Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes

N.P. Smart[1] and F. Vercauteren[2]

[1] Dept. Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom.
nigel@cs.bris.ac.uk

[2] COSIC - Electrical Engineering,
Katholieke Universiteit Leuven,
Kasteelpark Arenberg 10,
B-3001 Heverlee,
Belgium.
fvercaut@esat.kuleuven.ac.be

Abstract. We present a fully homomorphic encryption scheme which has both relatively small key and ciphertext size. Our construction follows that of Gentry by producing a fully homomorphic scheme from a "somewhat" homomorphic scheme. For the somewhat homomorphic scheme the public and private keys consist of two large integers (one of which is shared by both the public and private key) and the ciphertext consists of one large integer. As such, our scheme has smaller message expansion and key size than Gentry's original scheme. In addition, our proposal allows efficient fully homomorphic encryption over any field of characteristic two.

1 Introduction

A fully homomorphic public key encryption scheme has been a "holy grail" of cryptography for a very long time. In the last year this problem has been solved by Gentry [7, 8], by using properties of ideal lattices. Various cryptographic schemes make use of lattices, sometimes just to argue about their security (such as NTRU [11]), in other cases lattices are vital to understand the workings of the scheme algorithms (such as [9]). Gentry's fully homomorphic scheme falls into the latter category. In this paper we present a fully homomorphic scheme which can be described using the elementary theory of algebraic number fields, and hence we do not require lattices to understand its encryption and decryption operations. However, our scheme does fall into the category of schemes whose best known attack is based on lattices.

At a high level our scheme is very simple, and is mainly parametrized by an integer N (there are other parameters which are less important). The public key

consists of a prime p and an integer α modulo p. The private key consists of either an integer z (if we are encrypting bits), or an integer polynomial $Z(x)$ of degree $N-1$ (if we are encrypting general binary polynomials of degree $N-1$). To encrypt a message one encodes the message as a binary polynomial, then one randomizes the message by adding on two times a small random polynomial. To obtain the ciphertext, the resulting polynomial is evaluated at α modulo p. As such, the ciphertext is simply an integer modulo p (irrespective of whether we are encrypting bits or binary polynomials of degree $N-1$).

To decrypt in the case where we know the message is a single bit, we multiply the ciphertext by z and divide by p. We then round this rational number to the nearest integer value, and subtract the result from the ciphertext. The plaintext is then recovered by reducing this intermediate result modulo 2. When we are decrypting a binary polynomial we follow the same procedure, but this time we multiply by the polynomial $Z(x)$ and divide by p, to obtain a rational polynomial. Rounding the coefficients of this polynomial to the nearest integer, subtracting from the original ciphertext, and reducing modulo two will result again in recovering the plaintext.

Acknowledgements: The authors would like to thank the eCrypt NoE funded by the EU for partially supporting the work in this paper. The first author was supported by a Royal Society Wolfson Merit Award, whilst the second was supported by a post-doctoral fellowship of the Research Foundation - Flanders. The authors would also like to thank Florian Hess, Steven Galbraith and Joe Silverman for valuable comments on earlier versions of this paper and Craig Gentry for detailed explanations of his paper [7].

2 Preliminaries

2.1 Notation

Given a polynomial $g(x) = \sum_{i=0}^{t} g_i x^i \in \mathbb{Q}[x]$, we define the 2-norm and ∞-norm as

$$\|g(x)\|_2 = \sqrt{\sum_{i=0}^{t} g_i^2} \quad \text{and} \quad \|g(x)\|_\infty = \max_{i=0,\ldots,t} |g_i| \,.$$

For a positive value r, we define two corresponding types of "ball" centered at the origin:

$$\mathcal{B}_{2,N}(r) = \left\{ \sum_{i=0}^{N-1} a_i x^i : \sum_{i=0}^{N-1} a_i^2 \leq r^2 \right\},$$
$$\mathcal{B}_{\infty,N}(r) = \left\{ \sum_{i=0}^{N-1} a_i x^i : -r \leq a_i \leq r \right\}.$$

We have the usual inclusions $\mathcal{B}_{2,N}(r) \subset \mathcal{B}_{\infty,N}(r)$ and $\mathcal{B}_{\infty,N}(r) \subset \mathcal{B}_{2,N}(\sqrt{N} \cdot r)$. We also define the following half-ball

$$\mathcal{B}^{+}_{\infty,N}(r) = \left\{ \sum_{i=0}^{N-1} a_i x^i : 0 \leq a_i \leq r \right\}.$$

All reductions in this paper modulo an odd integer m are defined to result in a value in the range $[-(m-1)/2, \ldots, (m-1)/2]$. The notation $a \leftarrow b$, means assign the value on the left to the value on the right. Whereas $a \leftarrow_R A$ where A is a set, means select a from the set A using a uniform distribution.

