphertext $\texttt{ct}$ as the coefficients of a polynomial $\texttt{ct}(x)$ and evaluate this polynomial in $\mathbf{s}$ we obtain

$$[\texttt{ct}(\mathbf{s})]_q = \Delta \cdot \mathbf{m} + \mathbf{v}\,,$$

from which we can easily recover the message $\mathbf{m}$. Using this interpretation it is quite easy to derive homomorphic addition $\texttt{FV.SH.Add}$ and multiplication $\texttt{FV.SH.Mul}$.

Addition Let $\texttt{ct}_i$ for $i = 1, 2$ be two ciphertexts, with $[\texttt{ct}_i(\mathbf{s})]_q = \Delta \cdot \mathbf{m}_i + \mathbf{v}_i$, then it is easy to see that

$$[\texttt{ct}_1(\mathbf{s}) + \texttt{ct}_2(\mathbf{s})]_q = \Delta \cdot [\mathbf{m}_1 + \mathbf{m}_2]_t + \mathbf{v}_1 + \mathbf{v}_2 - \epsilon \cdot t \cdot \mathbf{r}\,,$$

where $\epsilon = q/t - \Delta = r_t(q)/t < 1$ and $\mathbf{m}_1 + \mathbf{m}_2 = [\mathbf{m}_1 + \mathbf{m}_2]_t + t \cdot \mathbf{r}$. Note that $||\mathbf{r}|| \leq 1$, which implies that the noise in the sum has grown *additively* by a maximum of t. As such, we can then simply define

$$\texttt{FV.SH.Add}(\texttt{ct}_1, \texttt{ct}_2) := \left([\texttt{ct}_1[0] + \texttt{ct}_2[0]]_q\,, [\texttt{ct}_1[1] + \texttt{ct}_2[1]]_q\right).$$

Multiplication Homomorphic multiplication consists of two steps: the first step is quite easy, and basically consists of multiplying the polynomials $\texttt{ct}_1(x)$ and $\texttt{ct}_2(x)$ together and scaling by t/q. The problem however is that we end up with a ciphertext consisting of 3 ring elements instead of 2. The second step resolves this issue and is called "relinearisation".

Basic Multiplication First write the evaluation of $\texttt{ct}_i(x)$ in $\mathbf{s}$ as an equality in R as follows

$$\texttt{ct}_i(\mathbf{s}) = \Delta \cdot \mathbf{m}_i + \mathbf{v}_i + q \cdot \mathbf{r}_i\,.$$

An easy computation shows that $||\mathbf{r}_i|| < \delta_R \cdot ||\mathbf{s}||$. If we then multiply these expressions together we obtain:

$$\begin{aligned}(\texttt{ct}_1 \cdot \texttt{ct}_2)(\mathbf{s}) = \Delta^2 \cdot \mathbf{m}_1 \cdot \mathbf{m}_2 + \Delta \cdot (\mathbf{m}_1 \cdot \mathbf{v}_2 + \mathbf{m}_2 \cdot \mathbf{v}_1) + q \cdot (\mathbf{v}_1 \cdot \mathbf{r}_2 + \mathbf{v}_2 \cdot \mathbf{r}_1) \\ + \mathbf{v}_1 \cdot \mathbf{v}_2 + q \cdot \Delta \cdot (\mathbf{m}_1 \cdot \mathbf{r}_2 + \mathbf{m}_2 \cdot \mathbf{r}_1) + q^2 \cdot \mathbf{r}_1 \cdot \mathbf{r}_2\,.\end{aligned} \tag{2}$$

The above expression shows that we need to scale with a factor of $1/\Delta$ to be able to recover a ciphertext that encrypts $[\mathbf{m}_1 \cdot \mathbf{m}_2]_t$. However, since Δ does not necessarily divide q, we would get a large noise caused by the rounding error of the last term. As such, we will scale by t/q which solves this rounding issue. Let $\texttt{ct}_1(x) \cdot \texttt{ct}_2(x) = \mathbf{c}_0 + \mathbf{c}_1 \cdot x + \mathbf{c}_2 \cdot x^2$, then we will use the approximation

$$\frac{t}{q} \cdot (\texttt{ct}_1 \cdot \texttt{ct}_2)(\mathbf{s}) = \lfloor t \cdot \mathbf{c}_0/q \rceil + \lfloor t \cdot \mathbf{c}_1/q \rceil \cdot \mathbf{s} + \lfloor t \cdot \mathbf{c}_2/q \rceil \cdot \mathbf{s}^2 + \mathbf{r}_a\,, \tag{3}$$

which introduces an approximation error $\mathbf{r}_a$ of size $< (\delta_R \cdot ||s|| + 1)^2/2$.

If we write $\mathbf{m}_1 \cdot \mathbf{m}_2 = [\mathbf{m}_1 \cdot \mathbf{m}_2]_t + t \cdot \mathbf{r}_m$, then $||\mathbf{r}_m|| < (t \cdot \delta_R)/4$. Similarly, if we write $\mathbf{v}_1 \cdot \mathbf{v}_2 = [\mathbf{v}_1 \cdot \mathbf{v}_2]_\Delta + \Delta \cdot \mathbf{r}_v$, then $||\mathbf{r}_v|| < (E^2 \cdot \delta_R)/\Delta$ where E is a bound on the original noise terms, i.e. $||\mathbf{v}_i|| < E$. By multiplying equation (2) by t/q and grouping terms together we obtain the somewhat complicated looking equality, where we mainly used that $t \cdot \Delta = q - r_t(q)$:

$$\begin{aligned}\frac{t \cdot (\mathtt{ct}_1 \cdot \mathtt{ct}_2)(\mathbf{s})}{q} &= \Delta \cdot [\mathbf{m}_1 \cdot \mathbf{m}_2]_t + (\mathbf{m}_1 \cdot \mathbf{v}_2 + \mathbf{m}_2 \cdot \mathbf{v}_1) + t \cdot (\mathbf{v}_1 \cdot \mathbf{r}_2 + \mathbf{v}_2 \cdot \mathbf{r}_1) \\ &+ \mathbf{r}_v + (q - r_t(q)) \cdot (\mathbf{r}_m + \mathbf{m}_1 \cdot \mathbf{r}_2 + \mathbf{m}_2 \cdot \mathbf{r}_1) + q \cdot t \cdot \mathbf{r}_1 \cdot \mathbf{r}_2 \\ &+ \frac{t}{q} \cdot [\mathbf{v}_1 \cdot \mathbf{v}_2]_\Delta - \frac{r_t(q)}{q} \cdot (\Delta \cdot \mathbf{m}_1 \cdot \mathbf{m}_2 + (\mathbf{m}_1 \cdot \mathbf{v}_2 + \mathbf{m}_2 \cdot \mathbf{v}_1) + \mathbf{r}_v)\,.\end{aligned}$$

