NTNU
Norwegian University of
Science and Technology

The Handling of Noise and Security of Two Fully Homomorphic Encryption Schemes

Martha Norberg Hovd

Master of Science in Mathematical Sciences
Submission date: June 2017
Supervisor: Kristian Gjøsteen, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

Noise is introduced as a means to ensure security of encryption schemes in general, and fully homomorphic encryption schemes in particular. Strategies to control the growth of this and thereby guarantee correct decryption are presented. These strategies are then applied in two similar fully homomorphic encryption schemes, and the requirements of sufficient and successful noise reductions are discussed in both. It is shown that this forces the parameters of either scheme to satisfy certain bounds, which compromises the security of one scheme, whilst it is of no great consequence with regards to security of the other.

Sammendrag

Støy introduseres som et hjelpemiddel for å sørge for sikkerhet i krypteringssystem generelt, og særlig i fullstendig homomorfe krypteringssystem. Strategier for å kontrollere veksten av støyen og dermed garantere for korrekt dekryptering presenteres. Disse strategiene anvendes dernest i to lignende fullstendig homomorfe krypteringssystem, og kravene for tilstrekkelig og vellykket støyreduksjon diskuteres i begge systemene. Det vises at dette krever at parametrene i systemene tilfredstiller visse skranker, som svekker sikkerheten i det ene systemet, mens det ikke er av nevneverdig konsekvens med hensyn til sikkerheten i det andre.

Acknowledgements

First and foremost, I would like to thank my supervisor Kristian Gjøsteen for his valuable help, in the form of shared insights, feed-backs, comments, interesting sidetracks, patience, friendly advice and enthusiasm in the entire process of writing this thesis.

I would also like to acknowledge the help of Kurt Rohloff, as he pointed out the weakness of his and Cousins' scheme to me, and also provided me with references on articles describing attacks exploiting this weakness.

My colleagues at the bookshop also deserve their share of gratitude, as they have been of great help throughout the years, by letting me work when I could, respecting me when I could not, cheering me on no matter what and making a 90 minute commute more than worth it every single time.

Finally, I would like to thank my friends and family for all their love and support in general, but in particular throughout this past year. My last shred of gratitude, though, is reserved for my parents, for teaching me that the most important thing was whether or not I enjoyed what I was doing; and for teaching me that all mathematics is based on logic and sense, even if neither was immediately obvious to me.

Contents

1 Introduction

We all have secrets, and although they may range from trivial to vastly complicated, in one sense they are all perfectly simple: if you do not want someone to know it, do not tell anyone. Reality is rarely this simple though, as certain secrets must be shared, but importantly not with everyone. Is it possible to make such a sharing of secrets safe from eavesdroppers? Stated somewhat differently: is it possible to ensure that only the people you trust with a secret has access to it?

This is of course not a new problem, and fortunately there is a solution: encryption, scrambling the message in such a way that only those who know how to read it may do so. For anyone else it will merely seem like gibberish. Metaphorically, this means putting the message into a solid box that only the receiver knows how to unlock and an adversary will have a hard time breaking open. Although an untrusted messenger knows that something is being sent, the shape of the box should tell him absolutely nothing about the content of it.

However, what if what you are sending is no longer a message to be read by someone you trust, but something to be processed by an *untrusted* recipient? Could you provide a way for someone to process it without gaining access to it? Continuing the metaphor of locked boxes containing secrets, the solution of this problem is creating a box only you know how to open with gloves leading into it, so one may work on the contents, but not remove it from the box. If the locked box only some know how to open corresponds to encryption, this box allowing processing of, but denying access to, the box's content corresponds to fully homomorphic encryption (FHE).

1.1 Fully Homomorphic Encryption

The goal of fully homomorphic encryption is to allow computation to be performed on encrypted data without decrypting it. This will make it possible to perform useful tasks on confidential data being stored in an untrusted environment without breaching the confidentiality. Suppose we have two ciphertexts c_1, c_2 encrypting m_1, m_2 respectively. Any encryption scheme such that $\text{Dec}(c_1 + c_2) = m_1 + m_2$ is said to be homomorphic with respect to addition, and is similarly referred to as homomorphic wrt. multiplication if $\text{Dec}(c_1 c_2) = m_1 m_2$. If the encryption scheme exhibits both these properties, it is fully homomorphic. Informally this means that evaluating any ciphertext in a function f and then decrypting it provides the same result as first decrypting a ciphertext and then evaluating it in f. For a given ciphertext c we must in other words have $\text{Dec}(f(c))=f(\text{Dec}(c))$.

However, seeing as we are evaluating bit-strings, i.e. $m \in \{0,1\}^*$, any function may be expressed as a Boolean circuit using only AND and XOR-gates. Using this circuit, we may evaluate the string(s) bit-wise, where a XOR-gate corresponds to addition and an AND-gate corresponds to multiplication. Hence, instead of saying a cipher- or plaintext is evaluated wrt. a function f, we may refer to it as being evaluated in a Boolean circuit. This means that demanding that any function may be applied to an encrypted plaintext and its decryption still being correct is equivalent to it being possible to pass the ciphertext through any Boolean circuit of arbitrary depth and it still giving the correct decryption afterwards.

The possibilities of fully homomorphic encryption are many, ranging from tailored personal advertising without the loss of privacy to the analysis of medical data without breaching of confidentiality, the latter has in fact already been performed [3, 5]. However, there are certain problems regarding the current schemes created, the main one being inefficiency. Because although all available schemes run in polynomial time, this does not necessarily make them particularly fast or practical. One of the reasons these schemes are rather slow is due to the difficulty of handling noise, which is used to conceal the encrypted messages from adversaries. Informally, the more a ciphertext is processed, the more noise is added to it. The risk is that if too much noise is added the ciphertext will no longer be correctly decrypted, making it worthless. To reduce this risk the scheme must continuously reduce the level of noise during the evaluation of a circuit, without compromising the security or correctness of the scheme, wherein the challenge lies.

1.2 Structure of Paper

This paper will study how two fully homomorphic encryption schemes handle the generated noise, as well as the security of these schemes. First noise and what it is, the necessity of it and the problems related to it will be discussed. Then two different and widely used techniques for reducing noise in ciphertexts, modulus switching and bootstrapping, are discussed. To make these discussions more relatable, a fairly simple example of a fully homomorphic encryption scheme is used throughout. As it merely serves as an example, properties not pertaining to the processes described will not be discussed.

Following this, background regarding lattices, ideal lattices and problems related to these structures are presented, as the two main encryption schemes of the paper are based on the difficulties of solving these problems.

Next, the two main schemes are presented. The first follows the blueprint of NTRU, but is based on slightly different assumptions to allow for homomorphic operations, which will be shown to render the scheme insecure. Seeing as this makes the scheme practically unusable, the efficiency of this scheme will not be discussed. The final encryption scheme is based on a fairly new mathematical problem known as RLWE, and is considered one of the more promising candidates for a practical implementation of FHE. The schemes are based on articles [21] and [6] respectively, and have been chosen as they are two of rather few fully homomorphic schemes that have actually been implemented [13, 3]. In addition, there are both key similarities and differences between them, as will be discussed in the penultimate section on comparison before the thesis is concluded.

1.3 Notation

All vectors are row vectors and will be denoted with bold lower case letters: $\mathbf{v}, \mathbf{w}$, meaning any column vectors are denoted $\mathbf{v}^T, \mathbf{w}^T$, whilst matrices will be denoted with upper case bold letters: $\mathbf{A}, \mathbf{B}$. Elements of either a vector, a matrix or a ring (which in our case will most often be a polynomial ring) will be denoted with a lower case letter in italics: a, b. Vectors will be written as $\mathbf{a} = [a_1, a_2, \ldots, a_n]$, whereas sets will be denoted by $\{0, 1, \ldots\}$.

Multiplication of integers, or an integer and a vector or ring element is denoted by simple juxtaposition: $ab, a\mathbf{v}, af(x)$. Multiplication of a vector and a matrix will be denoted by a single dot: $\mathbf{v}\cdot\mathbf{A}$, $\mathbf{A}\cdot\mathbf{w}^T$ and finally, the multiplication of two polynomials will be denoted by an asterisk: $f * g$. Furthermore, this polynomial multiplication always takes place in some polynomial ring $R = \mathbb{Z}[x]/(x^n + 1)$, and the main motivation of the multiplicative notation is to serve as a reminder of this during computations. It should be clear from the context whether or not a given element is a polynomial, and any polynomial f will therefore, with very few exceptions, not be written $f(x)$. Moreover, n, k, p and q will always denote integers. Let $\mathbf{v}, \mathbf{w}$ be vectors of the same length k over a polynomial ring R. We may then define the dot product of these two vector as $\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{i=1}^{k} v_i * w_i \in R$.

We will operate with the two following modular reductions: $p = r \ \underline{\text{mod}}\ q$ denotes reducing p modulo q to $r \in (-q/2, q/2]$, whilst $p = r \mod q$ denotes the modular reduction to $r \in [0, q-1]$. Note that the only difference in the notation is the underlining of the first mod. In both cases, we may also write $p \equiv r \ \underline{\text{mod}}\ q$ or $p \equiv r \mod q$ if we wish to stress that p is equivalent to r modulo q: $p = r + kq$. This generalizes to vectors and polynomials.

$\|\cdot\|$ denotes the Euclidean norm: $\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$, whilst $\|\cdot\|_\infty$ denotes the infinity norm: $\|\mathbf{v}\|_\infty = \max_i\{|v_i|\}$. Supposing f is a polynomial, $\|f\|, \|f\|_\infty$ refers to calculating either norm of the coefficient vector f.

Any logarithm log will be base 2.

For any set S, $x \leftarrow S$ refers to drawing x from S uniformly at random. For any probability distribution χ, $x \leftarrow \chi$ refers to sampling x according to χ.

2 Noise

Noise in encryption schemes may be viewed as a bounded element of randomness in the scheme added to the message in some way, so that it is easy to remove and unveil the original message for trusted parties holding some extra information, and concealing the message from adversaries. However, if the randomness added to the message were to exceed the given bound, decryption might not be correct. Noise may be viewed as a somewhat brittle form of randomness: if too much is used to cover a message, it might collapse and make it impossible to recover said message.

If randomness or noise is applied in the encryption it is not deterministic, but rather probabilistic when viewed as an algorithm, meaning a message m may have several different encryptions. The necessity and advantages of this property is best understood when it is lacking, and such a scheme, namely the simplest form of RSA, will be presented. Next an encryption scheme using noise in its encryption will be presented, and also the problems this may cause if the scheme is to be fully homomorphic.

2.1 The Necessity of Noise

For the construction of an RSA encryption scheme one needs two primes, p and q, as well as their product $n = pq$. Based on this, one computes the Euler totient function $\phi(n) = (p-1)(q-1)$. Finally, we generate e and d by ensuring $ed = 1 + k\phi(n)$, i.e. $ed \equiv 1 \mod \phi(n)$ [23].

The public key of this system is the pair (n, e), whereas the secret key is (n, d). Given these pairs, we may now define encryption and decryption of the scheme. For a message $m \in \{2, \ldots, n-1\}$ we have:

$$\text{Enc}(m) = m^e = c \mod n \qquad \text{Dec}(c) = c^d \mod n,$$

which works because

$$c^d = (m^e)^d = m^{ed} = m^{k\phi(n)+1} = m \mod n.$$

The security of RSA hinges on the hardness of factoring large numbers into their prime factors. This means that, given n, it should be difficult to find its prime factors p and q. If an adversary A is able to find this factorization of n, she may easily calculate $\phi(n)$, making it possible to derive $e^{-1} = d \mod \phi(n)$, as she has access to the public key, and therefore e. If an adversary is able to find d within reasonable time, the scheme is obviously not secure.

However, this is not the only way RSA may have its security compromised; it is also highly susceptible to a chosen plaintext sttack (CPA), meaning that if an adversary is presented with two messages m_0, m_1 and the encryption of one these, c_b, providing the adversary with encryptions of any message she desires gives her a significant advantage in guessing the value of b. For a formal definition of this security notion, see Definition 6.7 in Subsection 6.6. Suppose for example that RSA is used as an encryption scheme over a very small message space, e.g. {yes, no}. Seeing as RSA is a public key encryption scheme, A has access to the public key (n, e) and may thus encrypt {yes, no} herself. This allows her to know with full certainty what any ciphertext decrypts to without needing to actually decrypt it, and certainly without factoring n.

The reason CPA is an effective attack, even when the message space has much greater cardinality than 2, is because the ciphertext is entirely dependent on the plaintext. In order to make the scheme secure against such attacks this cannot be the case: we need the encryption algorithm to output different ciphertexts even if we run it on the same plaintext. This is done by introducing an element of randomness in the encryption. There are several examples of such schemes, one of which being ElGamal [23]. This also has the property of being homomorphic with respect to multiplication, but not addition, as is also the case with RSA.

It is worth mentioning that these schemes typically do not need to impose restrictions on the randomness used to encrypt to ensure correct decryption, so there is no risk of incorrect decryption due to too much randomness being added to the plaintext in these schemes. This is a stark contrast to the schemes presented throughout this thesis, which have to rely on noise to ensure the scheme is homomorphic with respect to both addition and multiplication.

2.2 The Problem with Noise

The following scheme [8] is an example of a scheme relying on noise to achieve a probabilistic encryption algorithm. It should be noted that this property in and of itself does not make the scheme secure, but it is a decent start.

KeyGen(μ): The key q is an odd integer chosen uniformly at random from some interval $q \in [2^{\mu-1}, 2^{\mu})$, so that the binary representation of q has length μ.

Enc(q, m): To encrypt the bit $m \in \{0, 1\}$ set $c = qe + 2r + m$ where the integers r and e are chosen at random in some prescribed intervals, such that $|2r| < |\frac{q}{2}|$.

Dec(q,c): $(c \underline{\text{mod}}\ q) \mod 2$.

The reason we insist that $|2r| < |\frac{q}{2}|$ in the encryption of m is that this ensures $c \in (qe - \frac{q}{2}, qe + \frac{q}{2}]$ and thus that $(c \underline{\text{mod}}\ q) = 2r + m$. However, if the term $2r$ grows large enough, this will no longer be the case, as this will result in $c \underline{\text{mod}}\ q = 2r + m - kq$ for some $k \geq 1$. When this happens, we have no guarantee that the parity of $c \underline{\text{mod}}\ q$ equals the parity of m, and by extension that Dec(q, Enc(q,m))= m. The noise of the ciphertext is in this case said to have become unmanageable.

This is not a problem in and of itself when encrypting, simply restrict the interval from which r is drawn to $(-q/4, q/4)$ and we are ensured to have a correct decryption. The problem arises with the realization that this simple scheme is fully homomorphic if the noise is kept manageable: for two ciphertexts c_1 and c_2, we have

$$\begin{aligned} c_1 + c_2 &= (qe_1 + 2r_1 + m_1) + (qe_2 + 2r_2 + m_2) = qe' + 2r' + (m_1 + m_2), \\ c_1c_2 &= (qe_1 + 2r_1 + m_1)(qe_2 + 2r_2 + m_2) = q\tilde{e} + 2\tilde{r} + m_1m_2, \end{aligned}$$

where the calculation of e', r', $\tilde{e}$ and $\tilde{r}$ are straightforward, but rather uninteresting.

What is interesting however, is the growth of r' and $\tilde{r}$: $r' = r_1 + r_2$, whereas $\tilde{r} = 2r_1r_2 + r_1m_2 + r_2m_1$. Unsurprisingly, the noise resulting from a multiplication is typically much larger than the noise resulting from addition, which also applies to other homomorphic encryption schemes. The problem is that no matter how small a noise we choose to introduce in the encryption of a bit $m \in \{0, 1\}$, the processing of this ciphertext will eventually cause the noise to grow until it is no longer manageable if the circuit is deep enough. Thus, this scheme is only somewhat homomorphic, meaning we may only evaluate Boolean circuits of a certain (in this case rather shallow) depth, but it is impossible to evaluate circuits of arbitrary depth. In order to make this scheme fully homomorphic, we need to find a way to reduce the noise in a ciphertext without sacrificing the correctness of the scheme or its security.

2.3 Modulus Reduction and Switching

One way to allow for noise reduction in the scheme in question is to introduce a ladder of moduli and reduce the noise of a ciphertext by gradually stepping down this ladder. This requires altering the symmetric scheme of the previous subsection, and the resulting scheme is also somewhat homomorphic encryption, but allows for evaluation of deeper circuits, as well as being asymmetric.

KeyGen(μ, γ, ρ, τ): Let the $sk = q$ be an odd μ-bit integer. For $i = 0, 1, \ldots, \tau$, draw $e_i \leftarrow [0, 2^\gamma/q), r_i \leftarrow (-2^\rho, 2^\rho), x_i = qe_i + r_i$. Relabel so that x_0 is the largest, but redraw unless x_0 is odd and $x_0 \ \underline{\text{mod}}\ q$ is even. Finally, for $i = 0, 1, \ldots, \gamma$ generate the following integers: $e'_i \leftarrow [2^{\gamma+i-1}/q, 2^{\gamma+i}/q), r'_i \leftarrow (-2^\rho, 2^\rho), x'_i = 2(qe'_i + r'_i)$. The public key is the set consisting of all the elements with approximate gcd q: $pk = \{x_0, x_1, \ldots, x_\tau, x'_0, x'_1, \ldots, x'_\gamma\}$. Output: sk, pk.

Enc(pk, m): Select a random subset S of $\{x_0, x_1, \ldots, x_\tau\}$, a random integer $r \leftarrow (-2^{2\rho}, 2^{2\rho})$ and output $c = (m + 2r + 2\sum_{x_i \in S} x_i) \ \underline{\text{mod}}\ x_0$.

Dec(sk, c): Output $m' = (c \ \underline{\text{mod}}\ q) \mod 2$.

The parameters of the scheme are all functions of the security parameter λ:

μ : Bit-length of the secret key q, which is the approximate gcd of the integers of the public key.

γ : Bit-length of the integers $\{x_0, x_1, \ldots, x_\tau\}$ in the public key.

ρ : Bit-length of the distance between any integer in the public key and an integer multiple of the secret key q.

τ : The number of integers in the subset $\{x_0, x_1, \ldots, x_\tau\}$ of the public key.

These parameters must be set to meet certain constraints to ensure both the security of the scheme and its homomorphic properties. See [8] for details.

Note that the public key may be viewed as consisting of two subsets, namely $\{x_0, x_1, \ldots, x_\tau\}$ and $\{x'_0, x'_1, \ldots, x'_\gamma\}$, where only the elements of the first set are used during encryption. These elements all have bit-length γ, whereas the elements of the latter subset all have bit-length greater than this, except x'_0. The bit-length of a ciphertext will at most double from any single operation, hence any ciphertext after any operation cannot be larger than $2x'_\gamma$. Therefore, whenever a ciphertext c grows larger than 2^γ, we will set $c_{\gamma+1} = c$ and perform the following operation for $i = \gamma, \gamma - 1, \ldots, 1, 0$:

$$c_i = c_{i+1} \underline{\bmod}\, x'_i$$

and set c_0 to be the new ciphertext c', which will have bit-length no more than γ. This is the gradual modular reduction and every modular operation involves the subtraction of a small multiples of x'_i. Thus, only a small multiple of $2r'_i$ is added to the ciphertext modulo q, meaning the modulus reduction only adds a small amount of noise to the ciphertext.