2.2 Ideals in Number Fields

Since the underlying workings of our scheme are based on prime ideals in a number field, we first recap on some basic properties. See [4] for an introduction to the elementary computational number theory needed.

Let K be a number field $\mathbb{Q}(\theta)$ where θ is a root of a monic irreducible polynomial $F(x) \in \mathbb{Z}[x]$ of degree N. Consider the equation order $\mathbb{Z}[\theta]$ inside the ring of integers $\mathcal{O}_K$. For our parameter choices we typically have $\mathcal{O}_K = \mathbb{Z}[\theta]$, but this need not be the case in general. Our scheme works with ideals in $\mathbb{Z}[\theta]$ that are assumed coprime with the index $[\mathcal{O}_K : \mathbb{Z}[\theta]]$, so there is little difference with working in $\mathcal{O}_K$. These ideals can be represented in one of two ways, either by an N-dimensional $\mathbb{Z}$-basis or as a two element $\mathbb{Z}[\theta]$-basis. When presenting an ideal $\mathfrak{a}$ as an N-dimensional $\mathbb{Z}$ basis we give N elements $\gamma_1, \ldots, \gamma_N \in \mathbb{Z}[\theta]$, and every element in $\mathfrak{a}$ is represented by the $\mathbb{Z}$-module generated by $\gamma_1, \ldots, \gamma_N$. It is common practice to present this basis as an $n \times n$-matrix. The matrix is then set to be $(\gamma_{i,j})$, where we set $\gamma_i = \sum_{j=0}^{N-1} \gamma_{i,j}\theta^j$, i.e. we take a row oriented formulation. Taking the Hermite Normal Form (HNF) of this basis will produce a lower triangular basis in which the leading diagonal $(d_1, \ldots, d_N)$ satisfies $d_{i+1}|d_i$. Note that this last property of the HNF of a basis only follows for matrices corresponding to ideals [5] (who use a different orientation).

However, every such ideal can also be represented by a $\mathbb{Z}[\theta]$-basis given by two elements, $\langle \delta_1, \delta_2 \rangle$. In particular one can always select δ_1 to be an integer. For ideals lying above a rational prime p, it is very easy to write down a two element representation of an ideal. If we factor $F(x)$ modulo p into irreducible polynomials

$$F(x) = \prod_{i=1}^{t} F_i(x)^{e_i} \pmod{p}$$

then, for p not dividing $[\mathcal{O}_K : \mathbb{Z}[\theta]]$, the prime ideals dividing $p\mathbb{Z}[\theta]$ are given by the two element representation

$$\mathfrak{p}_i = \langle p, F_i(\theta) \rangle .$$

We define the residue degree of $\mathfrak{p}_i$ to be equal to the degree d_i of the polynomial $F_i(x)$. Reduction modulo $\mathfrak{p}_i$ produces a homomorphism

$$\iota_{\mathfrak{p}_i} : \mathbb{Z}[\theta] \longrightarrow \mathbb{F}_{p^{d_i}}.$$

We will be particularly interested in prime ideals of residue degree one. These can be represented as a two element representation by $\langle p, \theta - \alpha \rangle$ where p is the norm of the ideal and α is a root of $F(x)$ modulo p. If $\chi \in \mathbb{Z}[\theta]$ is given by $\chi = \sum_{i=0}^{N-1} c_i \theta^i$ then the homomorphism $\iota_{\mathfrak{p}}$ simply corresponds to evaluation of the polynomial $\chi(\theta)$ in α modulo p.

Given a prime ideal of the form $\langle p, \theta - \alpha \rangle$, the corresponding HNF representation is very simple to construct, and is closely related to the two element representation, as we shall now show. We need to row reduce the $2N \times N$ matrix

$$\begin{pmatrix} p & & & & & & \\ & p & & & 0 & & \\ & & & \ddots & & & \\ & 0 & & & & & \\ & & & & & & p \\ -\alpha & 1 & & & 0 & & \\ & -\alpha & 1 & & & & \\ 0 & & & \ddots & \ddots & & \\ & & & & -\alpha & & 1 \\ -F_0 & -F_1 & -F_2 & \dots & -F_{N-2} & -F_{N-1} & -\alpha \end{pmatrix}$$

where $F(x) = \sum_{i=0}^{N} F_i x^i$. It is not hard to see that the HNF of the above matrix is then given by

$$H = \begin{pmatrix} p & & & 0 \\ -\alpha & 1 & & \\ -\alpha^2 & & 1 & \\ \vdots & & & \ddots \\ -\alpha^{N-1} & 0 & & 1 \end{pmatrix},$$

where all the integers in the first column, in rows two and onward, are taken modulo p.