The basic idea of writing the expression like this is to make clear which terms will disappear after reduction modulo q and which terms are affected by rounding. Note that in the above expression all terms are integral bar the terms $\mathbf{r}_r$ on the last line. The rounding therefore affects the last line only, and $||\mathbf{r}_r||$ is easily seen to be smaller than $\delta_R \cdot (t + 1/2)^2 + 1/2$. Reducing modulo q and substituting equation (3) then leads to

$$\begin{aligned}\left[\lfloor t \cdot \mathbf{c}_0/q \rceil + \lfloor t \cdot \mathbf{c}_1/q \rceil \cdot \mathbf{s} + \lfloor t \cdot \mathbf{c}_2/q \rceil \cdot \mathbf{s}^2\right]_q &= \Delta \cdot [\mathbf{m}_1 \cdot \mathbf{m}_2]_t + (\mathbf{m}_1 \cdot \mathbf{v}_2 + \mathbf{m}_2 \cdot \mathbf{v}_1) \\ + t \cdot (\mathbf{v}_1 \cdot \mathbf{r}_2 + \mathbf{v}_2 \cdot \mathbf{r}_1) + \mathbf{r}_v - r_t(q) \cdot (\mathbf{r}_m &+ \mathbf{m}_1 \cdot \mathbf{r}_2 + \mathbf{m}_2 \cdot \mathbf{r}_1) + \lfloor \mathbf{r}_r - \mathbf{r}_a \rceil\,.\end{aligned}$$

It is easy to bound the size of the new noise term in the right hand side which finally proves the following lemma.

Lemma 2. *Let ct_i for $i = 1, 2$ be two ciphertexts, with $[ct_i(\mathbf{s})]_q = \Delta \cdot \mathbf{m}_i + \mathbf{v}_i$ and $||\mathbf{v}_i|| < E < \Delta/2$, and let $\mathtt{ct}_1(x) \cdot \mathtt{ct}_2(x) = \mathbf{c}_0 + \mathbf{c}_1 \cdot x + \mathbf{c}_2 \cdot x^2$, then*

$$\left[\lfloor t \cdot \mathbf{c}_0/q \rceil + \lfloor t \cdot \mathbf{c}_1/q \rceil \cdot \mathbf{s} + \lfloor t \cdot \mathbf{c}_2/q \rceil \cdot \mathbf{s}^2\right]_q = \Delta \cdot [\mathbf{m}_1\mathbf{m}_2]_t + \mathbf{v}_3\,,$$

with $||\mathbf{v}_3|| < 2 \cdot \delta_R \cdot t \cdot E \cdot (\delta_R \cdot ||\mathbf{s}|| + 1) + 2 \cdot t^2 \cdot \delta_R^2 \cdot (||\mathbf{s}|| + 1)^2$.

This lemma shows that the noise does not grow quadratically upon multiplication, but is only multiplied roughly by the factor $2 \cdot t \cdot \delta_R^2 \cdot ||\mathbf{s}||$. As such we see that not only t, but especially the norm of the secret $\mathbf{s}$ has a significant influence on the noise growth. By using optimisation 1 again we have $||\mathbf{s}|| = 1$ and thus significantly limit the growth of the noise during multiplication.

Relinearisation Using Lemma 2 we already have a ciphertext that encrypts the multiplication of both plaintexts. However, a remaining problem is that the number of elements in the ciphertext has gone up. To rectify this phenomenon we need a procedure called *relinearisation* that takes a degree 2 ciphertext and reduces it again to a degree 1 ciphertext. It is precisely this step that will require the introduction of a relinearisation key $\mathtt{rlk}$. Let $\mathtt{ct} = [\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2]$ denote a degree 2 ciphertext, then we need to find $\mathtt{ct}' = [\mathbf{c}'_0, \mathbf{c}'_1]$ such that

$$\left[\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} + \mathbf{c}_2 \cdot \mathbf{s}^2\right]_q = \left[\mathbf{c}'_0 + \mathbf{c}'_1 \cdot \mathbf{s} + \mathbf{r}\right]_q\,,$$

where $||\mathbf{r}||$ is small. Since $\mathbf{s}^2$ is not known, a first idea would be to provide a masked version of $\mathbf{s}^2$ as follows (compare with `LPR.ES.PublicKeyGen`): sample $\mathbf{a}_0 \leftarrow R_q$, $\mathbf{e}_0 \leftarrow \chi$ and output $\texttt{rlk} := \left(\left[-(\mathbf{a}_0 \cdot \mathbf{s} + \mathbf{e}_0) + \mathbf{s}^2\right]_q, \mathbf{a}_0\right)$. Note that $\texttt{rlk}[0] + \texttt{rlk}[1] \cdot \mathbf{s} = \mathbf{s}^2 + \mathbf{e}_0$. The problem however is that since $\mathbf{c}_2$ is a random element in R_q, the noise $\mathbf{e}_0$ will be magnified too much resulting in a bad approximation of $\mathbf{c}_2 \cdot \mathbf{s}^2$ and thus causing a huge error $\mathbf{r}$.

Relinearisation: Version 1 A first possible solution is to slice $\mathbf{c}_2$ into parts of small norm by choosing a base T (note that T is totally independent of t) and to write $\mathbf{c}_2$ in base T, i.e. $\mathbf{c}_2 = \sum_{i=0}^{l} T^i \cdot \mathbf{c}_2^{(i)} \bmod q$, with $\ell = \lfloor \log_T(q) \rfloor$ and the coefficients of $\mathbf{c}_2^{(i)}$ are in R_T. The relinearisation key `rlk` then consists of the masked versions of $T^i \cdot \mathbf{s}^2$ for $i = 0, \ldots, \ell$:

$$\texttt{rlk} = \left[\left(\left[-(\mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i) + T^i \cdot \mathbf{s}^2\right]_q, \mathbf{a}_i\right) : i \in [0..\ell]\right].$$

Assumption 2: Note that the relinearisation key `rlk` contains masked versions of $T^i \cdot \mathbf{s}^2$, which are neither real samples of the RLWE distribution, nor real encryptions of $T^i \cdot \mathbf{s}^2$. This fact introduces an extra assumption on our scheme, namely that the scheme is still secure when the adversary has access to `rlk`. This property is a form of weak circular security.