Similar ideas of a gradual reduction using moduli is used in several fully homomorphic schemes, including the two presented in this paper. The common idea of all three schemes is that the encryption of a message consists of two layers of noise: the outer layer in this scheme is being controlled by x_0, and thus indirectly q, while the inner layer is controlled by the plaintext modulus, namely 2. The key is to ensure that these layers do not interfere with each other: the inner layer must not grow so large that it is affected when removing the outer layer of noise.

This is why we insist $|2r| < q/2$ in the original symmetric scheme. A consequence of this is that it does not matter, with regards to correctness of the scheme, how many multiples of q is added to the ciphertext, as this does not interfere with the inner layer, and therefore does not affect the risk of incorrect decryption. This is why we need not place any restrictions on e in the symmetric encryption scheme presented in Subsection 2.2.

No matter how small the inner layer of noise is, though, it will eventually spill into the outer layer for any circuit of a certain depth. It is therefore common to set the modulus related to the outer layer much larger than the plaintext modulus, to allow for some homomorphic operations to be performed before the noise might grow unmanageable and must therefore be reduced somehow.

However, whilst this integer scheme uses modulus reduction to decrease the outer layer of noise by stepping down a gradually decreasing modular ladder, where the smallest ciphertext modulus is always x_0, this is not the strategy of the two other schemes presented in this paper. In these, modulus switching is used to decrease the inner noise, at the expense of the size of the outer modulus.

That is: as opposed to gradually reducing the outer layer down to a constant smallest modulus, the ciphertext modulus is actually replaced, or switched, with a smaller one. This is done, roughly, by selecting a gradually decreasing ladder of moduli $q_k > \cdots > q_1 > q_0$, and performing a modulus switching on a ciphertext $c \underline{\bmod} q_i$ is simply setting $c' \approx \frac{q_{i-1}}{q_i} c$.

The main difference between the two schemes is that q_k is a composite in the first scheme, so the decreasing ladder of moduli quite simply starts with q_k, and one step down the ladder is a division by a factor of q_k equal to q_i/q_{i-1}. The various moduli of the second scheme share no such relationship. In either case, the procedure roughly divides the inner noise by a factor of q_i/q_{i-1}, but note that the ratio between the noise level and the outer modulus is the same, or might even be smaller than prior to the modulus switching. It might therefore seem slightly counterintuitive how this helps manage the noise. It is important to note that the absolute value of the noise is also important, as well as its ratio to the ciphertext modulus, as this governs how rapidly the noise grows, which is especially important during multiplication. Ensuring that the absolute value of the noise is low allows for a stunted noise growth, allowing more multiplications to be performed before the noise grows unmanageable.

By careful tuning of the modular ladder, one might actually achieve a leveled fully homomorphic encryption scheme: a scheme in which any circuit of specified depth may be evaluated by choosing a long enough ladder. However, such a scheme is unable to evaluate any circuit of unspecified depth without additional ways of reducing the noise. Also, the longer the ladder, the larger the initial modulus has to be, which typically comes at the expense of having to increase the other parameters as well to ensure the scheme is secure. This might have a negative impact on the efficiency of the scheme as a whole. Thus, whilst a leveled fully homomorphic encryption scheme may sound practical, it might actually be less efficient than fully homomorphic schemes. All such schemes so far rely on the rather expensive bootstrapping operation in order to be able to evaluate any given circuit, without needing its depth specified in advance.

3 Bootstrapping

The main idea of bootstrapping is to consider the decryption algorithm of any somewhat or leveled homomorphic encryption scheme as a function with a ciphertext and the secret key as input, which outputs the original message. With this perspective in mind, it is possible to evaluate the decryption circuit homomorphically if one is provided with an encryption of the secret key. This recrypts the ciphertext and will ideally result in an encryption of the same message, only with less noise, to allow further homomorphic evaluation of the ciphertext. Thus, any circuit of any depth may be evaluated: when the noise is about to become unmanageable, simply recrypt the ciphertext to bring the noise down and continue evaluation.

The standard way of showing that a scheme is bootstrappable is to define a set of circuits C the scheme is able to evaluate homomorphically (i.e. $\mathrm{Dec}(sk, C(\mathrm{Enc}(pk, m))) = C(m)$) and ensure that the growth of the output of the evaluated circuit $C(c)$ is polynomial in the security parameter for any ciphertext c. These two requirements are known as correctness and compactness, respectively, and a scheme is bootstrappable if the decryption circuit belongs to this set.

The trouble is that this often requires changes to be made to the decryption circuit, as is the case of our example scheme, where the circuit has to be "squashed": extra information about the secret key is published in the public key to allow for a shallower decryption circuit. It may then be shown that the slightly augmented circuit is both correct and compact, hence allowing bootstrapping to be performed [8].

However, this is not the only possibility as far as bootstrapping is concerned; as noted in [2], to bootstrap a ciphertext may be reinterpreted as preserving the meaningful coefficient of a ciphertext (according to some basis) and mapping the others to zero. This is based on the idea that any ciphertext may be viewed as consisting of one meaningful coefficient with the rest of the coefficients being mere noise terms in the decryption basis, defined in [2]. The procedure, which will be sketched here and applied later, isolates the message-encoding coefficient by applying the trace function and performs a homomorphic rounding on this coefficient to recover the message.

The following sketch applies to encryption schemes over the $2n'th$ cyclotomic ring, where n is a power of two. This is isomorphic to the polynomial ring $R = \mathbb{Z}[x]/(x^n + 1)$ by identifying any abstract element ω of order $2n$ over $\mathbb{Q}$ with x. Just as with the example scheme, we have two layers of noise controlled by two moduli: the ciphertext modulus q and the plaintext modulus p. However, unlike the example scheme, we will allow the ciphertext to lie in an extension ring of the message space $\mathbb{Z}_p$, namely $R_q = \mathbb{Z}_q/(x^n + 1)$. This is an extension of degree n, meaning there are n automorphisms φ_i on R that fix $\mathbb{Z}$ pointwise, defined by $\varphi_i(\omega) = \omega^i$ for $i \in \mathbb{Z}_{2n}^*$. In the ring R, this corresponds to evaluating the polynomial $a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$ at x^i: $\varphi_i(a(x)) = a(x^i)$. The trace function $\mathrm{Tr} = \mathrm{Tr}_{R/\mathbb{Z}} : R \to \mathbb{Z}$ is defined as the sum of these automorphisms:

$$\mathrm{Tr}_{R/\mathbb{Z}}(a) = \sum_i \varphi_i(a) \in \mathbb{Z}.$$

Note: if $a \in \mathbb{Z}$, then $\mathrm{Tr}_{R/\mathbb{Z}}(a) = na$.

Suppose the input of the bootstrapping procedure is a ciphertext $c \in R_q$ encrypting a message $m \in \mathbb{Z}_p$ under a secret key sk such that

$$\mathrm{Dec}(sk, c) = \frac{q}{p}m + e \underline{\mathrm{mod}}\, q = v$$

for moduli $p \ll q$, $\gcd(q,p) = 1$ and some manageable error term $e \in R_q$, meaning $\lfloor \frac{q}{p}m + e \rceil_p = \lfloor \frac{p}{q}(\frac{q}{p}m + e) \rceil = m$. The bootstrapping consists of the following steps:

1. Convert the ciphertext c to a ciphertext c' over a larger ring R'_Q encrypting a plaintext $u' \in R_{q'}$ for $q' = nq$ such that $u' \equiv v \mod q$. This procedure works in the following substeps:
   (a) Reinterpret c as a noiseless encryption of $v = \frac{q}{p}m + e \in R_q$, i.e. regarding v as a plaintext: this requires both the plaintext and ciphertext ring to be R_q.
   (b) Change both the moduli to $q' = nq$, which will yield a noiseless encryption of some $u' \in R_{q'}$ such that $u' \equiv v \mod q$.
   (c) Convert to a noiseless ciphertext c' that still encrypts $u' \in R_{q'}$, but using a larger ciphertext ring R'_Q for $R' = \mathbb{Z}/(x^N + 1)$ and modulus $Q \gg q'$. This simply involves embedding the ring R into R' and scaling the ciphertext by a factor of Q/q'.

2. Homomorphically apply the scaled trace function $\frac{1}{n}\mathrm{Tr}_{R/\mathbb{Z}}$ to the encryption of the "new" plaintext $u' \in R_{q'}$ to obtain an encryption of the plaintext
$$u = \frac{1}{n}\mathrm{Tr}_{R/\mathbb{Z}}(u') = \frac{q}{p}m + \bar{e} \in \mathbb{Z}_q,$$
where the error term $\bar{e}$ now lies in $\mathbb{Z}$. Note that this changes the plaintext ring from $R_{q'}$ to $\mathbb{Z}_q$.

3. Homomorphically apply a ring rounding function from $\lfloor \cdot \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$, which will yield an encryption of $\lfloor u \rceil_p = \lfloor u(p/q) \rceil = m$. This changes the plaintext ring from $\mathbb{Z}_q$ to $\mathbb{Z}_p$, and hence concludes the bootstrapping procedure.

Note that the two final steps are described only in terms of the plaintext, as the ciphertext ring R' and modulus Q may be made smaller as is secure and convenient. They are introduced as a security measure: Q is needed to prevent the growth of the noise to warp the message during the homomorphic operations. This necessitates the introduction of N as the new ring dimension, as increasing the ciphertext modulus typically lowers the security of the scheme, whereas increasing the ring dimension increases the security. Embedding R into R' should therefore balance the security of the scheme.

Correctness of the procedure needs to be shown, namely showing the resulting ciphertext will decrypt correctly. This will also merely be sketched, for a full discussion, see [2].

The first step is showing that the ciphertext actually decrypts, which might not be obvious seeing as we are applying a function to it. Assume the decryption algorithm is based on $\text{Dec}(c, sk) = sk{*}c$, and that c is an encryption of m. Then, for any automorphism φ described above, we have $\varphi(sk) * \varphi(c) = \varphi(sk * c)$, as any automorphism is a homomorphism. This means that $\varphi(c)$ is an encryption of $\varphi(m)$ under the secret key $\varphi(sk)$. If we can switch keys such that all $\varphi_i(c)$ are encrypted under the same secret key sk', the ciphertext $\text{Tr}_{R/\mathbb{Z}}(c')$ therefore encrypts u under the secret key sk'.

It also needs to be shown that $\bar{e}$ is a tolerable error for the encryption scheme under the assumption that e is, as $u' \equiv v \mod q$. The trace function applied to u' will therefore result in

$$\text{Tr}_{R/\mathbb{Z}}(\frac{q}{p}m + e) = \text{Tr}_{R/\mathbb{Z}}(\frac{q}{p}m) + \text{Tr}_{R/\mathbb{Z}}(e) = n\frac{q}{p}m + \text{Tr}_{R/\mathbb{Z}}(e) \in \mathbb{Z}_q,$$

seeing as $q, p, m \in \mathbb{Z}_p$. According to Corollary 2.2 of [11], $\|\text{Tr}_{R/\mathbb{Z}}(a)\| \leq \|a\|\sqrt{n}$, and it follows that $\|\frac{1}{n}\text{Tr}(e)\| \leq \frac{1}{\sqrt{n}}\|e\|$. Furthermore, the authors of [2] argue that $\bar{e}$ in fact is a subvector of e according to the decryption basis, and it thus follows that $\bar{e}$ is a manageable error, assuming e is.

4 Lattices and Short Vector Problems, Ideal Lattices and Learning with Errors over Rings

A vector space $\mathcal{V}$ is a set of vectors $\{\mathbf{v}_i\}$ which is closed under addition and multiplication by a scalar from any field. A lattice $\mathcal{L}$ is very similar to a vector space, only here we are restricted to multiplying only with integers.

Definition 4.1. *Let $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_\eta\}$ be a set of linearly independent vectors, with $\mathbf{v}_i \in \mathbb{R}^m \ \forall i \in \{1, \dots, \eta\}$. The lattice $\mathcal{L}$ generated by $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_\eta$ is the set of linear combinations of these vectors with coefficients in $\mathbb{Z}$:*

$$\mathcal{L} = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_\eta\mathbf{v}_\eta : a_1, a_2, \dots, a_\eta \in \mathbb{Z}\}.$$

A basis for the lattice $\mathcal{L}$ is any set of independent vectors that generates $\mathcal{L}$. As is the case with vector spaces, any two such sets will have the same number of elements - the same dimension. Another feature of vector spaces that applies to lattices is that for any $\mathcal{L} \subset \mathbb{R}^m$ of dimension η, the basis may be represented as an $\eta \times m$-matrix $\mathbf{B}$, where the basis vectors of $\mathcal{L}$ form the rows of the matrix.

Suppose $m = \eta$, then the aforementioned matrix of basis vectors will be square, and so we may calculate the determinant of it. There are of course many possible bases of a lattice $\mathcal{L}$; suppose one basis consists of the vectors $B = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_\eta\}$, the other $B' = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_\eta\}$. Because they are both bases, each vector in either basis may be expressed as a linear combination of the vectors in the other:

$$\mathbf{w}_i = a_{i1}\mathbf{v}_1 + a_{i2}\mathbf{v}_2 + \cdots + a_{i\eta}\mathbf{v}_{i\eta}.$$

We may thus form a new matrix, to change from basis B to B', with $\mathbf{B}' = \mathbf{A} \cdot \mathbf{B}$:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1\eta} \\ a_{21} & a_{22} & \dots & a_{2\eta} \\ \vdots & \vdots & \ddots & \vdots \\ a_{\eta 1} & a_{\eta 2} & \dots & a_{\eta\eta} \end{bmatrix}.$$

However, seeing as B' is a basis as well, we might just as well express any $\mathbf{v}_i \in B$ as a linear combination of the vectors in B': $\mathbf{B} = \mathbf{A}' \cdot \mathbf{B}'$. Obviously, we must have $\mathbf{A}' = \mathbf{A}^{-1}$, and furthermore that all the entries of either matrix are integers. It follows that the determinants of $\mathbf{A}$ and $\mathbf{A}^{-1}$ are either both 1 or -1, and so that $|\det(\mathbf{B})| = |\det(\mathbf{B}')|$. Finally, we have the following definition of a lattice invariant:

Definition 4.2. *Let $\mathcal{L}$ be a lattice of dimension η with basis $B = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_\eta\}$, where $\mathbf{v}_i \in \mathbb{R}^\eta \quad \forall i \in \{1, 2, \dots, \eta\}$. The determinant of $\mathcal{L}$ is defined as*

$$\det(\mathcal{L}) = |\det(\mathbf{B})|.$$

Just as we might refer to the length of a vector in a vector space, we may refer to the length of a vector in a lattice $\mathcal{L}$ over $\mathbb{R}^m$ by defining

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_m^2}\,,$$

where v_i is the i'th component of $\mathbf{v}$.

Based on this, we have the following problem: find the shortest vector of a lattice $\mathcal{L}$. More formally [15]:

The shortest vector problem (SVP): Find a shortest nonzero vector in a lattice $\mathcal{L}$, i.e. find a nonzero vector $\mathbf{v} \in \mathcal{L}$ that minimizes $\|\mathbf{v}\|$.

Note that the problem does not ask for *the* such vector, as we have no reason to believe that a vector satisfying these conditions will be unique; at the very least, if $\mathbf{u}$ is a solution, so is $-\mathbf{u}$. It may be shown that solving SVP is NP-hard under the randomized reduction hypothesis [15].

Due to this proven hardness of the problem, SVP is used in cryptographic settings, so that breaking an encryption scheme requires solving SVP for a certain instance. However, in such cases, solving SVP precisely is not always necessary; in some settings it may suffice to compute merely an approximation of the vectors in question. This is known as approximate-SVP, with the following formal definition [15]:

Approximate-SVP: Let $\psi(\eta)$ be a function of the lattice dimension η of a lattice $\mathcal{L}$, with a shortest vector $\mathbf{v}_0$. Find a vector $\mathbf{v} \in \mathcal{L}$ such that

$$\|\mathbf{v}\| \leq \psi(\eta)\|\mathbf{v}_0\|.$$

Of course, the length of the shortest vector $\mathbf{v}_0 \in \mathcal{L}$ is not always given, but an upper bound on $\|\mathbf{v}_0\|$ is always given by the following theorem:

Theorem 4.3 (Hermite's Theorem (Theorem 7.25 [15]))**.** *Every lattice $\mathcal{L}$ of dimension η has at least one vector $\mathbf{v} \in \mathcal{L}$ satisfying $\|\mathbf{v}\| \leq \sqrt{\eta}\det(\mathcal{L})^{1/\eta}$.*

Another result by Hermite is that for every lattice $\mathcal{L}$ there exists a constant, γ_η, known as the Hermite constant, such that $\|\mathbf{v}_0\| \leq \sqrt{\gamma_\eta}\det(\mathcal{L})^{1/\eta}$. This constant is only known for $1 \leq \eta \leq 8$ and 24. This gives rise to another, related, lattice problem, the Hermite shortest vector problem [9]:

HSVP: Given a lattice $\mathcal{L}$ and an approximation factor $\alpha > 0$, find a non-zero vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \leq \alpha\det(\mathcal{L})^{1/\eta}$

Often α may be expressed as δ^η, in which case δ is known as the Hermite root factor.

Ideal lattices are lattices with additional algebraic structure. Whereas lattices may be regarded as groups, ideal lattices may be viewed as ideals, meaning that we must regard it as a subset of some ring for this to make sense. More precisely: given a ring R with an ideal I, there must be an embedding of this ideal into a lattice. For a concrete example, suppose $n = 2^k$ for some $k \in \mathbb{N}$, and let R be the ring $R = \mathbb{Z}[x]/(x^n + 1)$. Then any element in this ring is of the form $a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$, with $a_i \in \mathbb{Z}\ \forall i \in \{0, 1, \ldots, n-1\}$. This may be embedded into $\mathbb{C}^n$ with the canonical embedding, defined by $\sigma(a(x)) = (a(\omega), a(\omega^3), \ldots, a(\omega^{2n-1}))$, where ω is defined as $\exp(2\pi i/2n)$. Due to the fact that an embedding in particular is a homomorphism, and that ideals are closed under addition (i.e. $a + b \in I\ \forall a, b \in I$), it follows that for any $I \in R$, $\sigma(I)$ is a lattice in $\mathbb{C}^n$ [17]. This is an example of an ideal lattice.

Just as there are problems related to regular lattices, there are problems related to ideal lattices, one of them being the learning with errors over rings (RLWE). However, whilst for example SVP is defined over the lattice itself, this is not the case for RLWE, as this problem is defined over the ring in which an ideal corresponds to an ideal lattice. Although RLWE is a general problem, it will only be discussed for a certain class of rings here, see [17] for more details.

Taking R to be the ring in the previous paragraph, we may define the quotient ring $R_q = R/qR = \mathbb{Z}_q[x]/(x^n + 1)$, where q is an integer larger than 2. For this ring, RLWE is: let $s = s(x) \in R_q$ be chosen uniformly at random from R_q and be kept secret. We then draw several random elements $\{a_i\}$ from R_q such that $i < \text{poly}(n)$ and compute $b_i = a_i * s + e_i \in R_q$ for some noise term e_i drawn at random from a certain error distribution, typically a Gaussian of some kind, and is considered small in some way (this will not be specified further). There are two versions of the RLWE problem: the search and the decision problem.