Recall that an ideal is called principal if it is generated by one element, i.e. we can write $\mathfrak{p} = \langle \gamma \rangle = \gamma \cdot \mathbb{Z}[\theta]$. Note that given an HNF or two-element representation of an ideal, determining whether it is principal, and finding a generator is considered to be a hard problem for growing N. Indeed the best known algorithms (which are essentially equivalent to finding the class and unit group of a number field) run in exponential time in the degree of the field. For fixed degree they run in sub-exponential time in the discriminant [2]. In addition the generator of a principal ideal output by these algorithms will be very large. Indeed, this generator will typically be so large that writing it down as a polynomial in θ may itself take exponential time [14]. Thus finding a small generator of a principal ideal is possibly an even harder problem. Quantumly finding a generator of a principal ideal is relatively easy [10], however writing down a small generator is not known to be easy.

3 Our Somewhat Homomorphic Scheme

In this section we present our somewhat homomorphic scheme and analyze for which parameter sets decryption works. To simplify the presentation we present the scheme at this point as one which just encrypts elements in $\mathcal{P} = \{0, 1\}$.

3.1 The Scheme

A somewhat homomorphic encryption scheme consists of five algorithms: {KeyGen, Encrypt, Decrypt, Add, Mult}. We shall describe each in turn; notice that the most complex phase is that of KeyGen. The scheme is parametrized by three values (N, η, μ). A typical set of parameters would be $(N, 2^{\sqrt{N}}, \sqrt{N})$. Later we shall return to discussing the effects of the sizes of these values on the security level λ and performance of the scheme.

KeyGen():

- Set the plaintext space to be $\mathcal{P} = \{0, 1\}$.
- Choose a monic irreducible polynomial $F(x) \in \mathbb{Z}[x]$ of degree N.
- Repeat:
  - $S(x) \leftarrow_R \mathcal{B}_{\infty,N}(\eta/2)$.
  - $G(x) \leftarrow 1 + 2 \cdot S(x)$.
  - $p \leftarrow \mathsf{resultant}(G(x), F(x))$.
- Until p is prime.
- $D(x) \leftarrow \gcd(G(x), F(x))$ over $\mathbb{F}_p[x]$.
- Let $\alpha \in \mathbb{F}_p$ denote the unique root of $D(x)$.
- Apply the XGCD-algorithm over $\mathbb{Q}[x]$ to obtain $Z(x) = \sum_{i=0}^{N-1} z_i x^i \in \mathbb{Z}[x]$ such that
$$Z(x) \cdot G(x) = p \mod F(x).$$
- $B \leftarrow z_0 \pmod{2p}$.
- The public key is $\mathsf{PK} = (p, \alpha)$, whilst the private key is $\mathsf{SK} = (p, B)$.

Encrypt(M, PK):

- Parse PK as (p, α).
- If $M \notin \{0, 1\}$ then abort.
- $R(x) \leftarrow_R \mathcal{B}_{\infty,N}(\mu/2)$.
- $C(x) \leftarrow M + 2 \cdot R(x)$.
- $c \leftarrow C(\alpha) \pmod{p}$.
- Output c.

Decrypt(c, SK):

- Parse SK as (p, B).
- $M \leftarrow (c - \lfloor c \cdot B/p \rceil) \pmod 2$.
- Output M.

Add(c_1, c_2, PK):

- Parse PK as (p, α).
- $c_3 \leftarrow (c_1 + c_2) \pmod{p}$.
- Output c_3.

Mult(c_1, c_2, PK):

- Parse PK as (p, α).
- $c_3 \leftarrow (c_1 \cdot c_2) \pmod{p}$.
- Output c_3.

3.2 Analysis

In this section we analyze for which parameter sets our scheme is correct and also determine how many homomorphic operations can be performed before decryption will fail.