If we now define

$$\mathbf{c}'_0 = \left[\mathbf{c}_0 + \sum_{i=0}^{\ell} \texttt{rlk}[i][0] \cdot \mathbf{c}_2^{(i)}\right]_q \quad \text{and} \quad \mathbf{c}'_1 = \left[\mathbf{c}_1 + \sum_{i=0}^{\ell} \texttt{rlk}[i][1] \cdot \mathbf{c}_2^{(i)}\right]_q, \tag{4}$$

then we can compute

$$\mathbf{c}'_0 + \mathbf{c}'_1 \cdot \mathbf{s} = \mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} + \mathbf{c}_2 \mathbf{s}^2 - \sum_{i=0}^{\ell} \mathbf{c}_2^{(i)} \cdot \mathbf{e}_i \bmod q\,.$$

By applying $[\cdot]_q$ to both sides we finally see that we can take $\mathbf{r} = \sum_{i=0}^{\ell} \mathbf{c}_2^{(i)} \cdot \mathbf{e}_i$. The above derivation also shows that T has the following effects:

- the size of the evaluation key is given by $\ell + 1 \simeq \log_T(q)$, so the greater T is, the smaller `rlk`,
- the number of multiplications in the relinearisation is $2 \cdot \ell \simeq 2 \cdot \log_T(q)$, where one is multiplying an element from R_T with an element from R_q,
- the noise introduced by relinearisation is bounded by $(l+1) \cdot B \cdot T \cdot \delta_R/2$, so greater T causes more noise.

Note that the noise introduced by relinearisation is totally *independent* of the noise inherent in the ciphertext being relinearised. Furthermore, we only need to relinearise after a multiplication (which causes the underlying error to grow as well), so we should choose T at least as large such that the relinearisation error is of similar size as the error contained in a ciphertext after one multiplication. This gives us a minimal value of T that we should consider using. However, when the error has grown large after several multiplications, we can relinearise with respect to T^2 instead of T. Note that all the information required to do this is already contained in the evaluation key `rlk`. We call this approach dynamic relinearisation.

The above strategy minimizes the relinearisation error, but another strategy is to minimize the relinearisation time and space. As such we want to take T very large, for instance $T = \lceil \sqrt{q} \rceil$, since then we only have two slices to take care off. For such large T, the size of the noise after the *first* relinearisation typically will make a huge jump, but all following relinearisations will not cause the noise to increase.

Relinearisation: Version 2 The second possible solution is akin towards some form of "modulus switching" and works as follows. Recall that the problem of simply masking $\mathbf{s}^2$ is that the error term $\mathbf{e}_0$ gets multiplied with $\mathbf{c}_2$ and thus results in a huge error term $\mathbf{r}$. Assume therefore that we do not simply give out the masked version of $\mathbf{s}^2$, but a masked version that can accommodate this extra error. For instance, instead of working modulo q, we could consider giving out a masked version modulo $p \cdot q$ for some integer p. Since we want to obtain an approximation of $\mathbf{c}_2 \cdot \mathbf{s}^2$ modulo q, we will have to scale by p. As such, we have to give out $\texttt{rlk} := ([-(\mathbf{a} \cdot \mathbf{s} + \mathbf{e}) + p \cdot \mathbf{s}^2]_{p \cdot q}, \mathbf{a})$, with $\mathbf{a} \in R_{p \cdot q}$ and $\mathbf{e} \leftarrow \chi'$. Here one has to take care to choose the variance of χ' such that the resulting system is secure. Simply taking $\chi = \chi'$ will result in a considerable loss of security. As shown in Section 6, if we write $p \cdot q = q^k$ for some real $k > 0$ and assume that $||\chi|| < B$, then we require that $||\chi'|| = B_k > \alpha^{1-\sqrt{k}} \cdot q^{k-\sqrt{k}} \cdot B^{\sqrt{k}}$, where α is a constant, e.g. $\alpha \simeq 3.758$.

To obtain a ciphertext corresponding to $\mathbf{c}_2 \cdot \mathbf{s}^2$, we can then simply compute

$$(\mathbf{c}_{2,0}, \mathbf{c}_{2,1}) = \left(\left[\left\lfloor \frac{\mathbf{c}_2 \cdot \texttt{rlk}[0]}{p} \right\rceil \right]_q , \left[\left\lfloor \frac{\mathbf{c}_2 \cdot \texttt{rlk}[1]}{p} \right\rceil \right]_q \right) .$$

An easy computation then shows that $\mathbf{c}_{2,0} + \mathbf{c}_{2,1} \cdot \mathbf{s} = \mathbf{c}_2 \cdot \mathbf{s}^2 + \mathbf{r}$ with

$$||\mathbf{r}|| < \frac{q \cdot B_k \cdot \delta_R}{p} + (\delta_R \cdot ||s|| + 1)/2 .$$

The above formula can be used to easily compute the p that results in a given relinearisation error. For instance, if we want to minimize the relinearisation error and thus demand it to be smaller than the error after one multiplication, we have to choose $p \geq q^3$ in case B is small and not dependent on q. For huge B, e.g. $B \simeq \sqrt{q}$ the situation is worse, since then we require $p \geq q^8$ to obtain minimal error. Note however that again we can apply dynamic relinearisation depending on the noise already present in the ciphertext that needs to be relinearised. Furthermore, if we take $p = b^n$ for some base b, then it is easy to transform `rlk` valid for $p \cdot q$ into an `rlk` valid for $p' \cdot q$ for $p' \mid p$ by rescaling by p'/p.