The search problem: given $[a_i, b_i] \in R_q \times R_q$ for $i < \text{poly}(n)$, find s.

The decision problem: given $[a_i, b_i]$ and $[a_i, a_i'] \in R_q \times R_q$ where all a_i' are drawn uniformly at random from R_q, distinguish with noticeable advantage the set calculated using s and e_i and set chosen uniformly at random.

There are reductions amongst these and the approximate-SVP for ideal lattices. More precisely, there is a quantum reduction from the worst case approximate-SVP on ideal lattices in R to the search problem in R_q, meaning that if one is able to find an algorithm which is able to decide s given $[a_i, b_i]$, one may use this to construct a quantum algorithm solving approximate-SVP in *any* ideal lattice. Furthermore, there is a classical reduction from the search to the decision problem in R_q, assuming q is a prime such that $q \equiv 1 \mod 2n$, which translates to: being able to distinguish $[a_i, b_i]$ from $[a_i, a_i']$ makes one able to calculate s. This also means that if it is not feasible to solve the search problem in RLWE, then the same RLWE distribution is in fact pseudorandom. This fact may be exploited in a cryptographic setting [17].

Throughout the rest of this paper, we have the following: for n a power of two, the ring R is $R = \mathbb{Z}[x]/(x^n + 1)$. For a positive integer q, we have the quotient ring $R_q = R/qR$, as explained above. Finally, the following lemmas will be used throughout, for any two elements $a = \sum_{i=0}^{n} a_i x^i, b = \sum_{i=0}^{n} b_i x^i \in R$.

The lemmas both view these elements and their product as coefficient vectors, where

$$\begin{aligned}
a * b &= (a_0 + \cdots + a_{n-1}x^{n-1}) * (b_0 + \cdots + b_{n-1}x^{n-1}) \\
&= (a_0b_0 - a_1b_{n-1} - \cdots - a_{n-1}b_1) \\
&+ (a_0b_1 + a_1b_0 - \cdots - a_{n-1}b_2)x + \dots \\
\dots &+ (a_0b_{n-1} + a_1b_{n-2} + \cdots + a_{n-1}b_0)x^{n-1}.
\end{aligned}$$

Thus, the coefficients of $a * b$ are the inner products of the coefficient vector of a and some rotation of the coefficient vector of b.

Lemma 4.4. *The following bound holds for any two elements $a, b \in R$:*

$$\|a * b\| \leq \sqrt{n}\|a\|\|b\|.$$

Proof. Using the fact that every coefficient of $a * b$ is the inner product of the coefficient vectors of a and a rotation of b, by Cauchy-Schwarz each such coefficient will be $\leq \|a\|\|b\|$. The bound $\|a * b\| \leq \sqrt{n}\|a\|\|b\|$ immediately follows. □

Lemma 4.5. *The following bound holds for any two elements $a, b \in R$:*

$$\|a * b\|_\infty \leq n\|a\|_\infty\|b\|_\infty.$$

Proof. Seeing as $a_i \leq \|a\|_\infty,\ b_i \leq \|b\|_\infty\ \forall i \in \{0, 1, \dots, n-1\}$, it follows that $a_ib_j \leq \|a\|_\infty\|b\|_\infty$, and thus that every coefficient of $a * b \leq n\|a\|_\infty\|b\|_\infty$. □

5 An implementation of FHE built on NTRU

The following encryption scheme is built on the lattice-based encryption scheme NTRU [14] with additional procedures presented in Subsections 5.2, 5.3 and 5.4 that provide sufficient noise reduction to make the scheme fully homomorphic when combined with the bootstrapping sketched in Section 3. It will be shown in Subsection 5.7 that for these procedures to be correct, the parameters q and n have to be so big that the scheme is susceptible to a sub-field lattice attack, which will be described in the same subsection.

Due to this, the scheme should for all intents and purposes be considered insecure. The efficiency and performance of the scheme is therefore not presented, as it is of little interest. Nor will the bootstrapping procedure be presented in any detail; it is presented for completeness, as it is a necessity for the scheme to be fully homomorphic, not merely somewhat homomorphic. As the sketch in Section 3 shows, the bootstrapping does not impose any particular bounds on the parameters q and n, nor does it require publishing any extra information regarding the scheme. The conclusion that the scheme is insecure may therefore be drawn without the details of the bootstrapping procedure, which is why they are not presented.

5.1 The Homomorphic Encryption Scheme

The message space of the scheme is R_p for some integer $p \geq 2$, whilst most arithmetic operations are performed modulo q chosen such that $q \gg p$ and $\gcd(p, q) = 1$. The scheme consists of the following operations:

KeyGen: Choose a short $f \in R$ such that $f \equiv 1 \mod p$ and $\exists f^{-1} \ \underline{\text{mod}}\ q$, i.e. $f * f^{-1} = f^{-1} * f = 1 \ \underline{\text{mod}}\ q$. Choose a short $g \in R$ as well, and output $pk = h = g * f^{-1} \ \underline{\text{mod}}\ q$ and $sk = f$.

Enc($pk = h, m \in R_p$): Choose a short $e \in R$ such that $e \equiv m \mod p$ and a short $r \in R$. Output $c = pr * h + e \ \underline{\text{mod}}\ q$, $d = 1$.

Dec($sk = f, c \in R_q$, d): Compute $\bar{b} = f^d * c \ \underline{\text{mod}}\ q$ and lift this to the integer polynomial $b \in R$ with coefficients in $(-q/2, q/2]$. Output $m = b \ \underline{\text{mod}}\ p$.

EvalAdd(c_0, c_1, d_0, d_1): Output: $c = c_0 + c_1 \ \underline{\text{mod}}\ q$, $d = \max(d_0, d_1)$.

EvalMult(c_0, c_1, d_0, d_1): Output: $c = c_0 * c_1 \ \underline{\text{mod}}\ q$, $d = d_0 + d_1$.

The polynomials f, g, r and e will typically be chosen according to some discrete Gaussian and the requirement of these polynomials being short in order to ensure correct decryption. What precisely this entails will be discussed at some length thoughout this section.

The two last operations are the homomorphic operations, and it is also these that necessitate the notion of the degree d of a ciphertext, which denotes the power of f^{-1} in the ciphertext. Note that $f^k * b = m \ \underline{\text{mod}}\ p$ for any power $k \geq 0$, whilst this is not necessarily the case for f^{-1}, as there is no guarantee that $f^{-1} = 1 \underline{\text{mod}}\ p$. Therefore, the decryption procedure will decrypt any ciphertext of degree at most the given d, assuming $f^d * c = f^k * b \underline{\text{mod}}\ q$, which is the reason $d = \max(d_0, d_1)$ in EvalAdd.

It should come as no surprise that the main challenge of this scheme is keeping the noise of the ciphertext manageable as long as it is being processed; that is, keeping it low enough so that the decryption will be valid. The scheme has several procedures to handle the growth of this noise, which may be combined to reduce the noise of a ciphertext as much as possible before bootstrapping is necessary.

Proposition 5.1. *The above encryption scheme decrypts freshly generated ciphertexts correctly, under certain assumptions.*

Proof. Suppose f, g, r and e are all chosen according to the stated scheme and let $c = pr * h + e \underline{\text{mod}}\ q$. The decryption of c proceeds as follows, when viewed as an operation in R, as opposed to R_q:

$$\begin{aligned} \bar{b} &= f * c = f * (pr * h + e) \\ &= pf * r * g * f^{-1} + f * e \\ &= pq * r * g * f' + pr * g + f * e, \end{aligned}$$

where $f * f^{-1} = qf' + 1$. The first step of the decryption procedure is completed by reducing the given polynomial modulo q. Consider the polynomial $pr*g+f*e$ as a member of R. To ensure correct decryption, we need every coefficient of this polynomial to be of absolute value less than $q/2$, that is: $pr * g + f * e$ must equal b, as defined in the decryption procedure. If this is not the case, we get

$$b = pr * g + f * e - q \sum_{i=0}^{n-1} a_i x^i \quad \exists a_i \neq 0$$

and hence, $b \underline{\text{mod}}\ p$ need not equal m. The limitations we place on f, g, r and e are the following: any coefficient of the polynomial $pr * g + f * e$ must lie in the interval $(-q/2, q/2)$. In other words, we require:

$$\|pr * g + f * e\|_\infty < q/2.$$

Using the triangle inequality and Lemma 4.5, we may compute:

$$\begin{aligned} \|pr * g + f * e\|_\infty &\leq \|pr * g\|_\infty + \|f * e\|_\infty && (1) \\ &\leq pn\|r\|_\infty\|g\|_\infty + n\|f\|_\infty\|e\|_\infty \\ &\leq pn\|r\|_\infty\|g\|_\infty + pn\|f\|_\infty\|e\|_\infty \\ &\leq 2pnB^2, && (2) \end{aligned}$$

for B a bound on the largest coefficient of r, g, f and e. If we assume (1) is less than $q/2$, then any fresh ciphertext will decrypt correctly. This requires the polynomials r, g, f and e to be sampled from a distribution which ensures that any coefficient is strictly less than $\sqrt{\frac{q}{4pn}}$. □

5.2 Key Switching

Key switching converts a ciphertext of degree at most d encrypted under f_1 into a ciphertext of degree 1 encrypted under the secret key f_2, not necessarily different from f_1. This procedure requires a hint, namely

$$a_{1\to 2} = \bar{a} * f_1^d * f_2^{-1} \underline{\bmod} \, q,$$

where $R \ni \bar{a} \equiv 1 \mod p$ is short. Based on this hint, the actual key switching is the procedure

KeySwitch$(c_1, a_{1\to 2})$: Output: $c_2 = a_{1\to 2} * c_1 \underline{\bmod} \, q$.

Proposition 5.2. *Suppose c_1 is an encryption of m and decrypts correctly: Dec$(f_1, c_1, d) = m$. Then, for $a_{1\to 2}$ and c_2 generated according to the described procedure, Dec$(f_2, c_2, 1) = m$, assuming manageable noise.*

Proof.

$$\begin{aligned} \bar{b}_2 &= f_2 * c_2 = f_2 * a_{1\to 2} * c_1 = f_2 * \bar{a} * f_1^d * f_2^{-1} * c_1. \\ &\equiv \bar{a} * f_1^d * c_1 \equiv \bar{a} * \bar{b}_1 \underline{\bmod} \, q. \end{aligned}$$

Assuming $\|\bar{a} * \bar{b_1}\|_\infty < q/2$, we get:

$$b_2 = \bar{a} * b_1 = \bar{a} * m = m \underline{\bmod} \, p.$$

□

Seeing as c_1 is a ciphertext of degree d, it must be the case that it is the result of $d-1$ multiplications, wlog let $\bar{b_1} = f_1^d * (pr * g * f_1^{-1} + e)^d \underline{\bmod} \, q$. For the assumption $\|\bar{a} * \bar{b_1}\|_\infty < q/2$ to hold, we must therefore have:

$$\begin{aligned} \|\bar{a} * f_1^d * (pr * g * f_1^{-1} + e)^d\|_\infty &= \|\bar{a} * f_1^d * \sum_{i=0}^{d} \binom{d}{i} p^i r^i * g^i * f_1^{-i} * e^{d-i}\|_\infty \\ &= \|\bar{a} * \sum_{i=0}^{d} \binom{d}{i} p^i r^i * g^i * f_1^{d-i} * e^{d-i}\|_\infty \\ &\le n^{2d} \|\bar{a}\|_\infty \sum_{i=0}^{d} \binom{d}{i} p^i \|r\|_\infty^i \|g\|_\infty^i \|f\|_\infty^{d-i} \|e\|_\infty^{d-i} \\ &\le p^d n^{2d} B^{2d+1} \sum_{i=0}^{d} \binom{d}{i} \\ &\le 2^d p^d n^{2d} B^{2d+1} < q/2. \end{aligned}$$

Again, B is a bound on the largest coefficient in $\bar{a}, r, g, f$ and e. In the cases $d = 1$ and $d = 2$, we have:

$$B^3 < \frac{q}{4pn^2} \quad \text{and} \quad B^5 < \frac{q}{8p^2n^4}. \tag{3}$$

5.3 Ring Reduction

Ring reduction maps a ciphertext from a ring of dimension n to a ring of smaller dimension n'; typically $n' = n/2$, although n may be divided by higher powers of 2. The ring reduction procedure uses a decomposition algorithm which essentially chooses certain coordinates of $c = [c_0, c_1, \dots, c_{n-1}]$ wrt. the power basis. To ensure that data is not lost during the ring reduction, we first employ a key switching procedure on c to ensure that it is on the appropriate form. The ring reduction requires the following decomposition procedure:

1. Express c as the coefficient vector, $c = [c_0, c_1, \dots, c_{n-1}]$, and let $w = n/n'$.

2. Output the ciphertexts c'_i for each $i = 0, \dots, w-1$ where

$$c'_i = [c_i, c_{w+i}, c_{2w+i}, \dots, c_{(n'-1)w+i}].$$

So: c'_i simply consists of the entries of c whose indices are $i \mod w$. In the typical case of $n/n' = 2$, the decomposition process outputs $c'_0 = [c_0, c_2, \dots, c_{n-2}]$ and $c'_1 = [c_1, c_3, \dots, c_{n-1}]$. In order to perform the reduction, the plaintext data has to be fully contained in one of these vectors to avoid any loss of this information.

The actual procedure of reducing the ring $\mathbb{Z}/(x^n + 1)$ to $\mathbb{Z}/(x^{n'} + 1)$ is the following, assuming the input ciphertext c_0 only has plaintext data in its indices $0 \mod w$:

1. Choose a sparse secret key f with nonzero coefficients wrt. the power basis only in indices $0 \mod w$.

2. Obtain a new ciphertext c by switching the keys so that f is the secret key.

3. Decompose c and f and output c'_0 as the new ciphertext encrypted under $f'_0 = [f_0, f_w, f_{2w}, \dots, f_{(n'-1)w}]$.

Proposition 5.3. *Suppose a ciphertext c_1 of degree d which is encrypted under f_1 only has plaintext data in coefficients of indices $i \equiv 0 \mod w$, and furthermore that $w = 2$. Suppose further that there exists a secret key f_2 which may be expressed as $[f_{2_0}, 0, f_{2_2}, 0, \dots, f_{2_{n-2}}, 0]$ in the power basis such that* $\mathrm{Dec}(f_1, c_1, d) = \mathrm{Dec}(f_2, c_2 = a_{1\to 2} * c_1, 1)$. *Then*

$$\mathrm{Dec}(f_1, c_1, d) = \mathrm{Dec}(f'_2, c'_2, 1),$$

for $f'_2 = [f_{2_0}, f_{2_2}, \dots, f_{2_{n-2}}]$ and $c'_2 = [c_{2_0}, c_{2_2}, \dots, c_{2_{n-2}}]$.

Proof. Seeing as c_1 only has plaintext data in the indices $0, 2, n-2$, it follows that the encrypted message m may be expressed as $[m_0, 0, m_2, 0, \dots, m_{n-2}, 0]$ in the power basis. As c_1 is encrypted under f_1, it must be the case that

$$f_1^d * c_1 = m + pr' \ \underline{\mathrm{mod}}\, q$$

for some $r' \in R$. It follows that, as $c_2 = \bar{a} * f_1^d * f_2^{-1} * c_1 \underline{\text{mod}}\, q$,

$$c_2 = \bar{a} * f_2^{-1} * (m + pr') = f_2^{-1} * m + ps \underline{\text{mod}}\, q,$$

for some $s \in R$, where the second equality follows from $\bar{a} = 1 \underline{\text{mod}}\, p$.
We have, with a slight abuse of notation:

$$\begin{aligned} f_2 * c_2 &= [f_{2_0}, 0, f_{2_2}, 0, \ldots, f_{2_{n-2}}, 0] * [c_{2_0}, c_{2_1}, \ldots, c_{2_{n-2}}, c_{2_{n-1}}] \\ &= f_{2_0}[c_{2_0}, c_{2_1}, \ldots, c_{2_{n-2}}, c_{2_{n-1}}] \\ &\quad + f_{2_2}[-c_{2_{n-2}}, -c_{2_{n-1}}, \ldots, c_{2_{n-4}}, c_{2_{n-3}}] + \ldots \\ \ldots &+ f_{2_{n-2}}[-c_{2_2}, -c_{2_3}, \ldots, c_{2_0}, c_{2_1}] \\ &= [m_0 + pt_0, pt_1, \ldots, m_{n-2} + pt_{n-2}, pt_{n-1}] \underline{\text{mod}}\, q, \end{aligned}$$

again, for some $t \in R$. This equality follows from the assumption that

$$\text{Dec}(f_2, c_2, 1) = (f_2 * c_2 \underline{\text{mod}}\, q) \underline{\text{mod}}\, p = m = [m_0, 0, m_2, \ldots, 0, m_{n-2}, 0].$$

Finally, with a similar abuse of notation as previously:

$$\begin{aligned} f_2' * c_2' &= [f_{2_0}, f_{2_2}, \ldots, f_{2_{n-2}}] * [c_{2_0}, c_{2_2}, \ldots, c_{2_{n-2}}] \\ &= f_{2_0}[c_{2_0}, c_{2_2}, \ldots, c_{2_{n-2}}] \\ &\quad + f_{2_2}[-c_{2_{n-2}}, c_{2_0}, \ldots, c_{2_{n-4}}] + \ldots \\ \ldots &+ f_{2_{n-2}}[-c_{2_2}, -c_{2_4}, \ldots, c_{2_0}] \\ &= [m_0 + pt_0, m_2 + pt_2, \ldots, m_{n-2} + pt_{n-2}] \underline{\text{mod}}\, q. \end{aligned}$$

We conclude that $\text{Dec}(f_2', c_2', 1) = [m_0, m_2, \ldots, m_{n-2}]$. Thus, the ciphertext decrypts correctly and the plaintext is preserved. □

5.4 Modulus Switching

Modulus switching converts a ciphertext from modulus q to a smaller modulus, $\bar{q} = q/q'$ for some factor q' of q, whilst also reducing the underlying noise by a factor of approximately q'. This works by adding a small multiple of p, Δ, to c which is equivalent to $-c$ modulo q', so that $c + \Delta$ is divisible by q'. This should only cause a slight increase in the noise of the ciphertext, and thus ensure that the underlying message is preserved. Here q' divides q, so it follows that $\gcd(q', p) = 1$, thus there exists an inverse of q' modulo p: $v = (q')^{-1} \underline{\text{mod}}\, p$. The procedure ModSwitch(c, q, q') is performed as follows:

1. Compute a short $d \in R$ such that $d = c \underline{\text{mod}}\, q'$.
2. Compute a short $\Delta \in R$ such that $\Delta = (q'v - 1)d \underline{\text{mod}}\, (pq')$.
   This ensures that all the coefficients of Δ lie in the interval $(-pq'/2, pq'/2]$.
3. Let $d' = c + \Delta \underline{\text{mod}}\, q$. Note that q' divides d' by construction.
4. Output $c' = (d'/q') \in R_{\bar{q}}$.

Note that the final step indirectly multiplies d with v, which may easily be compensated for by either multiplying with q' in the final step of the decryption procedure, or ensuring that $q' \equiv 1 \mod p$. This will be the case if $p = 2$, seeing as this requires any factor of q to be odd.