KeyGen algorithm: We can see that KeyGen generates an element $\gamma = G(\theta)$ of prime norm p in the number field K defined by $F(x)$. As such we have constructed a small generator of the degree one prime ideal $\mathfrak{p} = \gamma \cdot \mathbb{Z}[\theta]$. To find the two element representation of $\mathfrak{p}$, we need to select the correct root α of $F(x)$ modulo p. Since $\gamma = G(\theta) \in \mathfrak{p}$, we have that $G(\alpha) \equiv 0 \bmod p$, so $G(x)$ and $F(x)$ have at least one common root modulo p. Furthermore, there will be precisely one root in common, since otherwise γ would generate two different prime ideals, which clearly is impossible. This explains the fact that $D(x)$ has degree one; we are using $D(x)$ to select the precise root of $F(x)$ which corresponds to the ideal $\mathfrak{p}$ generated by γ. The two element representation of the ideal $\mathfrak{p}$ then simply is $\mathfrak{p} = p \cdot \mathbb{Z}[\theta] + (\theta - \alpha)\mathbb{Z}[\theta]$.

Encrypt algorithm: The message M is added to twice a small random polynomial $R(x)$ resulting in a polynomial $C(x)$. The ∞-norm of the polynomial $R(x)$ is controlled by the parameter μ. Encryption then simply equals reduction of $C(\theta)$ modulo $\mathfrak{p}$ using the public two element representation $\langle p, \theta - \alpha \rangle$. As explained before, this simply corresponds to evaluating $C(x)$ in α modulo p. Furthermore, note that this precisely implies that $C(\theta) - c \in \mathfrak{p}$.

Decrypt algorithm: By definition of encryption, we have that $C(\theta) - c \in \mathfrak{p}$ and $\mathfrak{p}$ is principal and generated by $\gamma = G(\theta)$. Hence, we can write

$$C(\theta) - c = q(\theta) \cdot \gamma \,,$$

with $q(\theta) \in \mathbb{Z}[\theta]$. It is clear that if we recover the element $C(\theta)$, then decryption will work since $C(\theta) = M + 2 \cdot R(\theta)$. Note that γ^{-1} is precisely given by $Z(\theta)/p$, where Z was computed in KeyGen. Dividing by γ therefore leads to the following equality

$$-c \cdot Z(\theta)/p = q(\theta) - \left(C(\theta) \cdot Z(\theta)\right)/p \,.$$

The above equation shows that if $\|C(\theta) \cdot Z(\theta)/p\|_\infty < 1/2$, then simply rounding the coefficients of $-c \cdot Z(\theta)/p$ will result in the correct quotient $q(\theta)$. This will allow for correct decryption by computing $C(\theta) = c + q(\theta) \cdot \gamma$. The crucial part therefore is to obtain a bound on $\|Z(x)\|_\infty$.

Lemma 1. *Let $F(x), G(x) \in \mathbb{Z}[x]$ with $F(x)$ monic, $\deg(F) = N$ and $\deg(G) = M < N$ and $\mathsf{resultant}(F, G) = p$, then there exists a polynomial $Z(x) \in \mathbb{Z}[x]$ with $Z(x) \cdot G(x) = p \bmod F(X)$ and*

$$\|Z(x)\|_\infty \leq \|G(x)\|_2^{N-1} \cdot \|F(x)\|_2^M \,.$$

PROOF: Over $\mathbb{Q}[x]$, we have that $\gcd(G(x), F(x)) = 1$, so there exists polynomials $S(x), T(x) \in \mathbb{Q}[x]$ with $\deg(S) < N$ and $\deg(T) < M$ such that $S(x) \cdot G(x) + T(x) \cdot F(x) = 1$. It is well known (see for instance Corollary 6.15 of [6]) that the polynomials S and T are given by $S = \sum_{i=0}^{N-1} s_i x^i$ and $T = \sum_{i=0}^{M-1} t_i x^i$, where the s_i and t_i are the solutions of

$$\mathsf{Syl}(G, F)^{\mathrm{T}} \cdot \begin{pmatrix} s_{N-1} \\ \vdots \\ s_0 \\ t_{M-1} \\ \vdots \\ t_0 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ \\ \vdots \\ 0 \\ 1 \end{pmatrix},$$

where $\mathsf{Syl}(G, F)$ is the Sylvester matrix of G and F. The resultant is precisely $\det(\mathsf{Syl}(G, F)) = p$, so by Cramer's rule we find an explicit expression for the coefficients s_i, namely, the determinant of a submatrix of $\mathsf{Syl}(G, F)^{\mathrm{T}}$ (remove one of the columns containing the coefficients of G and the last row) divided by p. Using Hadamard's inequality to bound the determinant of such submatrices, we finally conclude that $|z_i| \leq \|G\|_2^{N-1} \cdot \|F\|_2^M$. □

In the remainder of the paper we will assume that $M = N - 1$ which will happen with very high probability.