Definition of FV.SH This finally brings us to the definition of the scheme FV.SH. Using the notation introduced for the scheme LPR.ES, we have:

- FV.SH.SecretKeyGen(1^λ): sample $\mathbf{s} \leftarrow R_2$ and output $\mathtt{sk} = \mathbf{s}$
- FV.SH.PublicKeyGen(sk) = LPR.ES.PublicKeyGen(sk)
- FV.SH.EvaluateKeyGen:
  - Version 1: parameters $(\mathtt{sk}, T)$: for $i = 0, \ldots, \ell = \lfloor \log_T(q) \rfloor$, sample $\mathbf{a}_i \leftarrow R_q, \mathbf{e}_i \leftarrow \chi$ and return

$$\mathtt{rlk} = \left[\left(\left[-(\mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i) + T^i \cdot \mathbf{s}^2\right]_q, \mathbf{a}_i\right) : i \in [0..\ell]\right] .$$

  - Version 2: parameters $(\mathtt{sk}, p)$: sample $\mathbf{a} \leftarrow R_{p \cdot q}, \mathbf{e} \leftarrow \chi'$ and return

$$\mathtt{rlk} = \left([-(\mathbf{a} \cdot \mathbf{s} + \mathbf{e}) + p \cdot \mathbf{s}^2]_{p \cdot q}, \mathbf{a}\right) .$$

- FV.SH.Encrypt(pk, $\mathbf{m}$): to encrypt a message $\mathbf{m} \in R_t$, let $\mathbf{p}_0 = \mathtt{pk}[0]$, $\mathbf{p}_1 = \mathtt{pk}[1]$, sample $\mathbf{u} \leftarrow R_2$, $\mathbf{e}_1, \mathbf{e}_2 \leftarrow \chi$ and return

$$\mathtt{ct} = \left([\mathbf{p}_0 \cdot \mathbf{u} + \mathbf{e}_1 + \Delta \cdot \mathbf{m}]_q, [\mathbf{p}_1 \cdot \mathbf{u} + \mathbf{e}_2]_q\right)$$

- FV.SH.Decrypt(sk, ct) = LPR.ES.Decrypt
- FV.SH.Add($\mathtt{ct}_1, \mathtt{ct}_2$): return $([\mathtt{ct}_1[0] + \mathtt{ct}_2[0]]_q, [\mathtt{ct}_1[1] + \mathtt{ct}_2[1]]_q)$
- FV.SH.Mul($\mathtt{ct}_1, \mathtt{ct}_2, \mathtt{rlk}$): compute

$$\mathbf{c}_0 = \left[\left\lfloor \frac{t \cdot (\mathtt{ct}_1[0] \cdot \mathtt{ct}_2[0])}{q} \right\rceil\right]_q$$
$$\mathbf{c}_1 = \left[\left\lfloor \frac{t \cdot (\mathtt{ct}_1[0] \cdot \mathtt{ct}_2[1] + \mathtt{ct}_1[1] \cdot \mathtt{ct}_2[0])}{q} \right\rceil\right]_q$$
$$\mathbf{c}_2 = \left[\left\lfloor \frac{t \cdot (\mathtt{ct}_1[1] \cdot \mathtt{ct}_2[1])}{q} \right\rceil\right]_q$$

  - FV.SH.Relin Version 1: write $\mathbf{c}_2$ in base T, i.e. write $\mathbf{c}_2 = \sum_{i=0}^{\ell} \mathbf{c}_2^{(i)} T^i$ with $\mathbf{c}_2^{(i)} \in R_T$ and set

$$\mathbf{c}'_0 = \left[\mathbf{c}_0 + \sum_{i=0}^{\ell} \mathtt{rlk}[i][0] \cdot \mathbf{c}_2^{(i)}\right]_q \quad \text{and} \quad \mathbf{c}'_1 = \left[\mathbf{c}_1 + \sum_{i=0}^{\ell} \mathtt{rlk}[i][1] \cdot \mathbf{c}_2^{(i)}\right]_q .$$

    Return $(\mathbf{c}'_0, \mathbf{c}'_1)$.
  - FV.SH.Relin Version 2: compute

$$(\mathbf{c}_{2,0}, \mathbf{c}_{2,1}) = \left(\left[\left\lfloor \frac{\mathbf{c}_2 \cdot \mathtt{rlk}[0]}{p} \right\rceil\right]_q, \left[\left\lfloor \frac{\mathbf{c}_2 \cdot \mathtt{rlk}[1]}{p} \right\rceil\right]_q\right) ,$$

    and return $([\mathbf{c}_0 + \mathbf{c}_{2,0}]_q, [\mathbf{c}_1 + \mathbf{c}_{2,1}]_q)$.

Combining Lemma 2 with the analysis of the relinearisation step, we have proved the following lemma.

Lemma 3. *Let* $\mathtt{ct}_i$ *for* $i = 1, 2$ *be two ciphertexts, with* $[\mathtt{ct}_i(\mathbf{s})]_q = \Delta \cdot \mathbf{m}_i + \mathbf{v}_i$*, and* $||\mathbf{v}_i|| < E < \Delta/2$*. Set* $\mathtt{ct}_{\text{add}} = \mathtt{FV.SH.Add}(\mathtt{ct}_1, \mathtt{ct}_2)$ *and* $\mathtt{ct}_{\text{mul}} = \mathtt{FV.SH.Mul}(\mathtt{ct}_1, \mathtt{ct}_2, \mathtt{rlk})$ *then*

$$[\mathtt{ct}_{add}(\mathbf{s})]_q = \Delta \cdot [\mathbf{m}_1 + \mathbf{m}_2]_t + \mathbf{v}_{\text{add}},$$
$$[\mathtt{ct}_{mul}(\mathbf{s})]_q = \Delta \cdot [\mathbf{m}_1 \cdot \mathbf{m}_2]_t + \mathbf{v}_{\text{mul}},$$

with $||\mathbf{v}_{\text{add}}|| < 2 \cdot E + t$ *and* $||\mathbf{v}_{\text{mul}}|| < E \cdot t \cdot \delta_R \cdot (\delta_R + 1.25) + E_{\mathtt{Relin}}$ *where* `Version 1`*:* $E_{\mathtt{Relin}} = (l+1) \cdot B \cdot T \cdot \delta_R/2$ *and* `Version 2`*:* $E_{\mathtt{Relin}} = (q \cdot B_k \cdot \delta_R)/p + (\delta_R \cdot ||s|| + 1)/2$*, with* $p = q^{k-1}$ *and* $B_k > \alpha^{1-\sqrt{k}} \cdot q^{k-\sqrt{k}} \cdot B^{\sqrt{k}}$*.*

To see what the maximum depth (note that we are really talking about depth, and not about the degree of the corresponding Boolean function) is that we can evaluate, recall that the noise of a fresh ciphertext assuming $\mathbf{s}, \mathbf{u} \in R_2$ is roughly given by $2 \cdot \delta_R \cdot B$. Since we choose T or p such that the noise introduced by relinearisation is smaller than the noise from a multiplication, we will ignore the second term $E_{\mathtt{Relin}}$. Note that this implicitly introduces an assumption on the maximum size of B. The noise after L levels of multiplications is roughly of size $\simeq 2 \cdot B \cdot \delta_R^{2L+1} \cdot t^L$. Since we can only decrypt when this noise is smaller than $\Delta/2$, this bring us to the following theorem.