The procedure is easiest and most efficiently implemented when $q = q_1 \dots q_t$ is a product of several small and pairwise relatively prime moduli, and q' is one of these.

Proposition 5.4. *Suppose c is an encryption of degree 1 of the message m under the secret key f. Let $c' = \mathrm{ModSwitch}(c, q, q')$, then*

$$v\mathrm{Dec}(f, c, 1) = \mathrm{Dec}(f, c', 1),$$

assuming the noise is manageable.

Proof. Let $\bar{q} = q/q'$. As $d = c \underline{\bmod} q'$ and $v = (q')^{-1} \underline{\bmod} p$, we may write

$$\begin{aligned} d &= c - q'l && \text{for } l \in R, \\ q'v &= 1 + pk && \text{for } k \in \mathbb{Z}. \end{aligned}$$

Following the procedure, we have[1]:

$$\begin{aligned} &(q'v - 1)d = (pk + 1 - 1)(c - q'l) = pk(c - q'l) = pkc - pq'kl. \\ &\quad \Rightarrow \Delta = pkc - pq's \text{ for } s \in R, \quad \text{as} \quad R \ni \Delta = (q'v - 1)d \equiv pkc \underline{\bmod}\, pq'. \end{aligned}$$

$$\begin{aligned} d' &= c + \Delta \underline{\bmod}\, q \\ &= c + pkc - pq's = (1 + pk)c - pq's \\ &\equiv q'vc - pq's \underline{\bmod}\, q. \end{aligned}$$

$$c' = d'/q' \equiv vc - ps \underline{\bmod}\, \bar{q}.$$

If we assume we have equality rather than equivalence in the final equation, that is $\|vc - ps\|_\infty < \bar{q}/2$, we will have the following:

$$\begin{aligned} f * c' &= vf * c - pf * s = v(pr * g(qf' + 1) + f * e) - pf * s \in R \\ &\equiv vpg * r + vf * e - pf * s \underline{\bmod}\, \bar{q} \end{aligned}$$

Assuming we have manageable noise, $\|vf * e + vpg * r - pf * s\|_\infty < \bar{q}/2$, we will have equality rather than equivalence, and finally:

$$(f * c' \underline{\bmod}\, \bar{q}) \underline{\bmod}\, p = vm$$

It follows that the slightly altered decryption algorithm described prior to this proposition will output m. □

[1]Throughout this proof, pk denotes multiplication of the integers p and k, not the public key.

Thus, for the decryption of c' to be successful, we need to have the following:

$$\begin{aligned} \|f * c'\|_\infty = \|f * (c+\Delta)/q'\|_\infty &\leq \frac{1}{q'}(\|f * c)\|_\infty + \|f * \Delta\|_\infty) \\ \Rightarrow \frac{1}{q'}(\|pg * r + f * e\|_\infty + \|f * \Delta\|_\infty) &\leq \frac{1}{q'}(2pnB^2 + nB\|\Delta\|_\infty) \\ &\leq \frac{1}{q'}(2pnB^2 + nB\frac{pq'}{2}) < \bar{q}/2 = q/2q'. \end{aligned} \tag{4}$$

Where, as usual, B is the bound on the biggest coefficient of the polynomials r, g, e and f.

5.5 ComposedEvalMult

The procedure ComposedEvalMult is quite simply the sequential execution of the operations Evalmult, key switching, ring reduction and modulus switching. This reduces the noise of the ciphertext in question, and hence enables more computations to be performed before calling on bootstrapping to refresh the ciphertext. The growth of noise is usually much less severe during addition compared to multiplication, meaning measures such as modulus switching are typically not necessary, and therefore not performed after addition.

However, the operations constituting ComposedEvalMult also alter how secure the scheme is, as is to be expected when reducing the dimension of the ring and/or the ciphertext modulus. In particular, performing a modulus switching results in a more secure scheme, whereas reducing the ring dimension yields a less secure scheme. The reasons for this will be explained in Subsection 5.7.

A consequence of this is that ring reduction may not be performed in every call of ComposedEvalMult, but rather when the resulting scheme is sufficiently secure. This is of little consequence as far as noise reduction is concerned, as the operation does not actually reduce the noise of the scheme, as the proof of Proposition 5.3 shows. However, reducing the ring dimension removes "meaningless" coefficients, i.e. coefficients that are not needed to perform a correct decryption. This might lower the probability of decryption error if the ciphertext is processed further, as any resulting coefficients will be the sum of fewer summands. Still, the main advantage of the procedure is the fact that lower dimensionial rings are easier to work with, hence reducing the time of computation.

As a result of this, only the sequential execution of key switching and modulus switching will be considered in the following proof of correctness for ComposedEvalMult:

Proposition 5.5. *Suppose c_0, c_1 are two ciphertexts encrypted under the public key $h = g * f_1^{-1}$, both of degree 1. Then*

$$\text{Dec}(f_2, \text{ComposedEvalMult}(c_0, c_1), 1) = \text{Dec}(f_1, c_0, 1) * \text{Dec}(f_1, c_1, 1),$$

where f_2 is the new secret key after KeySwitch has been performed in ComposedEvalMult, under the assumption that the noise is manageable.

Proof. Based on the proofs of propositions 5.2 and 5.4, it follows that

$$f_2 * \text{ComposedEvalMult}(c_0, c_1) \equiv \bar{b} \underline{\text{ mod }} \bar{q},$$

where $\bar{b} = m_0 * m_1 \underline{\text{ mod }} p$. What needs to be shown is that the noise added during multiplication and switching keys is sufficiently lowered by switching the modulus, in other words: the noise of the procedure does not warp the decryption of ComposedEvalMult(c_0, c_1). The ciphertext ComposedEvalMult(c_0, c_1) outputs is on the form $c = \frac{1}{q'}(a_{1\to 2} * c_0 * c_1 + \Delta)$ for a factor q' of q. We have the following:

$$\begin{aligned}
f_2 * c = f_2 * \frac{1}{q'}(a_{1\to 2} * c_0 * c_1 + \Delta)& \\
= \frac{1}{q'} f_2 * (\bar{a} * f_2^{-1} * f_1^2 * (pr_0 * g * f_1^{-1} + e_0)(pr_1 * g * f_1^{-1} + e_1) + \Delta)& \\
= \ldots \equiv \frac{1}{q'}(p^2 \bar{a} * r_0 * r_1 * g^2 + p\bar{a} * r_0 * g * f_1 * e_1& \\
+ p\bar{a} * r_1 * g * f_1 * e_0 + \bar{a} * f_1^2 * e_0 * e_1 + f_2 * \Delta) = b' \equiv \bar{b} \underline{\text{ mod }} \bar{q}.&
\end{aligned}$$

What we need is $\|b'\|_\infty < \bar{q}/2$, so that $b' = \bar{b}$. Using Lemma 4.5 and setting

$$\|\bar{a}\|_\infty = \|r_0\|_\infty = \|r_1\|_\infty = \|g\|_\infty = \|f_1\|_\infty = \|f_2\|_\infty = \|e_0\|_\infty = \|e_1\|_\infty = B,$$

we have:

$$\begin{aligned}
\|b'\|_\infty &\leq \frac{1}{q'}(p^2 n^4 B^5 + 2pn^4 B^5 + n^4 B^5 + nB\|\Delta\|_\infty) \\
&\leq \frac{1}{q'}(4p^2 n^4 B^5 + nB\frac{pq'}{2}). \qquad (5)
\end{aligned}$$

Assuming it is possible to set the parameters B, q' and $\bar{q}$ so that $\|b'\|_\infty < \bar{q}/2$ and thus $b' = \bar{b}$, it follows that ComposedEvalMult indeed is correct. For a satisfying choice of such parameters, see Subsection 5.7. □

5.6 Bootstrapping

There is, however, a limit to how many times the ComposedEvalMult procedure may be performed, as q consists of a finite (and, for security reasons, relatively low) number of factors, and hence there are only so many modulus switchings that may be performed. When this limit is reached, or if the modulus switching does not sufficiently reduce the noise of the ciphertext, bootstrapping is performed.

The bootstrapping itself is essentially the procedure presented in Section 3 adapted to fit the scheme. It is worth repeating that the bootstrapping procedure requires $m \in \mathbb{Z}_p$, even though the scheme itself supports messages in R_p.

The procedure applied here is somewhat simplified compared to the earlier general presentation due to the structure of the scheme. In particular: converting the plaintext modulus from p to q as well as lifting both moduli to q' are no-ops.

The fact that the schemes supports key switching ensures that the ciphertext outputted by the trace function is decryptable, by the argument in Section 3. For further details on the bootstrapping procedure adapted to this scheme, see Section 3.6 of [21].

5.7 Security and Selection of Parameters

In order to discuss the security of the given encryption scheme and possible attacks it will be helpful to first discuss how the scheme is related to lattices, which might not be obvious at first glance. Following the description in Section 4, the following matrix defines the lattice $\mathcal{L}_{\text{NTRU}}$, where the row vectors are taken to be the basis vectors of the lattice, and the public key h is written as $h = h_0 + h_1x + \dots h_{n-1}x^{n-1}$:

$$\mathbf{B}_{\text{NTRU}} = \begin{bmatrix} 1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{n-1} \\ 0 & 1 & \dots & 0 & -h_{n-1} & h_0 & \dots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & -h_1 & -h_2 & \dots & h_0 \\ 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{bmatrix}.$$

This $2n \times 2n$ matrix may be viewed as the following 2×2 matrix instead: $\mathbf{B}_{\text{NTRU}} = \left[\begin{array}{c|c} \mathbf{I} & \mathbf{H} \\ \hline 0 & q\mathbf{I} \end{array}\right]$ where $\mathbf{H}$ is the matrix consisting of the cyclical permutations of the coefficients of h and $\mathbf{I}$ is the $n \times n$ identity matrix. Recall that $h = g * f^{-1}$ and $f * f^{-1} = 1 + qf'$, so we must have $f * h = g + qu$ for some polynomial $u = g * f'$.

Proposition 5.6. *For any two polynomials $a = \sum_{i=0}^{n-1} a_i x^i$, $b = \sum_{i=0}^{n-1} b_i x^i$, let $[a, b]$ denote the vector $[a_0, \dots, a_{n-1}, b_0, \dots, b_{n-1}]$. For the polynomials f, g and u described above, we have: $[f, -u] \cdot \mathbf{B}_{\text{NTRU}} = [f, g]$, meaning the vector $[f, g]$ belongs to to the lattice $\mathcal{L}_{\text{NTRU}}$.*

Proof. It is fairly obvious that the n first coefficients of the resulting vector of $[f, -u] \cdot \mathbf{B}_{\text{NTRU}}$ is f, as the upper left $n \times n$ corner of $\mathbf{B}_{\text{NTRU}}$ is the identity matrix and the lower $n \times n$ left corner is zero. The following $n+1+k$ coefficients, for $k \in \{0, 1, \dots n-1\}$ are expressed as:

$$\sum_{\substack{i,j=0 \\ i+j=k}}^{n-1} f_i h_j - \sum_{\substack{i,j=0 \\ i+j=k+n}}^{n-1} f_i h_j - qu_k = g_k + qu_k - qu_k = g_k.$$

Hence, $[f, -u] \cdot \mathbf{B}_{\text{NTRU}} = [f, g]$, meaning $[f, g]$ belongs to $\mathcal{L}_{\text{NTRU}}$, as the vector may be expressed as a linear combination of the basis vectors of $\mathcal{L}_{\text{NTRU}}$ using only integers. □

Proposition 5.7. *With overwhelming probability, the vector $[f, g]$ is one of the shortest vectors in the lattice $\mathcal{L}_{\text{NTRU}}$.*

Proof. Recall the result of Theorem 4.3, namely that the length of the shortest vector in any lattice $\mathcal{L}$ is at most $\sqrt{\eta}\det(\mathcal{L})^{1/\eta}$, with η being the dimension of the lattice. In the case of $\mathcal{L}_{\text{NTRU}}$, this bound is

$$\|\mathbf{v}_0\| \leq \sqrt{2n}(q^n)^{1/2n} = \sqrt{2nq}.$$

We may in addition calculate a bound on $\|[f, g]\|$, using the upper bound $\|f\|_\infty, \|g\|_\infty < \sqrt{\frac{q}{4pn}}$, derived in the proof of Proposition 5.1:

$$\begin{aligned} \|[f, g]\| &= \sqrt{f_0^2 + f_1^2 + \dots f_{n-1}^2 + g_0^2 + g_1^2 + \dots + g_{n-1}^2} \\ &\leq \sqrt{2n\Big(\sqrt{\frac{q}{4pn}}\Big)^2} = \sqrt{\frac{q}{2p}}. \end{aligned}$$

Comparing the two bounds, we have:

$$\sqrt{\frac{q}{2p}} \Bigg/ \sqrt{2nq} = \sqrt{\frac{1}{4pn}} \ll 1.$$

Thus, seeing as the bound on $\|[f, g]\|$ is substantially smaller than the Hermite bound, it is overwhelmingly probable that $[f, g]$ is one of the vectors of shortest length in $\mathcal{L}_{\text{NTRU}}$. □

It follows that if an adversary is able to solve SVP in $\mathcal{L}_{\text{NTRU}}$, she is able to compute f based solely on information made public by the scheme, and thus break the scheme. It should also be mentioned that the adversary does not need to find precisely $[f, g]$, as any permutation of these polynomials will suffice to perform a decryption. Furthermore, any pair of polynomials $[f', g']$ with sufficiently small coefficients satisfying the relation $f' * h = g' \ \underline{\text{mod}}\ q$ will also suffice, as will probably any solution to approximate-SVP for an approximation factor smaller than $\sqrt{n}$ [15]. Thus, recovering the secret key f of the encryption scheme reduces to solving approximate-SVP for the lattice $\mathcal{L}_{\text{NTRU}}$.

Such a solution is not apparent given the basis matrix $\mathbf{B}_{\text{NTRU}}$, and the most efficient way of solving either SVP, approximate-SVP or HSVP is to find a basis which is easier to work with, for instance one which contains the solution of either stated problem. This is known as basis reduction, and whilst there are several algorithms that achieve this, the main one is LLL and a generalization of it, BZK. Both of these algorithms are HSVP-algorithms, and achieve Hermite root factors $\delta = (1 + \epsilon)$ for some $\epsilon > 0$ [9]; for an explanation of HSVP or δ, see Section 4. LLL works by swapping two vectors in the basis and performing a reduction, whereas BKZ works similarly, only with more than two vectors. The number of vectors BKZ works with is known as the block size, denoted by β; the larger β is, the more exact the output of the algorithm will be.

LLL runs in polynomial time and outputs a basis with a vector with norm less than $(4/3)^{(\eta-1)/4}\det(\mathcal{L})^{1/\eta}$, where η is the dimension of the lattice [9]. Obviously, this bound is not very impressive as η grows large, in which case BKZ may be better suited. BKZ also outputs a reduced basis, containing a vector of length at most $\sqrt{\gamma_\beta}^{1+(\eta-1)/(\beta-1)}\det(\mathcal{L})^{1/\eta}$ [9], where γ_β is the Hermite constant mentioned in Section 4. The downside of this improved bound is that BKZ does not run in polynomial time, but exponential in β [15], and furthermore: there exists no good upper bound for its running time complexity.

Seeing as the time complexity of the algorithms best suited for attacking the system is not clearly understood, estimating the security of the scheme is not as straight forward as choosing a security parameter λ so that a successful attack takes roughly time 2^λ and choosing n, q and p based on λ such that 2^λ seems infeasible. Furthermore, both LLL and BKZ tend to perform better in practice than any theoretical bound suggests. Therefore, instead of relying on theory, the security of the scheme is determined experimentally by running BKZ on lattices of relatively low dimensions and block sizes until a short vector is found and then extrapolating the run-time [15, 7]. It is also possible to construct a simulation of BKZ to estimate the precision of the algorithm, that is: the expected Hermite root factor δ of the shortest vector in the resulting reduced basis [7].

The way to construct a secure system is then to choose parameters n, q and p and estimate the level of security these produce by computing an estimate of δ which would be needed to break the system using BKZ. One then runs simulations of BKZ for the given set of parameters to estimate how long it would take BKZ to reduce the given basis with sufficient precision. The length of the shortest vector in $\mathcal{L}_{\text{NTRU}}$ which may be used for decryption may often be expressed as a function of $q : \psi(q)$. Meanwhile, BKZ and LLL both find vectors of length expressible as $\delta^{2n}\det(\mathcal{L}_{\text{NTRU}})^{2n} = \delta^{2n}\sqrt{q}$, where n is the dimension of the ring R. Reducing the given basis with sufficient precision therefore requires obtaining a Hermite root factor δ so that $\delta^{2n}\sqrt{q} \leq \psi(q)$.

This is why performing a ring reduction lowers the security of the scheme and a modulus switching makes the scheme more secure. Suppose that for a certain set of parameters a vector of length $\delta^{2n}\sqrt{q}$ which may be used to decrypt a given ciphertext c may be found by either algorithm in reasonable time. Suppose then that ring reduction is applied to c, resulting in a new ciphertext c' in the ring of dimension $n' \leq n/2$. Due to the new lattice dimension $2n'$, a larger Hermite root factor $\delta' > \delta$ suffices to approximate a vector such that $(\delta')^{2n'}\sqrt{q} < \psi(q)$, meaning both LLL and BKZ will compute this approximation in less time, and the scheme is therefore less secure than before. Suppose, instead, that a modulus switching is performed on c, resulting in a new ciphertext modulus $\bar{q} < q$. Now, a smaller Hermite root factor $\delta'' < \delta$ is required to find a suitable vector, as we require $(\delta'')^{2n}\sqrt{\bar{q}} \leq \psi(\bar{q}) < \psi(q)$, meaning the scheme is now more secure than before the modulus switching was performed.

However, security is not the only important notion when it comes to choosing parameters, there are several other aspects that need to be taken into account:

- The moduli $q_1, \dots, q_t$ should be sufficiently large to enable sufficient noise reduction via modulus switching.
- The ring dimension n is not too large; it should not be overly timeconsuming and/or memory-intensive to manipulate the ciphertexts.
- The plaintext modulus p and any noise added to the ciphertext during encryption should be sufficiently small as to allow evaluation of reasonably sized circuits with correct decryption.

With regards to the last bullet point, "noise added to the ciphertext during encryption" refers, in essence, to the polynomials f, g, r and e in the encryption scheme, all of which should be short[2], $f = 1 \underline{\text{mod}}\ p$ and invertible modulo q, and $e = m \underline{\text{mod}}\ p$ as well. There are several ways these polynomials may be chosen, the naive one being from discrete Gaussians for g, r, f' and e', and setting $f = pf' + 1, e = pe' + m$. The downside of this is that it will cause f to have mean 1 and e to have mean m, however it does simplify the dependence of the various parameters.

Suppose any of the polynomials affecting the noise level are drawn from a discrete Gaussian distribution of parameter r, and set w as an assurance measure, meaning it will be highly unlikely for a normally distributed polynomial to have length more than $\sqrt{2\pi}w$ times the standard deviation $r/\sqrt{2\pi}$. As such it should be practically impossible for any polynomial drawn from this distribution to have Euclidean length greater than rw. It follows that we may set a bound on the infinity norm of any such distributed polynomial as $\frac{rw}{\sqrt{n}}$, where n is the degree of the polynomial.