Theorem 1. *Using the notation of the scheme* `FV.SH` *and assuming that* $||\chi|| < B$*,* `FV.SH` *can correctly evaluate circuits of multiplicative depth* L *with*

$$4 \cdot \delta_R^L \cdot (\delta_R + 1.25)^{L+1} \cdot t^{L-1} < \lfloor q/B \rfloor \ .$$

A very important remark here is to note that B does not appear on the left hand side, since the noise introduced by multiplication does not involve B. This shows that even for large values of B we are still able to perform a reasonable number of multiplications, which is different from the existing RLWE-based schemes.

5 Fully Homomorphic Encryption

To turn the somewhat homomorphic encryption scheme `FV.SH` into a fully homomorphic encryption scheme, we require a method to lower the noise before we hit the maximum noise level. Gentry's idea of bootstrapping [7] is really simple, namely run the decryption of `FV.SH` in the encrypted domain, i.e. homomorphically. The result of this operation will be an encryption of the same message, but with noise of a fixed size. If the decryption circuit can be evaluated in depth D, then the noise after bootstrapping equals the noise one would obtain by evaluating a depth D circuit, which is clearly independent of the starting noise present in the ciphertext. Therefore, if `FV.SH` can handle circuits of depth $D + 1$, we can still handle one multiplication after bootstrapping and thus obtain a fully homomorphic scheme. In practice it might be better to "over-design" the homomorphic capability of the scheme, i.e. choose parameters such that the maximum depth L is strictly greater than $D + 1$.

Since we need to run `FV.SH.Decrypt` homomorphically, we require the decryption circuit to be as simple as possible. In the early days of FHE schemes [8, 21, 18, 4, 5] this was typically handled by squashing the decryption circuit by writing the secret key $\mathbf{s}$ as the solution to a sparse subset sum problem, which introduced a new security assumption. In our case however, we can simply handle the real `FV.SH.Decrypt` without the need for squashing.

Recall that given a ciphertext $\mathtt{ct}$, we have $\mathtt{ct}[0]+\mathtt{ct}[1]\cdot\mathbf{s} = \Delta\cdot\mathbf{m}+\mathbf{v}+q\cdot\mathbf{r}$, for some error term $\mathbf{v}$ with $||\mathbf{v}|| < \Delta/2$. In a first step we will compute $\mathtt{ct}[0] + \mathtt{ct}[1] \cdot \mathbf{s} \bmod q$, which can then be transformed easily in the centered reduction $[\mathtt{ct}[0] + \mathtt{ct}[1] \cdot \mathbf{s}]_q$. The crucial observation is that if we do not allow the norm of $\mathbf{v}$ to grow to its maximum size, but only up to $||\mathbf{v}|| < \Delta/\mu$ for some $\mu > 2$, we can optimize the decryption by ignoring most of $\mathtt{ct}[0]$ and $\mathtt{ct}[1]$. Indeed, if we replace $\mathtt{ct}[0]$ and $\mathtt{ct}[1]$ by $\mathbf{c}_0 = \mathtt{ct}[0] + \mathbf{e}_0$ and $\mathbf{c}_1 = \mathtt{ct}[1] + \mathbf{e}_1$ with $||\mathbf{e}_i|| < \Delta/\nu$ (for instance by setting all lower order bits to zero), then we have

$$\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} = \Delta \cdot \mathbf{m} + \mathbf{v} + \mathbf{e}_0 + \mathbf{e}_1 \cdot \mathbf{s} + q \cdot \mathbf{r}\,.$$

The noise term in this truncated ciphertext therefore has gone up from $||\mathbf{v}||$ to $||\mathbf{v} + \mathbf{e}_0 + \mathbf{e}_1 \cdot \mathbf{s}||$. To bound the size of the new noise we require two functions: define $\mathrm{abs}(a(x))$ for $a(x) \in \mathbb{Z}[x]$ as the polynomial obtained by taking the absolute value of its coefficients, and define the function $H(f)$ by

$$H(f) = \max\left\{ || \sum_{i=0}^{d-1} \mathrm{abs}(x^{i+j} \bmod f(x))|| \;\mid\; j = 0, \ldots, d-1 \right\}. \qquad (5)$$

For polynomials of the form $f = x^d + 1$, we have that $H(f) = 1$. If we now let h denote the Hamming weight of $\mathbf{s}$, then we conclude that the new noise is bounded by $\Delta/\mu + (H(f) \cdot h + 1) \cdot \Delta/\nu$. As such the truncated ciphertext will still decrypt as long as $2 \cdot \mu \cdot (H(f) \cdot h + 1) < \nu \cdot (\mu - 2)$. By taking $\mu = 2^v + 4$ we can take $\nu = (2 + 2^{2-v}) \cdot (H(f) \cdot h + 1)$. This shows that we can simply work with the top part of $\mathtt{ct}[0] \bmod q$ and $\mathtt{ct}[1] \bmod q$ of size $S_R = \mathrm{size}(\lceil \nu \cdot t \rceil)$ bits. Note that since we first compute the result modulo q, all these coefficients will be positive and we do not have to mess about with a sign bit.

Each coefficient of the result $\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s}$ can thus be computed (recall that $\mathbf{s}$ is binary) as a sum of $d + 1$ integers each of which has S_R bits. We can represent this situation by a $(d + 1) \times S_R$ matrix M of the bits of each of these integers (least significant bit in first column, little endian notation). Note that roughly half of the bits in the matrix will be zero, and by moving all zeros down, we in fact will be working with a matrix of roughly half the size. To simplify the exposition, we present the analysis for two cases: firstly, the optimised case with $q = 2^n$ and $t \mid q$, which is very easy to analyse and provides optimal results, and secondly, the general case. Unlike previous papers where the general case was handled directly, we use a modulus switching trick to (almost) reduce it to the optimal case.

5.1 Optimised case: $q = 2^n$ and $t \mid q$

The reason why this case is easier to handle is due to the fact that the decryption function simplifies considerably: indeed since $q = 2^n$ and $t \mid q$, we can write $\Delta = 2^k$. It is easy to see that

$$\left[\left\lfloor \frac{t}{q} \cdot [\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s}]_q \right\rceil\right]_t = \left[\left\lfloor \frac{\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s}}{\Delta} \right\rceil\right]_t ,$$

by using the expression $\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} = \Delta \cdot \mathbf{m} + \mathbf{e} + q \cdot \mathbf{r}$. As such we do not require centered reduction, and the division by Δ boils down to a simple shift. Furthermore, since t also is a power of 2, the final centered reduction is a simple function of the bits obtained modulo t. To be precise we summarize how the decryption of one coefficient of the message (say the constant coefficient) will be performed:

– Given ciphertext `ct` consider the top S_R bits of `ct` modulo q, i.e. set $\mathbf{d}_0 = (\mathtt{ct}[0] \bmod q) \gg (n - S_R)$ and $\mathbf{d}_1 = (\mathtt{ct}[1] \bmod q) \gg (n - S_R)$, where $\gg$ denotes right shift
– Consider the $d + 1$ integers modulo 2^{S_R} obtained from the constant coefficients of $\mathbf{d}_0(x)$ and $\mathbf{s}_j \cdot x^j \cdot \mathbf{d}_1(x) \bmod f(x)$ for $j = 0, \ldots, d - 1$ and put the bits of each of these constants in the $(d + 1) \times S_R$ matrix M
– Add $d + 1$ integers together modulo 2^{S_R} resulting in an integer $0 \leq w < 2^{S_R}$
– Define the rounding bit $w_b = w[k - n + S_R - 1]$ and finally output $m_0 = [w \gg (k - n + S_R) + w_b]_t$

The main computation in decryption therefore is simply computing a sum of $(d + 1)$ integers modulo 2^{S_R} by computing the sum of the rows of the matrix M. For this we use a standard two stage process: we first repeatedly use a carry-save adder that takes as input 3 rows of M and reduces them to two rows with the same sum. More in detail: denote by a_i, b_i, c_i for $i = 1, \ldots, S_R$ the bits in the three rows A, B, C, then the carry save adder returns two rows containing in the i-th position

$$a_i \oplus b_i \oplus c_i \quad \text{and} \quad \overline{\overline{a}_{i-1}\overline{b}_{i-1} \oplus \overline{b}_{i-1}\overline{c}_{i-1} \oplus \overline{a}_{i-1}\overline{c}_{i-1}}$$

where $a_{-1}, b_{-1}, c_{-1}, a_{S_R+1}, b_{S_R+1}, c_{S_R+1}$ are zero by definition. Note that since we are computing the sum only modulo 2^{S_R} we can ignore any carry propagation beyond the S_R-th bit. The crucial point to remark here is that when we perform this computation in the encrypted domain, the noise of the first row has not increased much, since it does not involve any multiplications. In the next iteration, we should take care to maximally combine rows with similar noise level. The reason for this is that the noise of a product of two ciphertexts is a constant factor larger than the *maximum* of the noises. As such, we need to try to multiply ciphertexts in a balanced way by combining them according to their noise sizes. We can repeat the carry-save adder until we are left with two rows. In the second stage, we use a simple schoolbook addition to finally recover the full sum represented by S_R bits. It is easy to see that the degree of the k-th bit (counting from 0) as a Boolean expression in the bits $b_{i,j}$ is given by 2^k, so we need a depth $S_R - 1$ circuit to add these numbers together modulo 2^{S_R}. The rounding bit then simply is the $k - n + S_R - 1$-th bit (starting to count at 0 for LSB) and the result modulo t is obtained by adding this bit to the $n - k$ last bits. Note that this step does not increase the required depth since we are working modulo t. Furthermore, note that knowing the result modulo t is equivalent with knowing the centered reduction result, so we can skip the last step. By this we mean that if we would encrypt $m + t$ where $m \in \mathbb{Z}_t$, then decryption would still result in m.

The reason why we have written decryption really as a binary circuit is that this allows us to evaluate the circuit homomorphically, i.e. in the encrypted domain. Note that this requires giving out the encryption of the bits of the secret $\mathbf{s}$, so we introduce the following procedure:

– $\texttt{FV.FH.BootKeyGen}(\mathbf{s}, \texttt{pk})$: return

$$\texttt{bsk} = [\texttt{FV.FH.Encrypt}(\mathbf{s}_i, \texttt{pk}) : i \in [0..d-1]] \ .$$

Note that in practice one would not encrypt all bits separately, but use SIMD techniques to encrypt several bits simultaneously thereby significantly lowering the memory usage of the bootstrapping key $\texttt{bsk}$.

The analysis of the required multiplicative depth can be summarized in the following theorem.

Theorem 2. *The somewhat homomorphic encryption scheme* $\texttt{FV.SH}$ *for* $q = 2^n$, $t \mid q$ *and using a binary secret* $\mathbf{s}$ *of Hamming weight* h*, can be turned into a fully homomorphic encryption scheme* $\texttt{FV.FH}$ *by using a bootstrapping procedure if* $\texttt{FV.SH}$ *can evaluate circuits of depth* $L \geq \text{size}(\lceil \nu \cdot t \rceil)$ *where* $\nu = \gamma \cdot (H(f) \cdot h + 1)$ *with* $2 < \gamma < 3$ *and* $H(f)$ *as defined in Equation* (5).

The above theorem thus shows that for $t = 2$, $h = 63$ and $f(x) = x^d + 1$, we only require $L = 9$ if we use $\mu = 2^{10}$. Also note that the required L to obtain $\texttt{FV.FH}$ does not depend on the choice of q nor χ, but only on t, the Hamming weight h of the secret key and the properties of the polynomial f.