Using this bound and expression (5) in Subsection 5.5, we need

$$\frac{1}{q'}(4p^2n^4(\frac{rw}{\sqrt{n}})^5 + n\frac{rw}{\sqrt{n}}\frac{pq'}{2}) = \frac{1}{q'}(4p^2n^{1.5}r^5w^5 + \frac{1}{2}pq'\sqrt{n}rw) < q/2q'$$

for decryption to still be correct after a call to ComposedEvalMult. Suppose $4p^2n^{1.5}r^5w^5 < q'$, we then have

$$1 + \frac{1}{2}p\sqrt{n}rw < q/2q', \tag{6}$$

where the value $q/q' \geq q_1$, a single factor of q, and thus the smallest possible ciphertext modulus. The authors of [21] suggest setting $q_1 > 4prw\sqrt{n}$ as the theoretical lower bound, which certainly satisfies bound (6). However, if we in addition require q_1 to be sufficiently large as to guarantee correct decryption of a freshly generated ciphertext assuming $q = q_1$, we must also have $rw > \sqrt{n}$ in order to satisfy

$$4pn(\frac{rw}{\sqrt{n}})^2 = 4pr^2w^2 < q_1, \tag{7}$$

[2]Meaning the infinity norm of either polynomial should satisfy certain bounds to guarantee correct decryption.

according to expression (1). Alternatively, we could set a somewhat higher lower bound for q_1, such as $4pr^2w^2\sqrt{n}$, which satisfies both (6) and (7), or do as the authors of [21] (motivated by implementation simplicity), and simply set a universal bound for any factor of q, namely

$$q_i > 4p^2r^5w^5n^{1.5} \tag{8}$$

for any $i \in \{1, 2, \ldots, t\}$.

As a quick aside: this bound clearly satisfies the first part of bound (3), so performing KeySwitch on a ciphertext of degree 1 will not result in unmanageable noise. It might seem to contradict the second part, though, which would require any ciphertext modulus to be greater than $8p^2r^5w^5n^{1.5}$. However, this would only be the case if one were to perform a KeySwitch on a ciphertext of degree 2 without subsequently performing a modulus switching, which by the construction of ComposedEvalMult cannot happen.

We may therefore conclude that ensuring any factor of q satisfies bound (8) results in an encryption scheme which reduces the noise sufficiently to guarantee correct decryption of any freshly generated ciphertext and output of ComposedEvalMult, given that the input ciphertexts has at most the same noise level as any freshly generated ciphertexts for the current ciphertext modulus $\bar{q}$. If this is not the case, bootstrapping may be performed.

However, ensuring that the factors of q provide sufficient noise reduction, and thus correct decryption, is not the only aspect one has to take into account during the selection of q. One must also consider the depth of computation, $D = t - 1$: how many times modulus switching may be performed. The more such operations are required, the more factors of q is needed, and the less secure the scheme is, unless the dimension of the ring is increased, which would make computation more time-consuming. What is more, we have the following lower bound on n, dependent on both q and δ:

$$n \geq \log(q)/(4\log(\delta))$$

to prevent the lattice $\mathcal{L}_{\text{NTRU}}$ from being too sparse to contain sufficiently short vectors [20]. Thus, D (or rather, t) should be kept as low as practically possible to avoid a cumbersome ring and minimize the amount of calls to the time consuming bootstrapping procedure.

Of more importance is the effect the size of q and its polynomial dependence on n has on the security of the scheme, which is quite severe: unlike the original NTRU encryption scheme, or provably secure variants of it, q will not be of the same order of magnitude as n [16, 22]. This is a problem, as it turns out that this may be exploited in an attack on the lattice $\mathcal{L}_{\text{NTRU}}$ [1]. The attack maps the given instance to a lattice of smaller dimension, which is less labour intensive to perform a basis reduction in, and finally the solution obtained in the smaller lattice is lifted to the original lattice.

The attack utilizes the algebraic structure of the ring in a similar fashion as the bootstrapping does, namely the isomorphism of $R = \mathbb{Z}[x]/(x^n + 1)$ and

$\mathbb{Z}[\omega]$, which is the ring of integers of the field $\mathbb{Q}[\omega] = \mathbb{K}$, for ω some root of unity of order $2n$. Let $\mathbb{L}$ be a subfield of $\mathbb{K}$ such that $\mathbb{L} = \mathbb{Q}[\omega']$ for some root of unity ω' of order $2n'$, $\mathbb{L}$ will then have a ring of integers $R' = \mathbb{Z}[x]/(x^{n'} + 1)$. Just as described in Section 3, there are automorphisms φ_i on $\mathbb{K}$ fixing $\mathbb{L}$ pointwise, corresponding to $\varphi_i(a(x)) = a(x^i) \quad \forall i \in \mathbb{Z}_{2n}^* \wedge i \equiv 1 \mod 2n'$ in the rings of integers [3]. Given these, we may define the norm function, $\mathrm{N}_{\mathbb{K}/\mathbb{L}} : \mathbb{K} \to \mathbb{L}$, as

$$\mathrm{N}_{\mathbb{K}/\mathbb{L}}(a) = \prod_i \varphi_i(a).$$

Given f, g and h for a given implementation of the encryption scheme, we define $f' = \mathrm{N}_{\mathbb{K}/\mathbb{L}}(f), g' = \mathrm{N}_{\mathbb{K}/\mathbb{L}}(g), h' = \mathrm{N}_{\mathbb{K}/\mathbb{L}}(h)$ and let $r = n/n'$. The attack follows from the observation that $[f', g']$ is a vector in the lattice $\mathcal{L}'_{\mathrm{NTRU}}$ spanned by the basis $\mathbf{B}'_{\mathrm{NTRU}} = \left[\begin{array}{c|c} \mathbf{I} & \mathbf{H}' \\ \hline 0 & q\mathbf{I} \end{array}\right]$ and the following heuristic:

Heuristic 5.8 (Heuristic 1 [1])**.** *For any n, $f, g \in R$ with reasonable isotropic distribution of variance σ^2 and any constant $c > 0$, there exists a constant C such that $\|f'\| \leq (\sigma n^C)^r$ and $\|g'\| \leq (\sigma n^C)^r$, except with probability $\mathcal{O}(n^{-c})$.*

The authors of [1] note that the (spherical) discrete Gaussian distributions used for drawing secrets and errors are isotropic, and thus the heuristic applies to the scheme presented here.

Of course this is also the case for any version of the NTRU scheme, the reason this attack is effective against this particular scheme is the following: Theorem 1 of [1] assures us of the existence of a lattice reduction algorithm with block-size β able to find a vector $[x', y'] \in R'$ such that

$$\|[x', y']\| \leq \beta^{\Theta(2n'/\beta)} \|\mathbf{v}_0\| \leq \beta^{\Theta(n/\beta r)} \|[f', g']\| \leq \beta^{\Theta(n/\beta r)} (n\sigma)^{\Theta(r)},$$

when applied to the basis $\mathbf{B}'_{\mathrm{NTRU}}$, where $\|\mathbf{v}_0\|$ is the length of the shortest vector in $\mathcal{L}'_{\mathrm{NTRU}}$.

Theorem 5.9 (Theorem 2 [1])**.** *Let $f', g' \in R'$ be such that $\langle f' \rangle$ and $\langle g' \rangle$ are ideals such that $\langle f' \rangle + \langle g' \rangle = R'$ (i.e. coprime ideals) and $h' * f' = g'$ $\underline{\text{mod}}$ q for some $h' \in R'$. If $[x', y'] \in \mathcal{L}'_{\mathrm{NTRU}}$ has length satisfying*

$$\|[x', y']\| < \frac{q}{\|[f', g']\|},$$

then $[x', y'] = v[f', g']$ for some $v \in R'$.

The authors of [1] note that the probability of $\langle f' \rangle$ and $\langle g' \rangle$ being coprime is roughly 3/4, and furthermore that it does not seem strictly necessary in practice for the attack to be successful.

[3]The requirement $i \equiv 1 \mod 2n'$ is not included in Section 3 as in the case described there $n' = 1$, and so this automatically holds for any $i \in \mathbb{Z}_{2n}^*$.

Thus, if $\beta^{\Theta(n/\beta r)}(n\sigma)^{\Theta(r)} \leq q$, this bound may be satisfied. This is obviously more likely as q grows larger in relation to n, as is the case for our encryption scheme: the more ComposedEvalMult operations the scheme allows to be performed, the larger q grows, roughly by a factor of $n^{1.5}$.

The final step of the attack is to lift the vectors $x', y' \in R'$ to R. This is simply done by using the canonical inclusion map $L : \mathbb{L} \to \mathbb{K}$:

$$\begin{aligned}
&x = L(x') = L(v) * L(f'),\\
&y = L(y') * h/L(h') \underline{\text{mod}}\, q = L(v) * L(g') * h/L(h') \underline{\text{mod}}\, q,
\end{aligned}$$

where v is as in Theorem 5.9. For simplicity, we set $\tilde{f} = L(f')/f, \tilde{g} = L(g')/g$ and $\tilde{h} = L(h')/h$; we then have

$$\begin{aligned}
x &= L(v) * \tilde{f} * f \ \underline{\text{mod}}\, q\\
y &= L(v) * L(g')/\tilde{h} = L(v) * g * \tilde{g}/\tilde{h} = L(v) * \tilde{f} * g \ \underline{\text{mod}}\, q\\
\Rightarrow [x, y] &= u * [f, g] \in \mathcal{L}_{\text{NTRU}} \quad \text{with } u = L(v) * \tilde{f} \in R.
\end{aligned}$$

In other words: the subfield attack yields a (small) multiplicative of $[f, g]$ under certain reasonable assumptions. The attack is most efficient for cases in which q is super-polynomial in n, as opposed to the encryption scheme at hand, in which case the growth is merely polynomial.

However, as the experimental reports of [1] show, the attack would still be effective. As an example: a successful attack was carried out in 3.5 hours for $n = 2^{11}$ when $\log(q) \geq 165$. Setting $q = (4p^2r^5w^5n^{1.5})^{D+1}$, we may conclude that this attack would be successful for any computational depth greater than 3 in the encryption scheme presented here, assuming $p = 2, r = w = 6$, which are the parameter values the authors of [21] suggest. In a full field, this corresponds to running BKZ with block size 27 to achieve $\delta = 1.0141$. The highest dimension the attack was carried out in was $n = 2^{12}$, with success for $\log(q)$ as low as 190, yet again corresponding to a computational depth of just 3, with the same parameter values as previously. This attack took 120 hours; an attack of the full lattice would require running BKZ with block size 131 to achieve $\delta = 1.0081$, an attack that seems unfeasible at this point in time. Note also that the subfield attacks used LLL, it therefore seems reasonable to expect better attacks if, for example, BKZ was used instead.

Based on these findings, the scheme must be considered insecure except for very large values of n combined with low computational depth at this point in time. This is far from an ideal property, as it either results in a scheme only capable of evaluating very shallow circuits, or one overly dependent on the costly bootstrapping procedure, making it very inefficient. In addition, there are no guarantees of this being the most effective attack, and further progress in the field might lead to a full breaking of the scheme. It is, however, important to note that this does not affect the security of the original NTRU scheme, as its choice of q is typically much too small for the attack to be successful.

6 Leveled FHE without Bootstrapping

The scheme presented in this section is a leveled fully homomorphic encryption scheme, meaning it is able to evaluate any circuit as long as its depth is specified, alongside the security parameter, before an instance of the scheme is created. To achieve this, it uses the procedures described in Subsections 6.2 and 6.3 in combination with the basic encryption scheme in Subsection 6.1. The correctness of the homomorphic encryption scheme is proved in Subsection 6.5 and the parameters required for the scheme to be secure is discussed in Subsection 6.6, where it is also proven that the scheme satisfies the IND-CPA security notion. Finally, the theoretical and practical efficiency of the scheme is presented in Subsection 6.7.

6.1 The Basic Encryption Scheme

The following encryption scheme is based on RLWE presented in section 4, and we therefore require the following procedure:

DrawRLWE($q, n, N, \chi, s \in R_q$): Generate the row vector $\mathbf{a} \leftarrow R_q^N$ uniformly at random as well as the vector $\mathbf{e} \leftarrow \chi^N$, then let $\mathbf{b}^T = \mathbf{a}^T s + 2\mathbf{e}^T$. Finally set $\mathbf{A}$ as the $N \times 2$ matrix consisting of $\mathbf{b}^T$ followed by the column vector $-\mathbf{a}^T$. Output: $\mathbf{A}$.

Note that this is simply drawing an instance of the set $\{[a_i * s + e_i, a_i]\}$ used for the RLWE problems presented in Section 4, just expressed as a matrix. The reason we introduce this as a separate procedure and not just part of the key generation is because some of the subroutines of the leveled fully homomorphic encryption scheme presented later requires this particular procedure as well.

We now present the basic encryption scheme, upon which the leveled fully homomorphic encryption scheme will be based:

ESetup(1^λ): Choose a modulus $q = q(\lambda)$ and choose the other parameters n and $N = \text{polylog}(q)$ as well as the noise distribution χ to ensure 2^λ security against known attacks. Output $params = \{q, n, N, \chi\}$.

EKeyGen($params$): Draw $s \leftarrow \chi$, run DrawRLWE($params, s$) $= \mathbf{A}$ and set $\mathbf{s} = [1, s]$. Output: $sk = \mathbf{s}$, $pk = \mathbf{A}$.

EEnc($params, pk, m$): To encrypt the message $m \in R_2$, set $\mathbf{m} = [m, 0] \in R_2^2$, draw $\mathbf{r} \leftarrow R_2^N$ and output the ciphertext

$$\mathbf{c} = \mathbf{m} + \mathbf{r} \cdot \mathbf{A} \mod q.$$

EDec($params, sk = \mathbf{s}, \mathbf{c}$): Output

$$m = (\langle \mathbf{c}, \mathbf{s} \rangle \underline{\text{mod}}\, q) \mod 2.$$

Correctness of this basic encryption scheme is easily observed, assuming the noise is kept manageable:

$$\begin{aligned}
\mathrm{EEnc}(m) &= \mathbf{c} = \mathbf{m} + \mathbf{r}\cdot\mathbf{A} \\
&= \left[m + \sum_{i=1}^{N}(a_i * s + 2e_i) * r_i,\ -\sum_{i=1}^{N} a_i * r_i\right] \underline{\mathrm{mod}}\, q \\
\Rightarrow \mathrm{EDec}(\mathbf{c}) &= (\langle \mathbf{c}, \mathbf{s}\rangle \ \underline{\mathrm{mod}}\, q) \mod 2 \\
&= (m + 2\sum_{i=1}^{N} e_i * r_i \ \underline{\mathrm{mod}}\, q) \mod 2 = m.
\end{aligned}$$

6.2 Key Switching

Just as with the key switching procedure in the previous scheme, the idea is to take a ciphertext $\mathbf{c}_1$ encrypted under a secret key $\mathbf{s}_1$ and convert it into a ciphertext $\mathbf{c}_2$ encrypted under a different secret key $\mathbf{s}_2$ such that $\mathrm{EDec}(params, \mathbf{s}_1, \mathbf{c}_1) = \mathrm{EDec}(params, \mathbf{s}_2, \mathbf{c}_2)$. The main reason for wanting to change the keys in this scheme however, is to reduce the Euclidean length of the components of the secret key, which will be shown to grow exponentially during multiplication and be of great importance to the success of modulus switching in particular. The procedure itself is dependent on the two following subroutines:

BitDecomp($a \in R_q, q$): $a = \sum_{i=0}^{n-1} a_i x^i$ with $a_i \in [0, \ldots, q-1]$, a is decomposed by decomposing all the coefficients into their bit representations: $a = \sum_{i=0}^{\lfloor \log q \rfloor} 2^i u_i$ for $u_i \in R_2\ \forall i$. Output: $[u_0, u_1, \ldots, u_{\lfloor \log q \rfloor}] \in R_2^{\lfloor \log q \rfloor + 1}$.

Powersof2($a \in R_q, q$): Output: $[a, 2a, \ldots, 2^{\lfloor \log q \rfloor} a] \in R_q^{\lfloor \log q \rfloor + 1}$.

Note that we have the following for any ring elements c, s:

$$\begin{aligned}
\langle \mathrm{BitDecomp}(c, q), \mathrm{Powersof2}(s, q)\rangle &= \sum_{i=0}^{\lfloor \log q \rfloor} \langle u_i, 2^i s\rangle = \sum_{i=0}^{\lfloor \log q \rfloor} \langle 2^i u_i, s\rangle \\
&= \left\langle \sum_{i=0}^{\lfloor \log q \rfloor} 2^i u_i, s \right\rangle = \langle c, s\rangle.
\end{aligned}$$

Here, $\langle c, s\rangle$ denotes the regular polynomial multiplication obtained by viewing c and s as vectors of length 1.

Both the operations BitDecomp and Powersof2 are applicable to any vector $\mathbf{v} \in R_q^k$ by obvious generalisations, which in both cases will result in output of dimension $k(\lfloor \log(q) \rfloor + 1)$. The derived result therefore also applies to any vectors $\mathbf{c}, \mathbf{s}$ of the same length k.

Switching from key $\mathbf{s}_1$ to $\mathbf{s}_2$ requires hint, which is calculated as follows:

SwitchKeyGen($\mathbf{s}_1 \in R_q^{k_1}, \mathbf{s}_2 \in R_q^2$): Here $\mathbf{s}_1 = [1, \underline{\mathbf{s}}_1]$, $\underline{\mathbf{s}}_1 \leftarrow \chi^{k_1 - 1}$ for $k_1 > 1$ and $\mathbf{s}_2 = [1, s_2], s_2 \leftarrow \chi$. Set $N = k_1(\lfloor \log(q) \rfloor + 1)$ and generate the matrix

$\mathbf{A} = \text{DrawRLWE}(q, n, N, \chi, s_2)$. Set $a_{1\to 2} = \mathbf{A} + [\text{Powersof2}(\mathbf{s}_1)^T, 0^T]$, i.e. $\mathbf{A}$ with $\text{Powersof2}(\mathbf{s}_1)^T$ added to its first column. Output $a_{1\to 2}$.

Once this hint has been generated, the actual key switching procedure is:

KeySwitch($a_{1\to 2}$, $\mathbf{c}_1$): Output: $\mathbf{c}_2 = \text{BitDecomp}(\mathbf{c}_1) \cdot a_{1\to 2}$.

Proposition 6.1 (Lemma 4.2 [6])**.** *Let $\mathbf{s}_1, \mathbf{s}_2$ and $a_{1\to 2}$ be as in SwitchKeyGen. Then*

$$\langle \mathbf{c}_2, \mathbf{s}_2 \rangle = 2\langle \text{BitDecomp}(\mathbf{c}_1), \mathbf{e}\rangle + \langle \mathbf{c}_1, \mathbf{s}_1 \rangle$$

Proof.

$$\begin{aligned}
\langle \mathbf{c}_2, \mathbf{s}_2 \rangle &= \text{BitDecomp}(\mathbf{c}_1) \cdot a_{1\to 2} \cdot \mathbf{s}_2^T \\
&= \text{BitDecomp}(\mathbf{c}_1) \cdot [\mathbf{a}^T s_2 + 2\mathbf{e}^T + \text{Powersof2}(\mathbf{s}_1)^T, -\mathbf{a}^T] \cdot \mathbf{s}_2^T \\
&= \text{BitDecomp}(\mathbf{c}_1) \cdot [2\mathbf{e}^T + \text{Powersof2}(\mathbf{s}_1)^T] \\
&= 2\langle \text{BitDecomp}(\mathbf{c}_1), \mathbf{e}\rangle + \langle \text{BitDecomp}(\mathbf{c}_1), \text{Powersof2}(\mathbf{s}_1)\rangle \\
&= 2\langle \text{BitDecomp}(\mathbf{c}_1), \mathbf{e}\rangle + \langle \mathbf{c}_1, \mathbf{s}_1 \rangle.
\end{aligned}$$

□

Note that $\langle \text{BitDecomp}(\mathbf{c}_1), \mathbf{e}\rangle$ is fairly small, as $\text{BitDecomp}(\mathbf{c}_1) \in R_2^{k(\lfloor \log(q) \rfloor + 1)}$ for $\mathbf{c}_1 \in R_q^k$, but it still increases the noise of the new ciphertext $\mathbf{c}_2$ slightly when compared to $\mathbf{c}_1$.

An important property of the hints and public keys published is: any adversary A given a pair of matrices $(\mathbf{A}, a_{1\to 2})$ she knows is either a public key matrix and a key switching matrix of the stated encryption scheme, or two matrices generated completely at random is unable to guess which distribution her sample is drawn from with any significant advantage. More formally:

Lemma 6.2 (Lemma 4.2 [6])**.** *For every valid secret key $\mathbf{s}_1 \in R_q^{k_1}$, $\mathbf{s}_2 = [1, s_2]$ such that $s_2 \leftarrow \chi$ and any polynomial-time adversary A described above, there exists a negligible function ϵ such that:*

$$\begin{aligned}
\text{Adv}(A) = |\Pr[&\mathbf{A}_0 \leftarrow \text{DrawRLWE}(q, n, N, \chi, s_2);\ a_{1\to 2_0} \leftarrow \text{SwitchKeyGen}(\mathbf{s}_1, \mathbf{s}_2); \\
&\mathbf{A}_1 \leftarrow R_q^{N\times 2};\ a_{1\to 2_1} \leftarrow R_q^{N\times 2}:\ A(\mathbf{A}_0, a_{1\leftarrow 2_0}) = 0] \\
-\Pr[&\mathbf{A}_0 \leftarrow \text{DrawRLWE}(q, n, N, \chi, s_2);\ a_{1\to 2_0} \leftarrow \text{SwitchKeyGen}(\mathbf{s}_1, \mathbf{s}_2); \\
&\mathbf{A}_1 \leftarrow R_q^{N\times 2};\ a_{1\to 2_1} \leftarrow R_q^{N\times 2}:\ A(\mathbf{A}_0, a_{1\leftarrow 2_0}) = 1]| \le \epsilon.
\end{aligned}$$

That is: the distributions $(\mathbf{A}_0, a_{1\leftarrow 2_0}), (\mathbf{A}_1, a_{1\leftarrow 2_1})$ are computationally indistinguishable, even for a distinguisher who knows $\mathbf{s}_1$.

Proof. This follows from the assumption that the decision problem for RLWE is hard in the ring R_q, that is: it is impossible to distinguish between the matrices $\mathbf{A}_0$ and $\mathbf{A}_1$, as one is generated according to RLWE, and the other is generated uniformly at random. Assume for the sake of contradiction that there does exist an adversary A able to distinguish between the pairs $(\mathbf{A}_0, a_{1\to 2_0})$ and $(\mathbf{A}_1, a_{1\to 2_1})$. We may then construct an adversary B able to solve the decision problem for RLWE in the following way: B is presented with a challenge matrix $\mathbf{A}$ of dimension $N \times 2$ for some N. B may then choose any vector $\mathbf{s}_1$ of length $k_1 = N/(\lfloor \log(q) \rfloor + 1)$ where the first coordinate is 1, compute Powersof2$(\mathbf{s}_1)$ and add its transpose to $\mathbf{A}$'s first column. This results in the matrix $a_{1\to 2}$, which will be a valid key switching matrix from $\mathbf{s}_1$ to $\mathbf{s}_2$, assuming $\mathbf{A}$ is a public key generated by $\mathbf{s}_2$. Finally B may send $(\mathbf{A}, a_{1\to 2})$ as well as $\mathbf{s}_1$ to A. If A outputs 0, B guesses that $\mathbf{A}$ has been generated according to RLWE, if A outputs 1, B guesses $\mathbf{A}$ has been generated at random. It is easy to see that B is able to solve the decision problem of RLWE with the same advantage of A, and it must thus follow, under the assumption that this problem is hard, that the advantage of A must be negligible. □

6.3 Modulus Switching

Seeing as switching keys increases the noise, we need something that will reduce it as well, which we get by switching the ciphertext modulus of the system. Suppose $\mathbf{c}$ is an encryption of a message m under a short secret key $\mathbf{s}$. Then, for $q' < q$ and $\mathbf{c}'$ the ciphertext closest to $(q'/q)\mathbf{c}$ in the Euclidean norm when both are viewed as coefficient vectors which also satisfies $\mathbf{c}' \equiv \mathbf{c} \mod 2$, $\mathbf{c}'$ is a valid encryption of the same m under the same secret key $\mathbf{s}$; at least under some specifications.

Definition 6.3 (Definition 4.4 [6]). *For an integer vector $\mathbf{v}$ and the integers $2 < q' < q$, define the integer vector $\mathbf{v}' = \text{Scale}(\mathbf{v}, q, q', 2)$ to be the R-vector closest to $(q'/q)\mathbf{v}$ in the Euclidean norm such that $\mathbf{v}' \equiv \mathbf{v} \mod 2$.*

Definition 6.4 (Definition 4.5 [6]). *The R-norm of a vector $\mathbf{v}$ in R^k is defined as $\ell_1^{(R)}(\mathbf{v}) = \sum_{i=1}^{k} \|v_i\|$.*

The procedure of switching moduli in the encryption scheme is simply setting the new ciphertext as $\mathbf{c}' = \text{Scale}(\mathbf{c}, q, q', 2)$, where $\mathbf{c}$ is the original ciphertext, q is the original ciphertext modulus and q' is the new modulus.

Proposition 6.5 (Lemma 4.6 [6]). *Let n be the degree of the ring R and let q and q' be positive integers satisfying: $2 < q' < q$ and $q' \equiv q \equiv 1 \mod 2$. Let $\mathbf{c} \in R^2$ and $\mathbf{c}' = \text{Scale}(\mathbf{c}, q, q', 2)$. Then, for any $\mathbf{s} \in R^2$ such that*

$$\|\langle \mathbf{c}, \mathbf{s} \rangle \underline{\text{mod}}\, q\| < q/2 - (q/q')n\ell_1^{(R)}(\mathbf{s}),$$

we have

$$\langle \mathbf{c}', \mathbf{s} \rangle \underline{\text{mod}}\, q' = \langle \mathbf{c}, \mathbf{s} \rangle \underline{\text{mod}}\, q \qquad \text{and}$$
$$\|\langle \mathbf{c}', \mathbf{s} \rangle \underline{\text{mod}}\, q'\| < (q'/q)\|\langle \mathbf{c}, \mathbf{s} \rangle \underline{\text{mod}}\, q\| + n\ell_1^{(R)}(\mathbf{s}).$$

Proof. For some $l \in R$ we have

$$\langle \mathbf{c}, \mathbf{s} \rangle \underline{\text{mod}}\, q = \langle \mathbf{c}, \mathbf{s} \rangle - ql.$$

For the same l, let

$$e_{q'} = \langle \mathbf{c}', \mathbf{s} \rangle - q'l \in R.$$

Note that $e_{q'} \equiv \langle \mathbf{c}', \mathbf{s} \rangle \underline{\bmod}\, q'$, with equality if $\|e_{q'}\| < q'/2$. This is what must be proved.

$$\begin{aligned}
\|e_{q'}\| &= \| -q'l + \langle (q'/q)\mathbf{c}, \mathbf{s} \rangle + \langle \mathbf{c}' - (q'/q)\mathbf{c}, \mathbf{s} \rangle \| \\
&\leq \| -q'l + \langle (q'/q)\mathbf{c}, \mathbf{s} \rangle \| + \| \langle \mathbf{c}' - (q'/q)\mathbf{c}, \mathbf{s} \rangle \| \\
&\leq (q'/q)\| \langle \mathbf{c}, \mathbf{s} \rangle \underline{\bmod}\, q \| + \sqrt{n} \sum_{i=1}^{2} \|c_i' - (q'/q)c_i\| \|s_i\| \\
&\leq (q'/q)\| \langle \mathbf{c}, \mathbf{s} \rangle \underline{\bmod}\, q \| + \sqrt{n}\sqrt{n}\,\ell_1^{(R)}(\mathbf{s}) \\
&< (q'/q)(q/2 - (q/q')n\ell_1^{(R)}(\mathbf{s})) + n\ell_1^{(R)}(\mathbf{s}) = q'/2.
\end{aligned}$$

□

This, hopefully, explains the necessity of switching keys: if the length of the key $\mathbf{s}$ is too long, the inequality $\|\langle \mathbf{c}, \mathbf{s} \rangle\| < q/2 - (q/q')n\ell_1^{(R)}(\mathbf{s})$ will not hold, and thus the ciphertext $\mathbf{c}'$ might not decrypt correctly.

Furthermore, assuming the inequality does hold, we have an upper bound on the noise of the new ciphertext $\mathbf{c}'$, namely:

$$\|\langle \mathbf{c}', \mathbf{s} \rangle \underline{\bmod}\, q'\| < (q'/q)\|\langle \mathbf{c}, \mathbf{s} \rangle \underline{\bmod}\, q\| + n\ell_1^{(R)}(\mathbf{s}).$$

Thus, assuming $\ell_1^{(R)}(\mathbf{s})$ is small in relation to q, switching the ciphertext modulus from q to q' reduces the noise of the ciphertext. It is worth noting that this reduction may be performed without explicit knowledge of the secret key $\mathbf{s}$ apart from a bound on its R-norm $\ell_1^{(R)}(\mathbf{s})$.

6.4 The Homomorphic Encryption Scheme

Based on the basic encryption scheme as well as key switching and modulus switching, we may form an encryption scheme which is fully homomorphic without having to turn to bootstrapping, assuming the scheme allows for sufficiently many modulus switchings to be performed.

FHESetup$(1^\lambda, 1^L)$: Let $\mu = \mu(\lambda, L) = \theta(\log(\lambda) + \log(L))$ be a parameter related to the bit-length of the the modulo q. Just as in the algorithm ESetup we generate a set of parameters $params$, only now we generate a ladder of moduli q_i and hence also of sets of parameters. Set $params_L = \{q_L = q_L(\lambda, (L+1)\mu), n = n(\lambda, (L+1)\mu), N_L = \text{polylog}(q_L), \chi = \chi(\lambda, (L+1)\mu)\}$. For $i = L-1, \ldots, 0$ set $params_i = \{q_i = q_i(\lambda, (i+1)\mu), n_i = n, N_i = \text{polylog}(q_i), \chi_i = \chi\}$. Thus we have a ladder of decreasing moduli, from q_L with $(L+1)\mu$ bits to q_0 with μ bits.
Output $params = \{params_0, params_1, \ldots, params_L\}$

FHEKeyGen($params$): For $i = L, \dots, 1, 0$, do the following:

1. Draw $s_i \leftarrow \chi$ and set $\mathbf{s}_i = [1, s_i]$.
   Generate $\mathbf{A}_i \leftarrow \text{DrawRLWE}(params_i, s_i)$.
2. Set $\mathbf{s}'_i = [1, s_i, s_i^2] \in R_{q_i}^3$.
3. Set $a_{\mathbf{s}'_{i+1} \to \mathbf{s}_i} = \text{SwitchKeyGen}(\mathbf{s}'_{i+1}, \mathbf{s}_i)$. This step is omitted when $i = L$.

The secret key sk consists of the $\mathbf{s}_i$'s and the public key pk consists of the matrices $\mathbf{A}_i$ and $a_{\mathbf{s}'_{i+1} \to \mathbf{s}_i}$. Output sk, pk.

FHEEncrypt($params, pk, m$): Run EEnc($params_L, \mathbf{A}_L, m$).

FHEDec($params_i, sk, \mathbf{c}$): Run EDec($params_i, \mathbf{s}_i, \mathbf{c}$), supposing $\mathbf{c}$ is encrypted under key $\mathbf{s}_i$. A ciphertext may easily be modified to carry an index indicating which level it belongs to.

FHEAdd($pk, \mathbf{c}_1, \mathbf{c}_2$): The two ciphertexts have to be encrypted under the same secret key $\mathbf{s}_i$, use FHERefresh if needed. Set $\mathbf{c}_3 = \mathbf{c}_1 + \mathbf{c}_2 \mod q_i$. $\mathbf{c}_3$ may be interpreted as a ciphertext under the secret key $\mathbf{s}'_i$. [4] Output:

$$\mathbf{c}_4 = \text{FHERefresh}(\mathbf{c}_3, a_{\mathbf{s}'_i \to \mathbf{s}_{i-1}}, q_i, q_{i-1}).$$

FHEMult($pk, \mathbf{c}_1, \mathbf{c}_2$): The two ciphertexts have to be encrypted under the same secret key $\mathbf{s}_i$, use FHERefresh if needed. Set

$$\mathbf{c}_3 = [c_{1_1} * c_{2_1}, c_{1_1} * c_{2_2} + c_{1_2} * c_{2_1}, c_{1_2} * c_{2_2}] \mod q_i,$$

where $\mathbf{c}_k = [c_{k_1}, c_{k_2}]$ for $k = 1, 2$. $\mathbf{c}_3$ is encrypted under the secret key $\mathbf{s}'_i$. Ouput:

$$\mathbf{c}_4 = \text{FHERefresh}(\mathbf{c}_3, a_{\mathbf{s}'_i \to \mathbf{s}_{i-1}}, q_i, q_{i-1}).$$

FHERefresh($\mathbf{c}, a_{\mathbf{s}'_i \to \mathbf{s}_{i-1}}, q_i, q_{i-1}$): Do the following:

1. Set $\mathbf{c}_1 = \text{KeySwitch}(a_{\mathbf{s}'_i \to \mathbf{s}_{i-1}}, \mathbf{c}, q_i)$, a ciphertext encrypted under the key $\mathbf{s}_{i-1}$ for modulus q_i.
2. Set $\mathbf{c}_2 = \text{Scale}(\mathbf{c}_1, q_i, q_{i-1}, 2)$, a ciphertext encrypted under the key $\mathbf{s}_{i-1}$ for modulus q_{i-1}. Ouput $\mathbf{c}_2$.

6.5 Correctness

This entire subsection will prove the following:

Proposition 6.6 (Theorem 5.6 [6]). *The encryption scheme in Subsection 6.4 decrypts correctly, assuming the procedure FHERefresh is performed at most L times.*

[4]Seeing as $\mathbf{c}_3 = [c_{3_1}, c_{3_2}] \in R_q^2$ wheras $\mathbf{s}'_i = [1, s_i, s_i^2] \in R_q^3$, it is strictly speaking necessary to set $\mathbf{c}_3 = [c_{3_1}, c_{3_2}, 0]$ for the inner-product $\langle \mathbf{c}_3, \mathbf{s}'_i \rangle$ to be mathematically valid.

To avoid having to state the assumption several times, $\langle \mathbf{c}, \mathbf{s} \rangle \underline{\bmod}\ q$ will throughout this subsection denote the modular reduction of the ciphertext so that it may be expressed as $m + 2\tilde{e}$ for some error polynomial $\tilde{e}$. For example, in the case of fresh ciphertexts, $\langle \mathbf{c}, \mathbf{s}_L \rangle \underline{\bmod}\ q_L = m + 2\sum_{i=1}^{N_L} r_i * e_i$. This is the reduced ciphertext which will result in correct decryption, and in general we only have $\langle \mathbf{c}, \mathbf{s} \rangle \equiv m + 2\sum r * e \ \underline{\bmod}\ q$. Correctness of the scheme essentially boils down to showing that

$$\|\langle \mathbf{c}, \mathbf{s} \rangle \ \underline{\bmod}\, q_i\| < q_i/2$$

for ciphertexts outputted by the scheme, either by FHEEncrypt, in which case $q = q_L$ or by FHERefresh, in which case $q \in \{q_0, \ldots, q_{L-1}\}$.

FHEEncrypt: The first step is to show the correct decryption of a fresh ciphertext $\mathbf{c} = \text{FHEEnc}(params_L, pk, m)$, that is: $\|\langle \mathbf{c}, \mathbf{s}_L \rangle \ \underline{\bmod}\ q_L\| < q_L/2$. Recall that $\langle \mathbf{c}, \mathbf{s} \rangle \ \underline{\bmod}\ q_L = m + 2\sum_{i=1}^{N} e_i * r_i$ where $m \in R_2,\ r_i \leftarrow R_2,\ e_i \leftarrow \chi$, as shown in Subsection 6.1. Seeing as χ is a noise distribution, we let B_χ be a bound such that with overwhelming probability $\|e\| < B_\chi$ for any $e \leftarrow \chi$. We may then, using Lemma 4.4, bound the noise of a freshly generated ciphertext as:

$$\begin{aligned} \|\langle \mathbf{c}, \mathbf{s}_L \rangle \ \underline{\bmod}\, q_L\| &= \|m + 2\sum_{i=1}^{N_L} r_i * e_i\| \leq \|m\| + 2\sqrt{n}\sum_{i=1}^{N_L} \|r_i\|\|e_i\| \\ &\leq \sqrt{n} + 2nN_L B_\chi = B_L. \end{aligned} \tag{9}$$

Assuming $B_L < q_L/2$, we have:

$$\text{FHEDec}(params_L, sk, \text{FHEEncrypt}(params_L, pk, m)) = m.$$

FHEAdd: The correctness of the procedure per se depends on FHERefresh, however, this requires that the ciphertext it receives as input will decrypt correctly. Hence, we need $\mathbf{c}_3 = \mathbf{c}_1 + \mathbf{c}_2 \mod q_i$ to satisfy the inequality

$$\|\langle \mathbf{c}_3, \mathbf{s}'_i \rangle \ \underline{\bmod}\, q_i\| < q_i/2.$$