5.2 General case

Although the general case can be handled directly, i.e. by analysing $\texttt{FV.SH.Decrypt}$ as is, we will use a little trick that simplifies the analysis immensely. Recall that a valid ciphertext $\texttt{ct}$ satisfies $\texttt{ct}[0] + \texttt{ct}[1] \cdot \mathbf{s} = \Delta \cdot \mathbf{m} + \mathbf{v} + q \cdot \mathbf{r}$, with $||\mathbf{v}|| < \Delta/2$ and $||\mathbf{r}|| < \delta_R \cdot ||\mathbf{s}||$. If we assume that the noise $\mathbf{v}$ is not maximal size, we can simply switch from modulus q to 2^n where $n = \lfloor \log_2(q) \rfloor$ by a simple scaling over $2^n/q$. Indeed, if we define $\mathbf{c}_i = \lfloor 2^n \cdot \texttt{ct}[i]/q \rceil$ for $i = 0, 1$, then one can easily verify that

$$\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s} = \left\lfloor \frac{2^n}{t} \right\rfloor \cdot m + \mathbf{e} + 2^n \cdot \mathbf{r} \, ,$$

with $||\mathbf{e}|| < ||\mathbf{v}|| + t + (1 + \delta_R \cdot ||\mathbf{s}||)$. As long as $||\mathbf{e}|| < \lfloor 2^n/t \rfloor /2$, we obtain a valid ciphertext with respect to the modulus 2^n instead of q.

The decryption now almost becomes as simple as in the optimised case, by considering $(\mathbf{c}_0, \mathbf{c}_1)$ as the ciphertext to decrypt. The first step is exactly the same as before: we work with the top bits only via $\mathbf{d}_0$ and $\mathbf{d}_1$ and thus obtain an approximation to

$(\mathbf{c}_0 + \mathbf{c}_1 \cdot \mathbf{s}) \bmod 2^n$ by $2^\ell \cdot (\mathbf{d}_0 + \mathbf{d}_1 \cdot \mathbf{s}) \bmod 2^n$ with $\ell = n - S_R$ using a circuit of depth $S_R - 1$. Let w be the constant coefficient of $(\mathbf{d}_0 + \mathbf{d}_1 \cdot \mathbf{s}) \bmod 2^{S_R}$, then the constant term of the message is given by

$$\left[\left\lfloor \frac{t \cdot [2^\ell \cdot w]_{2^n}}{2^n} \right\rceil\right]_t .$$

Since t no longer necessary divides 2^n however, we really need to work with the centered reduction instead of just modulo 2^n, but this is rather easy. If $w > 2^{S_R - 1}$, we need to replace it by $w - 2^{S_R}$ so if we define $w_c = w[S_R - 1] \cdot w[S_R - 2] \bmod 2$, then the centered reduction is given by the combination of

$$w_r := \big((1 \oplus w_c) \cdot w + w_c \cdot (2^{S_R} - w)\big) \bmod 2^{S_R}$$

and a sign bit equal to w_c (a 1 indicates negative number). Note that $(2^{S_R} - w)$ is very easy to compute by negating all bits of w and adding 1. Since these operations add two levels, the centered reduction w_r can be obtained with a circuit of depth $S_R + 1$. To compute $t \cdot w_r$ we require an extra $HW(t) - 1$ levels, where $HW(t)$ denotes the Hamming weight of t. Dividing by 2^n is for free and the rounding is taken care of as in the optimal case, i.e. by simply adding the first bit after the binary point, which requires at most one level of depth extra. As before, we can again ignore the final centered reduction modulo t. This finally proves the more general theorem.

Theorem 3. *The somewhat homomorphic encryption scheme* `FV.SH` *with binary secret* $\mathbf{s}$ *of Hamming weight h, can be turned into a fully homomorphic encryption scheme* `FV.FH` *by using a bootstrapping procedure if* `FV.SH` *can evaluate circuits of depth* $L \geq \text{size}(\lceil \nu \cdot t \rceil) + HW(t) + 2$ *where $HW(t)$ denotes Hamming weight of t and* $\nu = \gamma \cdot (H(f) \cdot h + 1)$ *with* $2 < \gamma < 3$ *and $H(f)$ as defined in Equation* (5).

6 Choosing Parameters

In this section we explain how to choose parameters that guarantee a given level of security, and allow a depth L circuit to be evaluated. Combining this with the minimal L to obtain FHE, we thus derive parameters that allow FHE.

6.1 Practical Hardness of RLWE

To assess the hardness of the RLWE-problem, we will follow the analysis of Lindner and Peikert [14] for the standard LWE problem. We therefore implicitly assume that the analysis of Lindner and Peikert also holds for the RLWE-problem.

As before, let q denote the modulus, d the degree of the polynomial ring R and let σ^2 denote the variance of the probability distribution χ. Gamma and Nguyen [6] defined the Hermite factor δ^m of a basis $\mathbf{B}$ of an m-dimensional lattice Λ as $||\mathbf{b}_1|| = \delta^m \cdot \det(\Lambda)^{1/m}$, with $\mathbf{b}_1$ the shortest vector in $\mathbf{B}$. Lindner and Peikert [14] call δ the root-Hermite factor. Furthermore, Gamma and Nguyen showed that the run time of reduction required to achieve a given δ (in large enough dimension) mainly depends on δ, and not on the dimension or determinant. Lindner and Peikert [14] show that the time in seconds to compute a basis with root-Hermite factor δ is roughly given by

$$\log_2(time) = 1.8/\log_2(\delta) - 110\,.$$

If we assume a security level of λ bits, i.e. we set $time = 2^\lambda$, then the minimal δ we can achieve according to the above estimate is $\log_2(\delta) = 1.8/(\lambda + 110)$. For instance, when we set $\lambda = 128$, we obtain that $\delta \simeq 1.0052$.

In order for the distinguishing attack described in [14] to succeed with advantage ε we need to find vectors of length $\alpha \cdot (q/\sigma)$, with $\alpha = \sqrt{\ln(1/\varepsilon)/\pi}$, so for an advantage of $\varepsilon = 2^{-64}$, we have $\alpha \simeq 3.758$. For a fixed root-Hermite factor δ, Lindner and Peikert [14] show that using the optimal attack strategy, the length of the shortest vector one can compute is given by $2^{2\sqrt{d \log_2(q) \log_2(\delta)}}$. This leads to the condition

$$\alpha \cdot \frac{q}{\sigma} < 2^{2\sqrt{d \log_2(q) \log_2(\delta)}}\,. \tag{6}$$

Note that the above equation shows that for fixed q and growing σ, we can take a lower degree d, providing more flexibility than previous works where σ was always chosen to be very small. Furthermore, this equation also shows that for fixed d and a fixed security level, we can transform one valid parameter pair (q, σ) into another valid pair (q^k, σ_k) for any real $k > 1$ as long as we choose $\sigma_k > \alpha^{1-\sqrt{k}} \cdot q^{k-\sqrt{k}} \cdot \sigma^{\sqrt{k}}$. Note that it is precisely this bound that was used in `FV.SH.Relin`, `Version 2`.