Let $\langle \mathbf{c}_k, \mathbf{s}_i \rangle \ \underline{\bmod}\, q_i = m_k + 2\sum_{j=1}^{N_i} e_{k_j} * r_{k_j}$, so that $(\langle \mathbf{c}_k, \mathbf{s}_i \rangle \ \underline{\bmod}\, q_i) \mod 2 = m_k$ for $k = 1, 2$. We then have:

$$\begin{aligned} \|\langle \mathbf{c}_3, \mathbf{s}'_i \rangle \ \underline{\bmod}\, q_i\| &= \|m_1 + m_2 + 2\sum_{j=1}^{N_i}(e_{1_j} * r_{1_j} + e_{2_j} * r_{2_j})\| \\ &\leq \|m_1\| + \|m_2\| + 2\sum_{j=1}^{N_i}(\|e_{1_j} * r_{1_j}\| + \|e_{2_j} * r_{2_j}\|) \\ &\leq 2\sqrt{n} + 2\sqrt{n}\sum_{j=1}^{N_i}(\|e_{1_j}\|\|r_{1_j}\| + \|e_{2_j}\|\|r_{2_j}\|) \end{aligned}$$

$$\leq 2\sqrt{n} + 4nN_iB_\chi = 2B_i. \tag{10}$$

FHEMult: Just as with the FHEAdd procedure, we need to establish that the ciphertext $\mathbf{c}_3$, which is used to calculate the output ciphertext of FHEMult using FHERefresh, will decrypt correctly, meaning $\|\langle \mathbf{c}_3, \mathbf{s}'_i\rangle \underline{\text{mod}}\ q_i\| < q_i/2$ for all $i \in \{0, 1, \dots L\}$. Recall that $\mathbf{c}_3 = [c_{1_1} * c_{2_1}, c_{1_1} * c_{2_2} + c_{1_2} * c_{2_1}, c_{1_2} * c_{2_2}]$ and $\mathbf{s}'_i = [1, s_i, s_i^2]$ for

$$\mathbf{c}_k = [c_{k_1}, c_{k_2}] = [m_k + \sum_{j=1}^{N_i} a_{k_j} * r_{k_j} * s_i + 2e_{k_j} * r_{k_j},\ -\sum_{j=1}^{N_i} a_{k_j} * r_{k_j}],$$

and assume $(\langle \mathbf{c}_k, \mathbf{s}_j\rangle \underline{\text{mod}}\ q_i) \mod 2 = m_k$ for $k = 1, 2$. We thus have the following bound on the noise of $\mathbf{c}_3$:

$$\begin{aligned}
\|\langle \mathbf{c}_3, \mathbf{s}'_i\rangle \underline{\text{mod}}\, q_i\| &= \|c_{1_1} * c_{2_1} + (c_{1_1} * c_{2_2} + c_{1_2} * c_{2_1}) * s_i + c_{1_2} * c_{2_2} * s_i^2 \ \underline{\text{mod}}\, q_i\| \\
&= \|m_1 * m_2 + 2m_1 * \sum_{j=1}^{N_i} e_{2_j} * r_{2_j} + 2m_2 * \sum_{j=1}^{N_i} e_{1_j} * r_{1_j} \\
&+ \Big(2\sum_{j=1}^{N_i} e_{1_j} * r_{1_j}\Big)\Big(2\sum_{j=1}^{N_i} e_{2_j} * r_{2_j}\Big)\| \\
&\leq \|m_1 * m_2\| + \|2m_1 * \sum_{j=1}^{N_i} e_{2_j} * r_{2_j}\| + \|2m_2 * \sum_{j=1}^{N_i} e_{1_j} * r_{1_j}\| \\
&+ \|4\Big(\sum_{j=1}^{N_i} e_{1_j} * r_{1_j}\Big)\Big(\sum_{j=1}^{N_i} e_{2_j} * r_{2_j}\Big)\| \\
&\leq \sqrt{n}\Big(\|m_1\|\|m_2\| + 2\|m_1\| \sum_{j=1}^{N_i} \|e_{2_j} * r_{2_j}\| \\
&+ 2\|m_2\| \sum_{j=1}^{N_i} \|e_{1_j} * r_{1_j}\| + 4\sum_{j=1}^{N_i} \|e_{1_j} * r_{1_j}\| \sum_{j=1}^{N_i} \|e_{2_j} * r_{2_j}\|\Big) \\
&\leq \sqrt{n}\Big(n + 2n\sum_{j=1}^{N_i} (\|e_{1_j}\|\|r_{1_j}\| + \|e_{2_j}\|\|r_{2_j}\|) \\
&+ 4n\sum_{j=1}^{N_i} \|e_{1_j}\|\|r_{1_j}\| \sum_{j=1}^{N_i} \|e_{2_j}\|\|r_{2_j}\|\Big) \\
&\leq \sqrt{n}(n + 4n^{3/2}N_iB_\chi + 4n^2N_i^2B_\chi^2) = \sqrt{n}B_i^2.
\end{aligned} \tag{11}$$

FHERefresh: The first step of this procedure is to expand the ciphertext by setting $\mathbf{c}_1 = \text{KeySwitch}(a_{\mathbf{s}'_i \to \mathbf{s}_{i-1}}, \mathbf{c})$. Seeing as $\mathbf{c} \in R_q^3$ after either FHEAdd or FHEMult, it must be the case that $\text{BitDecomp}(\mathbf{c}, q_i) \in R_2^{3\lfloor \log(q_i)\rfloor + 3}$. The following bound on the noise follows from the proof of correctness of the key switching procedure, Proposition 6.1, and the assumption that $2B_i \leq \sqrt{n}B_i^2$:

$$\begin{aligned}\|\langle \mathbf{c}_1, \mathbf{s}_{i-1}\rangle \ \underline{\mathrm{mod}}\ q_i\| &\leq \|\langle \mathbf{c}, \mathbf{s}_i'\rangle \ \underline{\mathrm{mod}}\ q_i\| + \|2\langle \mathrm{BitDecomp}(\mathbf{c}, q_i), \mathbf{e}\rangle\| \\ &\leq \|\langle \mathbf{c}, \mathbf{s}_i'\rangle \ \underline{\mathrm{mod}}\ q_i\| + 2\sqrt{n}\|\mathrm{BitDecomp}(\mathbf{c}, q_i)\|\|\mathbf{e}\| \\ &\leq \sqrt{n}B_i^2 + 6nB_\chi(\lfloor \log(q_i)\rfloor + 1).\end{aligned}$$

The final step is switching the modulus of the ciphertext by setting

$$\mathbf{c}_2 = \mathrm{Scale}(\mathbf{c}_1, q_i, q_{i-1}, 2).$$

Recall that the secret key $\mathbf{s}_{i-1} = [1, s_{i-1}]$, where $B_\chi \geq s_{i-1} \leftarrow \chi$. Based on this and the proof of correctness for modulus switching, Proposition 6.5, we get the following bound of the ciphertext:

$$\begin{aligned}\|\langle \mathbf{c}_2, \mathbf{s}_{i-1}\rangle \ \underline{\mathrm{mod}}\ q_{i-1}\| &\leq \frac{q_{i-1}}{q_i}\|\langle \mathbf{c}_1, \mathbf{s}_{i-1}\rangle \ \underline{\mathrm{mod}}\ q_i\| + n\ell_1^{(R)}(\mathbf{s}_{i-1}) \\ &\leq \frac{q_{i-1}}{q_i}\|\langle \mathbf{c}_1, \mathbf{s}_{i-1}\rangle \ \underline{\mathrm{mod}}\ q_i\| + 2nB_\chi.\end{aligned}$$

Based on these two bounds, we get this final bound of an output ciphertext from FHERefresh:

$$\begin{aligned}\|\langle \mathbf{c}_2, \mathbf{s}_{i-1}\rangle \ \underline{\mathrm{mod}}\ q_{i-1}\| &\leq \frac{q_{i-1}}{q_i}(\sqrt{n}B_i^2 + 6nB_\chi(\lfloor \log(q_i)\rfloor + 1)) + 2nB_\chi \qquad (12) \\ &= \frac{q_{i-1}}{q_i}(\sqrt{n}B_i^2 + \nu_{\mathrm{KeySwitch}_i}) + \nu_{\mathrm{ModRed}_i}.\end{aligned}$$

In order to guarantee correct decryption, (12) must be strictly less than $q_{i-1}/2$. The last part of the proof of correctness is thus showing, among other things, that this bound holds for all values of i.

The strategy is to find a universal bound B on the length of noise, showing that any fresh ciphertext and any ciphertext outputted by FHERefresh will have length less than this bound, and finally showing that any ciphertext satisfying this bound will decrypt correctly for any ciphertext modulus q_i.

For this to be the case, we need $B < q_L/2$ to ensure correct decryption of a fresh ciphertext, according to (9), and also $2B < q_i/2$, $\sqrt{n}B^2 < q_i/2$ $\forall i \in \{1, 2, \ldots L\}$ to prevent excessive noise before FHERefresh, as by (10) and (11), respectively. Finally, the value of (12) needs to be less than B for all values of $1 \leq i \leq L$ and $B < q_i/2$ to ensure proper decryption of a processed ciphertext, for any possible value of i.

Setting $B = \tilde{\mathcal{O}}(B_\chi nL)$ and $q_i = (2\sqrt{n}B)^{i+1}$ ensures this. It is easy to see that $B < q_L/2 = 2^L(\sqrt{n}B)^{L+1}$, meaning any fresh ciphertext will decrypt correctly. Regarding (10) and (11), setting $i = 1$ results in:

$$q_1/2 = (2B\sqrt{n})^2/2 = 2nB^2 > \sqrt{n}B^2 > 2B.$$

It follows that (10) and (11) are strictly less than $q_i/2$, $1 \leq i \leq L$. Furthermore, we have the following properties for the same values of i:

1. $q_i/q_{i-1} = 2\sqrt{n}B$.
2. The noise added during the FHERefresh procedure due to the key switching and modulus switching,

$$\frac{q_{i-1}}{q_i} 6nB_\chi(\lfloor\log(q_i\rfloor) + 1) + 2nB_\chi = \frac{q_{i-1}}{q_i}\nu_{\text{KeySwitch}_i} + \nu_{\text{ModSwitch}_i},$$

is dominated by the last term, and it thus follows, in particular, that $2(\frac{q_{i-1}}{q_i}\nu_{\text{KeySwitch}_i} + \nu_{\text{ModRed}_i}) \leq B$.

Based on these two properties, we may compute:

$$\frac{q_{i-1}}{q_i}(\sqrt{n}B^2 + \nu_{\text{KeySwitch}_i}) + \nu_{\text{ModRed}_i} \leq \frac{1}{2\sqrt{n}B}\sqrt{n}B^2 + \frac{1}{2}B \leq B.$$

Meaning: any ciphertext outputted by FHERefresh will have noise less than B if the input ciphertext has noise at most $\sqrt{n}B^2$.

Finally: $B < \sqrt{n}B = q_0/2 < q_1/2 < \cdots < q_L/2$, and it thus follows that any ciphertext outputted by FHERefresh with a valid ciphertext as input will decrypt correctly. Correctness of the leveled fully homomorphic encryption scheme as such follows.

6.6 Selection of Parameters and Security

Setting $B = \tilde{\mathcal{O}}(B_\chi nL)$ and $q_i = (2B\sqrt{n})^{i+1}$ provides correctness of the scheme, but this is not the only aspect to be taken into account when choosing parameters, as the desired security level in particular affects the choice of n. The authors of [6] state that to achieve hardness against 2^λ time adversaries, it is required setting $n = \Omega(\log(q_L/B)\lambda)$. It follows that n may be expressed as a polynomial of λ, and thus that $B\sqrt{n}$ also may be expressed as the product of a polynomial of λ and a term $\tilde{\mathcal{O}}(L)$. Based on this, we have that $\log(q_i) = i\theta(\log(\lambda)+\log(L)) = i\mu$. Thus, μ is a measure of how many bits is needed to represent a given modulus q to ensure a correct decryption, even after a call to the FHERefresh procedure. Furthermore, we may set $n \approx \mu\lambda L$ to achieve 2^λ security.

Intuitively, it is easy to se that the security of the scheme is based on RLWE, presented in Section 4; after all, if an adversary given a public key matrix $\mathbf{A} = [\mathbf{a}^T s + 2\mathbf{e},\ -\mathbf{a}^T]$ has a significant chance of finding the secret key $\mathbf{s} = [1, s]$, that is, solving the search problem of RLWE, the scheme is obviously insecure. Furthermore, the public key of the homomorphic scheme is a collection of both encryption matrices and key switching matrices, and according to Lemma 6.2 any pair of public key matrices $\mathbf{A}_i$ and $a_{\mathbf{s}'_{i+1}\to\mathbf{s}_i}$ will be indistinguishable from two randomly generated matrices, under the assumption that the decision problem of RLWE is hard. In addition, by Miccianco's regularity lemma, the vector $\mathbf{r}\cdot\mathbf{A}$ has negligible statistical distance from uniform[5] when both $\mathbf{A} \in R_q^{N\times 2}$ and $\mathbf{r} \in R_q^N$ for $N = \text{polylog}(q)$ [18, 6]. It follows that, under the assumption that

[5]I.e. it may for all intents and purposes be regarded as sampled from a uniform distribution.

the decision problem of RLWE is hard, and thus that any public key matrix $\mathbf{A}$ is indistinguishable from uniform for an adversary, an adversary has negligible advantage in guessing the plaintext m, as the encryption procedure outputs ciphertexts that are statistically independent of this plaintext.

How difficult are these two problems, though? It is easy to see that the search problem is at least as difficult as the decision problem, and under certain conditions they may be shown to be equal, as mentioned in Section 4. However, no general reduction is known, but the conjecture is that the decision problem is hard. The search problem is provably at least as hard as solving approximate-SVP for any ideal lattice, as proven by [17]. The conjecture is that this is in fact hard also, though there is strictly speaking no proof of this conjecture. It is proven that approximate-SVP is NP-hard for lattices in general, to within any constant factor less than $\sqrt{2}$ [19], but first of: the approximation factors sufficient to solve RLWE may be greater than $\sqrt{2}$, and second: the extra algebraic structure of ideal lattices might make approximate-SVP easier to solve than in general lattices, even though such an algorithm has not been found yet, despite considerable effort [17]. Hence, the conjecture is believed to be reasonable, making the RLWE problems hard.

Assuming the hardness of these problems, it is possible to prove the scheme to be secure under the notion of indistinguishability under a chosen plaintext attack (IND-CPA). Informally, IND-CPA means: an adversary A with access to the encryption of any plaintext of her choosing is unable to distinguish between the encryption of two randomly chosen messages m_0 and m_1 with significant advantage.

This is fairly weak as far as security notions go, but this is an inherent weakness in fully homomorphic encryption schemes in general. The reason for this is that stronger security notions, such as IND-CCA (being unable to distinguish between two randomly selected messages given access to both an encryption and decryption oracle) may be proven to imply the ciphertexts of the scheme being non-malleable (NM): given a ciphertext c, an adversary is unable to output a ciphertext c' such that the underlying messages m and m' are related according to some relation of the adversary's own choosing [4]. This security goal obviously cannot hold for any homomorphic encrytpion scheme, as an adversary may ask for the encryption of any message m', add the resulting ciphertext to the challenge ciphertext and thus present an encryption of $m + m'$.

In order to prove the scheme to be IND-CPA secure, we need a more formal definition of the security notion:

Definition 6.7 (Definition 2.3 [6]). *An encryption scheme $E=\{\mathcal{K},\mathcal{E},\mathcal{D}\}$ with key generation algorithm, encryption algorithm and decryption algorithm $\mathcal{K},\mathcal{E},\mathcal{D}$ respectively is IND-CPA secure if there is a negligible function ϵ for every polynomial-time adversary $A = (\mathcal{A},\mathcal{B})$ such that*

$$\begin{aligned}\mathrm{Adv}(A) = |&\Pr[pk \leftarrow \mathcal{K}(\lambda);\ (m_0, m_1) \leftarrow \mathcal{A}(pk);\ c_0 \leftarrow \mathcal{E}(pk, m_0) :\ \mathcal{B}(pk, c_0) = 1] \\ &- \Pr[pk \leftarrow \mathcal{K}(\lambda);\ (m_0, m_1) \leftarrow \mathcal{A}(pk);\ c_1 \leftarrow \mathcal{E}(pk, m_1) :\ \mathcal{B}(pk, c_1) = 1]| \\ &\le \epsilon(\lambda).\end{aligned}$$

We may now prove the following:

Theorem 6.8 (Theorem 5.7 [6])**.** *Let $n = n(\lambda), q = q(\lambda)$ and $\chi = \chi(\lambda)$ be functions of the security parameter, and let $N = \text{polylog}(q, \lambda)$. Then the scheme FHE described in Subsection 6.4 is IND-CPA secure, under the assumption that the RLWE decision problem is hard for parameters n, q and χ.*

The proof is based on a hybrid argument, meaning we start with the original scheme and the advantage an IND-CPA adversary has against it, and gradually change the scheme in steps (the scheme of each step being a hybrid of the original) until we find one which is provably IND-CPA secure. Expressing the adversary's advantage as a sum of the advantages in the hybrid will provide proof that the original scheme is itself secure.

Proof. Throughout the proof, A is an IND-CPA adversary for the FHE scheme in Subsection 6.4, and $\text{Adv}_H(A)$ denotes the probability of success of A in hybrid H

- Hybrid H_0: This is an IND-CPA game in the original scheme: A gets a public key pk distributed according the distribution set by λ. Recall that this key consists of the following:
  - A matrix $\mathbf{A}_L = [\mathbf{a}_L^T s_L + 2\mathbf{e}_L^T,\ -\mathbf{a}_L^T]$.
  - L pairs $(\mathbf{A}_l, a_{\mathbf{s}'_{l+1}\to\mathbf{s}_l})$ for $l = L-1$ to 0, where $\mathbf{A}_l = [\mathbf{a}_l^T s_l + 2\mathbf{e}_l^T,\ -\mathbf{a}_l^T]$ and $a_{\mathbf{s}'_{l+1}\to\mathbf{s}_l} = \text{SwitchKeyGen}([1, s_{l+1}, s_{l+1}^2], [1, s_l])$.

  Assume for the sake on contradiction that A has a non-negligible advantage in winning the IND-CPA game, meaning: for every negligible function ϵ, it is the case that:

  $$\begin{aligned}\text{Adv}_{H_0}(A) = |&\Pr[A(pk, \text{FHEEnc}(pk, m_0)) = 1] \\ &- \Pr[A(pk, \text{FHEEnc}(pk, m_1)) = 1]| > \epsilon(\lambda),\end{aligned} \tag{13}$$

  where m_0, m_1 are drawn at random from R_2.

- Hybrid H_l for $l = 1, \ldots L-1$: Hybrid H_l is identical to H_{l-1} except for the public key $\mathbf{A}_{l-1}$ and the hint $a_{\mathbf{s}'_l\to\mathbf{s}_{l-1}}$ are drawn uniformly at random from the appropriate domains. The claim is that the two hybrids H_l and H_{l-1} are computationally indistinguishable, that is: no polynomial adversary will be able to distinguish between the hybrids H_l and H_{l-1}. Assume for the sake of contradiction that there does exist an adversary A' able to distinguish between the two hybrids with non-negligible advantage. We may then construct an adversary B able to distinguish between a correctly generated pair $(\mathbf{A}_l, a_{\mathbf{s}'_{l+1}\to\mathbf{s}_l})$ and a pair of matrices of the same dimension generated uniformly at random. This distinguisher B is constructed as follows:

  B is presented with a challenge consisting of two matrices, and should be able to distinguish between a pair of matrices belonging to some public key pk of FHE, $(\mathbf{A}_0, a_{1\to 2_0})$, and a pair of randomly generated matrices, $(\mathbf{A}_1, a_{1\to 2_1})$. If B is going to employ the advantage of A' in order to win this challenge, the problem needs to be transformed into one A' can solve. Thus, B sets her own challenge input to be $(\mathbf{A}_{l-1}, a_{\mathbf{s}'_l\to\mathbf{s}_{l-1}})$, generates the pair of matrices of index lower than $l-1$ at random and the matrix pairs of index l or higher, as well as $\mathbf{A}_L$, are generated as $\mathbf{A}_i = \text{DrawRLWE}(s_i)$, $a_{\mathbf{s}'_i\to\mathbf{s}_{i-1}} = \text{SwitchKeyGen}(\mathbf{s}'_i, \mathbf{s}_{i-1})$ for $s_i \leftarrow \chi$ $\mathbf{s}_{i-1} = [1, s_{i-1}]$ and $\mathbf{s}'_i = [1, s_i, s_i^2]$. The distinguisher B then feeds the L pairs of matrices and the matrix $\mathbf{A}_L$ to the adversary A'. If A' outputs H_{l-1}, B guesses that her own pair of matrices are generated according to RLWE and therefore outputs 0, and outputs 1 if A' guesses H_l.

Based on this construction, B will be able to distinguish between the two cases $(\mathbf{A}_0, a_{1\to 2_0})$ and $(\mathbf{A}_1, a_{1\to 2_1})$ with the same non-negligible advantage as A' has of distinguishing two hybrids H_{l-1} and H_l, contradicting Lemma 6.2. Thus, the two hybrids are computationally indistinguishable, and it must follow that

$$|\text{Adv}_{H_l}(A) - \text{Adv}_{H_{l-1}}(A)| \leq \epsilon(\lambda)$$

for some negligible function ϵ. If this is not the case, meaning there is an adversary A significantly better at winning the IND-CPA game in one of the hybrids H_l or H_{l-1}, A' may easily employ A to distinguish between the two hybrids. Finally: note that in hybrid H_L all the pairs $(\mathbf{A}_{l-1}, a_{\mathbf{s}'_l\to\mathbf{s}_{l-1}})$ are uniformly random for every $l \in \{1, 2, \dots L\}$.

- Hybrid H_{L+1}: The final hybrid is H_{L+1} and is equal to H_L except that in H_{L+1} the matrix $\mathbf{A}_L$ is uniformly distributed as well, rather than as a public key of the basic encryption scheme in Subsection 6.1. By an argument essentially equal to the one above, it must be the case that

$$|\text{Adv}_{H_{L+1}}(A) - \text{Adv}_{H_L}(A)| < \epsilon(\lambda).$$

Thus, in the final hybrid, all the elements of the public key are uniformly random and completely independent of the message. Then, according to Micciancio's regularity lemma, any ciphertext generated by either public key matrix $\mathbf{A}_i$ is statistically independent from the message it encrypts [6, 18]. It must therefore follow that

$$\text{Adv}_{H_{L+1}} < \epsilon(\lambda),$$

that is, the scheme in hybrid H_L is IND-CPA secure. Combining all these findings, we have:

$$\text{Adv}_{H_0}(A) \leq \text{Adv}_{H_{L+1}}(A) + \sum_{l=0}^{L} |\text{Adv}_{H_l} - \text{Adv}_{H_{l+1}}| \leq \epsilon(\lambda),$$

for some negligible function ϵ, and thus a contradiction to (13). It follows that the leveled fully homomorphic encryption scheme of Subsection 6.4 is IND-CPA secure. □

6.7 Efficiency

The main measure of efficiency in this scheme is the per-gate computation: the ratio between the time it takes to evaluate a circuit (i.e. a function) homomorphically and the time it takes to evaluate it in the clear. Asymptotically this per-gate computation is $\tilde{\mathcal{O}}(\lambda L^3)$, which follows from an asymptotic analysis of FHEMult in particular, as it is the most time consuming operation.

First, the ciphertext $\mathbf{c}_3 = [c_{1_1} * c_{2_1}, c_{1_1} * c_{2_2} + c_{1_2} * c_{2_1}, c_{1_2} * c_{2_2}]$ must be computed, which requires $\tilde{\mathcal{O}}(n \log(q_i))$. Next, as part of FHERefresh, the keys are switched, which involves a multiplication of the transpose of BitDecomp$(\mathbf{c}_3, q_i)$ with a key switching matrix. The length of the vector depends on $\log(q_i)$, and so it follows that the computation required for this step is $\tilde{\mathcal{O}}(n \log(q_i)^2)$. The final step is switching the modulus, the computation of which is $\tilde{\mathcal{O}}(n \log(q_i))$. It follows that the computational cost of the entire operation is $\tilde{\mathcal{O}}(n \log(q_i)^2)$, where $\log(q_i) \leq \log(q_L)$, which depends quasilinearly on L, and n depends quasilinearly on λL, so the per-gate computation of $\tilde{\mathcal{O}}(\lambda L^3)$ immediately follows.

This scheme can however be made more efficient, i.e. have a lower per-gate computation, with several optimizations, one of which is bootstrapping, which brings it down to $\tilde{\mathcal{O}}(\lambda^2)$. This might seem slightly odd, seeing as scheme is advertised as a fully homomorphic encryption scheme *without* bootstrapping. However, note that this is an optimization, as opposed to a necessity. For shallow circuits that require a small L, it might in fact be better to not implement bootstrapping at all, as it is a somewhat costly operation: in [13], it took 320 seconds to bootstrap vectors of 1024 elements from GF(2^{16}) with a security level of 76, and required 3.4 GB of space.

There have also been implementations of the encryption scheme without bootstrapping, one of which homomorphically evaluated the AES-128 circuit, which consists of ten application of one keyed rounding function, with different keys. Three different implementations are presented in [10], varying in the representation of the ciphertext and how packed it is. Several optimizations of the scheme presented in Subsection 6.4 are added, among others not refreshing the ciphertext after each addition or multiplication. Instead, each ciphertext has an estimate of its noise level attached to it, and FHERefresh is only performed when this estimate gets too high. For more details, see [10]. It should be noted that for all of the implementations, the evaluation time for each AES-round decreased as more round were performed, and that the implementation based on the presented encryption scheme took the least amount of time. The first round took 7 hours to evaluate, whereas the last one required "merely" 30 minutes, and 36 hours to evaluate the entire AES circuit. This implementation was run on one core of a supercomputer with 256 GB RAM.

7 Comparison

Throughout this section, the schemes presented in Section 5 and 6 will be referred to as RC-NTRU and BGV, respectively.

At a high level, the two schemes are quite similar: they both work over the ring $R = \mathbb{Z}[x]/(x^{2^k} + 1)$ and support encryption of messages in R_p, for some modulus p, though BGV is only presented for the case $p = 2$, and the bootstrapping procedure of RC-NTRU requires that the encrypted message lies in $\mathbb{Z}_2$. They both use modulus switching to manage the noise, support ring reduction [12] and employ key switching to aid other operations, which increases the noise somewhat in both schemes. In the case of RC-NTRU, key switching is used during bootstrapping and ring reduction, whilst BGV switches keys during FHERefresh and ring reduction. The main motivation to reduce the dimension of the polynomial ring in either scheme is to allow for easier computation. Care must however be taken as to when this reduction is performed, as it results in a less secure scheme in both cases [12].

They both encrypt messages using polynomials, and in both cases, the polynomials employed are subject to certain bounds, which again subject the other parameters of the schemes to bounds. These bounds are necessary due to another key similarity: botch schemes use the same approach as the system in Subsection 2.2: enveloping the message in two layers of noise, where the inner layer must not be allowed to interfere with the outer layer, as this may result in incorrect decryption.

The question of correct or incorrect decryption boils down to the same central question in both schemes: whether or not the modular reduction of a polynomial dependent on both the ciphertext and secret key equals a particular polynomial v, which is equivalent to the plaintext modulo some modulus. These requirements result in the various bounds derived for the schemes, where the bounds for RC-NTRU are derived with respect to the infinity norm of any polynomial affecting the noise, whilst BGV uses Euclidean norm. The bounds of RC-NTRU might therefore be viewed as stricter than those of BGV, as FHEDec$(\mathbf{c}, \mathbf{s})$ might still be correct even though $\|m + 2\sum_{j=1}^{N_i} r_j * e_j\| > q_i/2$. Using Euclidean norms during the derivation of bounds on B, and therefore also q, might thus result in slightly larger parameters than strictly necessary.

Both schemes require sets of parameters in some sense to enable modulus switching. BGV generates these sets directly in FHESetup, whilst this set is the possible ciphertext moduli in the case of RC-NTRU. The fact that they both use modulus switching results in another common factor: they may both be calibrated to allow for a certain amount for modulus switchings to be performed, referred to as both depth of computation D in RC-NTRU and level L in BGV[6]. The larger this number is, the larger the initial ciphertext modulus q has to be, as q is reduced during a modulus switching. Furthermore, the ciphertext modulus depends on both the desired level of security as well as the depth of

[6]Strictly speaking, the number of modulus switchings which may be performed in BGV is $L - 1$, meaning L corresponds to t in RC-NTRU: the number of factors in q.

computation in both schemes. In addition, the dimension of the ring depends on the logarithm of q and the chosen security level. Increasing the computational depth therefore often comes at the cost of increasing the dimension of the ring to ensure the desired level of security in both schemes.

A central difference is how this security level is estimated in the two schemes: whereas BGV has a more traditional approach in choosing a parameter λ such that any attack on the scheme takes more than time 2^λ, the security level of RC-NTRU is estimated through experimental approximations. This is because RC-NTRU is based on a very particular problem: how hard is it to approximate a certain vector in a specific lattice. Seeing as the time complexity of the best attack is not clear the traditional approach is not the best one suited to estimate the hardness of this problem. Instead, estimating the security requires first calculating the constant δ so than if an adversary is able to find a vector of length $\delta^{2n}\sqrt{q}$, she breaks the scheme. Next, the time required by various algorithms to enable such an estimate must be approximated, based on experimental data, before one may conclude whether the security level represented by δ actually is secure. It should go without saying that any security estimate of the latter type has to be continuously updated based on technical advances.

Another, perhaps more subtle difference is that even though both schemes apply modulus switching, RC-NTRU requires the largest modulus to be a composite which is gradually reduced by dividing out factors from both the modulus and (the slightly altered) ciphertext. This subject the various factors q_i to a lower bound, which is the same for all factors. On the other hand, BGV uses a ladder of ever decreasing moduli, where the relation of two adjacent moduli is roughly the same, making this relation the subject of a lower bound. The result is the same in one regard: the ratio between two adjacent moduli of both schemes is proportional to $n^{3/2}$, making the largest modulus proportional to $n^{3(D+1)/2}$, where D is the computational depth, and thus the number of modulus switchings which may be performed.

Despite this similarity the consequences are very different, as this dependency makes RC-NTRU susceptible to the attack described in Subsection 5.7 and [1], whereas it is of no consequence as far as security goes for BGV. This is without a doubt the difference of most importance for the two schemes, and stems from the different problems they base their security on. BGV is based on the assumption that RLWE is hard, which is provably as least as difficult as approximate-SVP in *any* ideal lattice. What is more: under this assumption, BGV is provably IND-CPA secure. On the other hand, RC-NTRU assumes that approximate-SVP is hard in the single lattice generated by the public key, an assumption which does not hold for a large set of parameters. Although this is believed to be hard for the original NTRU scheme (which is unaffected by the attack), there is no proof of this hardness, nor is there is any guarantee that this is transferable to RC-NTRU. Furthermore, it does not provide any provable security, merely experience and the fact that no one has been able to find a successful attack yet, despite not inconsiderable effort.

Of course, this last comment also applies to RLWE, as there is no concrete proof that approximate-SVP is hard in ideal lattices. Nevertheless, this is an entire class of relatively well-studied lattices, not merely a set of very particular lattices, as is the case for RC-NTRU. The conjecture that RLWE is hard is therefore considered fairly strong.

It should be pointed out that there does exist a provably IND-CPA secure version of NTRU which is based on RLWE with great similarities to RC-NTRU [22]. The difference lies mainly in the choice of q, which is chosen to be prime and congruent to 1 modulo $2n$ to enable basing the security on the classical reduction from the search to decision problem of RLWE mentioned in Section 4. The fact that q of the provably secure scheme is set much lower than what is the case of RC-NTRU, namely the same order of magnitude as n, also means the attack described in Subsection 5.7 and [1] may not be applied to the scheme.

This also means the scheme presented in [22] is not homomorphic, as q is much too small to be able to handle the growth of noise, seeing as it does not satisfy the strict bounds derived in this thesis. The two properties, homomorphic and secure, seem to be mutually exclusive for encryption schemes following the NTRU blueprint, as the problem security relies on appears to be hard for a rather too restricted set of parameters. This is a stark contrast to RLWE, which admittedly also holds for a limited set of parameters, but this set is much larger and versatile than what is the case for the secure NTRU setting.

8 Conclusion

The thesis has presented two schemes with great similarities in both structure and strategy of handling the inevitable growth of the noise the homomorphic operations cause. Most importantly, both schemes are over the same ring, support encryption of entire messages, as opposed to bit-wise encryption, and gradually reduce the ciphertext modulus to keep the noise manageable. However, this also imposes such large bounds on the parameters of one of the schemes that it must be deemed insecure, as the hardness of the problem it is based on is not transferable to the required parameters. Although the requirements for parameter size of the other encryption scheme are comparable to the first, these do not affect the security of this scheme.

Great care must therefore be taken when constructing a homomorphic encryption scheme, to ensure that the desired security may be combined with an effective way of handling the generated noise. The safest strategy therefore seems to be basing the scheme on problems proven to hold for a wide range of parameters and with a proven hardness reduction, such as RLWE. Basing a scheme on heuristics, such as NTRU is, might easily result in problems, not to mention difficulties correctly estimating the level of security, as was the case for the NTRU-based scheme presented here.

As far as efficiency goes, none of the currently developed fully homomorphic encryption schemes may be called efficient, as mentioned in the introduction. There have however been made great progress in the field in under 10 years, so there does seem to be cause for hope. The most likely route to an efficient scheme seems to be steering clear of the bootstrapping procedure, or greatly improving upon it, as it is easily the most costly procedure in the schemes.

An alternative to this would be attempting to develop a fully homomorphic encryption scheme which bases encryption on randomness instead of noise. This is likely to greatly ease the bounds the various parameters of an encryption scheme are subject to, and would probably result in a more efficient scheme compared to the currently implemented schemes.

There are thus three factors to be weighed against each other: security, homomorphism and efficiency, that ought to be prioritized in that order, as an efficient homomorphic scheme is of little use if it is not secure. Further investigation should be made into RLWE, to determine whether or not there are certain cases that are easier than others, and whether or not any findings affect the security of any (homomorphic) encryption schemes.

References

[1] Martin R. Albrecht, Shi Bai, and Léo Ducas. "A Subfield Lattice Attack on Overstretched NTRU Assumptions - Cryptanalysis of Some FHE and Graded Encoding Schemes". In: *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*. 2016, pp. 153–178. URL: http://dx.doi.org/10.1007/978-3-662-53018-4_6.

[2] Jacob Alperin-Sheriff and Chris Peikert. "Practical Bootstrapping in Quasilinear Time". In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*. 2013, pp. 1–20. URL: http://dx.doi.org/10.1007/978-3-642-40041-4_1.

[3] Frederik Armknecht et al. "A Guide to Fully Homomorphic Encryption". In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 1192. URL: http://eprint.iacr.org/2015/1192.

[4] Mihir Bellare et al. "Relations Among Notions of Security for Public-Key Encryption Schemes". In: *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*. 1998, pp. 26–45. URL: https://doi.org/10.1007/BFb0055718.

[5] Joppe W. Bos, Kristin E. Lauter, and Michael Naehrig. "Private predictive analysis on encrypted medical data". In: *Journal of Biomedical Informatics* 50 (2014), pp. 234–243. URL: http://dx.doi.org/10.1016/j.jbi.2014.04.003.

[6] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) Fully Homomorphic Encryption without Bootstrapping". In: *TOCT* 6.3 (2014), 13:1–13:36. URL: http://doi.acm.org/10.1145/2633600.

[7] Yuanmi Chen and Phong Q. Nguyen. "BKZ 2.0: Better Lattice Security Estimates". In: *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*. 2011, pp. 1–20. URL: https://doi.org/10.1007/978-3-642-25385-0_1.

[8] Marten van Dijk et al. "Fully Homomorphic Encryption over the Integers". In: *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*. 2010, pp. 24–43. URL: http://dx.doi.org/10.1007/978-3-642-13190-5_2.

[9] Nicolas Gama and Phong Q. Nguyen. "Predicting Lattice Reduction". In: *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*. 2008, pp. 31–51. URL: http://dx.doi.org/10.1007/978-3-540-78967-3_3.

[10] Craig Gentry, Shai Halevi, and Nigel P. Smart. "Homomorphic Evaluation of the AES Circuit". In: *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. 2012, pp. 850–867. URL: `http://dx.doi.org/10.1007/978-3-642-32009-5_49`.

[11] Craig Gentry et al. "Field switching in BGV-style homomorphic encryption". In: *Journal of Computer Security* 21.5 (2013), pp. 663–684. DOI: `10.3233/JCS-130480`. URL: `http://dx.doi.org/10.3233/JCS-130480`.

[12] Craig Gentry et al. "Ring Switching in BGV-Style Homomorphic Encryption". In: *Security and Cryptography for Networks - 8th International Conference, SCN 2012, Amalfi, Italy, September 5-7, 2012. Proceedings*. 2012, pp. 19–37. URL: `http://dx.doi.org/10.1007/978-3-642-32928-9_2`.

[13] Shai Halevi and Victor Shoup. "Bootstrapping for HElib". In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*. 2015, pp. 641–670. URL: `http://dx.doi.org/10.1007/978-3-662-46800-5_25`.

[14] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. "NTRU: A Ring-Based Public Key Cryptosystem". In: *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*. 1998, pp. 267–288. URL: `https://doi.org/10.1007/BFb0054868`.

[15] Jeffrey Hoffstein et al. *An introduction to mathematical cryptography*. Second edition. Springer, 2014, pp. 373–454. ISBN: 9781493917105.

[16] Jeffrey Hoffstein et al. "Choosing Parameters for NTRUEncrypt". In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 708. URL: `http://eprint.iacr.org/2015/708`.

[17] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *Advances in Cryptology – EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 – June 3, 2010. Proceedings*. Ed. by Henri Gilbert. Springer Berlin Heidelberg, 2010, pp. 1–23.

[18] Daniele Micciancio. "Generalized Compact Knapsacks, Cyclic Lattices, and Efficient One-Way Functions". In: *Computational complexity* 16.4 (2007), pp. 365–411.

[19] Daniele Micciancio. "The shortest vector in a lattice is hard to approximate to within some constant". In: *SIAM journal on Computing* 30.6 (2001), pp. 2008–2035.

[20] Daniele Micciancio and Oded Regev. "Lattice-based cryptography". In: *Post-quantum cryptography*. Springer, 2009, pp. 147–191.

[21] Kurt Rohloff and David Bruce Cousins. “A Scalable Implementation of Fully Homomorphic Encryption Built on NTRU”. In: *Financial Cryptography and Data Security - FC 2014 Workshops, BITCOIN and WAHC 2014, Christ Church, Barbados, March 7, 2014, Revised Selected Papers*. 2014, pp. 221–234. URL: `http://dx.doi.org/10.1007/978-3-662-44774-1_18`.

[22] Damien Stehlé and Ron Steinfeld. “Making NTRU as Secure as Worst-Case Problems over Ideal Lattices”. In: *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*. 2011, pp. 27–47. URL: `http://dx.doi.org/10.1007/978-3-642-20465-4_4`.

[23] Douglas R Stinson. *Cryptography: theory and practice.* Third edition. CRC press, 2005, pp. 173, 235.