6.2 Parameters for FHE

To generate parameters for an FHE scheme at a given security level λ, we first compute $\log_2(\delta) = 1.8/(\lambda + 110)$. Equation 6 then allows us to either choose the degree d first and compute a valid (q, σ)-pair; or to choose (q, σ) first and derive d from this. Recall that the distribution χ can be considered B-bounded when we set $B = \beta(\epsilon) \cdot \sigma$ for some tiny ϵ. According to Theorem 1, the maximum multiplicative depth $L_{\max}$ that `FV.SH` can handle satisfies

$$4 \cdot \beta(\epsilon) \cdot \delta_R^{L_{\max}} \cdot (\delta_R + 1.25)^{L_{\max}+1} \cdot t^{L_{\max}-1} < \frac{q}{\sigma}\,. \tag{7}$$

Recall that the above inequality assumes that the noise introduced by relinearisation is smaller than the noise after the first multiplication. For both versions of relinearisation, this noise depends on B, so this implicit assumption bounds the maximum allowable B.

From Theorem 3 we can easily derive the minimum $L_{\min}$ (given t and a Hamming weight h) to obtain FHE, and this $L_{\min}$ is independent of (q, σ). In fact, if we use a parametrised family for $f(x)$, e.g. $f(x) = x^d + 1$, for which $H(f)$ is constant, then $L_{\min}$ is even independent of d. Of course we need to choose h large enough such that χ has sufficient entropy, but this is a minor restriction. Substituting this $L_{\min}$ in Equation (7), multiplying by α and combining with Equation (6) then finally gives:

$$4 \cdot \alpha \cdot \beta(\epsilon) \cdot \delta_R^{L_{\min}} \cdot (\delta_R + 1.25)^{L_{\min}+1} \cdot t^{L_{\min}-1} < 2^{2\sqrt{d \log_2(q) \log_2(\delta)}}\,.$$

Note that the left hand side does not depend on q and only δ_R depends on f and thus d. The above formula thus allows us to either choose d first and then compute a valid (q, σ)-pair or vice-versa. To provide a simple example, consider the family $f_d(x) = x^d + 1$, then $\delta_R = d$, $H(f_d) = 1$ and for $t = 2$, $h = 63$ we have $L_{\min} = 9$. For $\epsilon = 2^{-64}$ we have $\beta(\epsilon) \simeq 9.2$ and $\alpha \simeq 3.8$ and for 128-bit security level we have $\log_2(\delta) = 0.0076$. If we choose $q = 2^n$ and $d = 2^k$, then substituting all these values then finally leads to

$$15.13 + 19 \cdot k < 0.174 \cdot \sqrt{n} \cdot 2^{k/2} .$$

So if we choose $k = 10$, then we require $n > 1358$ to guarantee FHE capabilities.

7 Conclusions and Future Work

In this paper we ported Brakerski's FHE scheme from the LWE to the RLWE setting and provided a detailed analysis of all subroutines involved such as multiplication, relinearisation and bootstrapping. This analysis results in tight worst case bounds on the noise caused by homomorphic operations, which can be used to derive very concrete parameters for which the scheme can be made fully homomorphic. We introduced two optimised versions of relinearisation that result both in a smaller relinearisation key and faster computations than existing solutions. Furthermore, we simplified the analysis of the bootstrapping step by a modulus switching trick.

The results in this paper provide a sound theoretical basis for practical implementations. We note however that the derived bounds are worst case bounds and not average case bounds that can be easily derived from the central limit theorem. In a follow up paper we will consider very practical aspects of this scheme, including the average case bounds and we will report on an implementation in the Magma computer algebra system and on a highly optimised dedicated polynomial multiplier hardware implementation.

Two further major optimisations are possible and are work in progress, namely the bootstrapping step can be highly improved by using a much better SIMD approach than the current state-of-the-art; and the actual homomorphic system itself should be replaced by our "t-adic approach to FHE", which keeps the size of the ciphertext minimal w.r.t. the size of the noise it contains.

Acknowledgments

The authors would like to thank Damien Stehlé, Ron Steinfeld, Oded Regev and Chris Peikert for sharing their deep insights about lattice and (R)LWE related matters (amongst many other topics).

References

1. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.

2. Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. *IACR Cryptology ePrint Archive*, 2012:78, 2012.
3. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science 2012*, pages 309–325. ACM, 2012.
4. Zvika Brakerski and Vinod Vaikuntanathan. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In *Advances in Cryptology - CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.
5. Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In *Advances in Cryptology - CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 487–504. Springer, 2011.
6. Nicolas Gama and Phong Q. Nguyen. Predicting Lattice Reduction. In *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, 2008.
7. Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.
8. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC 2009*, pages 169–178. ACM, 2009.
9. Craig Gentry and Shai Halevi. Fully Homomorphic Encryption without Squashing Using Depth-3 Arithmetic Circuits. In *FOCS 2011*, pages 107–109. IEEE, 2011.
10. Craig Gentry and Shai Halevi. Implementing Gentry's Fully-Homomorphic Encryption Scheme. In *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.
11. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic Evaluation of the AES Circuit. *IACR Cryptology ePrint Archive*, 2012:99, 2012.
12. Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the Learning with Errors Assumption. In *ICS 2010*, pages 230–240. Tsinghua University Press, 2010.
13. Adeline Langlois and Damien Stehlé. Hardness of decision (R)LWE for any modulus. *IACR Cryptology ePrint Archive*, 2012:91, 2012.
14. Richard Lindner and Chris Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. In *Topics in Cryptology - CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011.
15. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010. Full version of the paper available upon request from authors.
16. Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *CCSW 2011*, pages 113–124. ACM, 2011.
17. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC 2005*, pages 84–93. ACM, 2005.
18. Nigel P. Smart and Frederik Vercauteren. Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In *PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.
19. Nigel P. Smart and Frederik Vercauteren. Fully Homomorphic SIMD Operations. *IACR Cryptology ePrint Archive*, 2011:133, 2011.
20. Damien Stehlé and Ron Steinfeld. Faster Fully Homomorphic Encryption. In *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 377–394. Springer, 2010.

21. Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.