Define

$$\delta_\infty := \sup \left\{ \frac{\|g(x) \cdot h(x) \bmod F(x)\|_\infty}{\|g(x)\|_\infty \cdot \|h(x)\|_\infty} \mid \deg(g), \deg(h) < N \right\}.$$

We then have that

$$\|g(\theta) \cdot h(\theta)\|_\infty \leq \delta_\infty \cdot \|g\|_\infty \cdot \|h\|_\infty,$$

where $\deg(g), \deg(h) < N$. Gentry [8, Section 7.4] derives several bounds on the above quantity but for the 2-norm and it is easy to obtain the equivalent bounds for the ∞-norm. To illustrate the two extreme cases, i.e. that δ_∞ can range from fully exponential in N to linear in N, we give the following lemma, which also motivates why we propose to use $F(x) = x^{2^n} + 1$ in practice.

Lemma 2. *Let $F_1(x) = x^N - a$ and $F_2(x) = x^N - ax^{N-1}$ then*

$$\delta_\infty(F_1) \leq |a|N \qquad \text{and} \qquad \delta_\infty(F_2) \leq |a|^{N-1}N\,.$$

PROOF: Let $g = \sum_{i=0}^{N-1} g_i x^i$ and $h = \sum_{i=0}^{N-1} h_i x^i$, then

$$g \cdot h \bmod F_1 = \sum_{k=0}^{N-1} \left(\sum_{0 \leq i \leq k} g_i h_{k-i} + a \sum_{k < i < N} g_i h_{N+k-i} \right) x^k\,,$$

from which the bound on $\delta_\infty(F_1)$ immediately follows. Similarly, write $g \cdot h = \sum_{k=0}^{2N-2} c_k x^k$, then $g \cdot h \bmod F_2 = \sum_{k=0}^{N-1} d_k x^k$ with $d_k = c_k$ for $k = 0, \ldots, N-2$ and

$$d_{N-1} = \sum_{i=0}^{N-1} c_{N-1+i} a^i$$

Since all c_i clearly are smaller than $N\|g\|_\infty\|h\|_\infty$ the bound on $\delta_\infty(F_2)$ follows. □

From this we can conclude that

$$\left\| \frac{C(\theta) \cdot Z(\theta)}{p} \right\|_\infty \leq \frac{\delta_\infty \cdot \|C\|_\infty \cdot \|G\|_2^{N-1} \cdot \|F\|_2^{N-1}}{p},$$

so decryption will work as long as

$$\|C\|_\infty < \frac{p}{2 \cdot \delta_\infty \cdot \|G\|_2^{N-1} \cdot \|F\|_2^{N-1}} = \mathsf{r}_{\mathsf{Dec}}.$$

Note that the expected value of $\mathsf{r}_{\mathsf{Dec}}$ will be roughly $\|G\|_2/2\delta_\infty$, since the resultant p will be about $\|G\|_2^N \cdot \|F\|_2^{N-1}$. So for $\|C\|_\infty < \mathsf{r}_{\mathsf{Dec}}$, we have

$$C(x) = c + q(\theta) \cdot \gamma = c - \lfloor c \cdot Z(x)/p \rceil \gamma,$$

and since $M \equiv C(x) \bmod 2$ and $\gamma \equiv 1 \bmod 2$ we finally obtain the simplified decryption function

$$M \equiv c - \lfloor c \cdot B/p \rceil \bmod 2,$$

where B is z_0. Note, we can take B as z_0 modulo $2p$ as we are only interested in rounding $c \cdot B/p$ to the nearest integer and then taking the result modulo 2. Furthermore, Lemma 1 implies that all coefficients of $Z(x)$ typically will be smaller than p, since $p = \mathsf{resultant}(F, G)$ and thus $p \simeq \|G(x)\|_2^N \cdot \|F(x)\|_2^M$. This means that the reduction modulo $2p$ in the key generation will have no effect in most cases. However, it will turn out to be a necessary assumption in assuring a uniform distribution when we switch to the full homomorphic scheme.

For our KeyGen algorithm we have that each coefficient of G has size approximately η, which implies that we have the estimate

$$\mathsf{r}_{\mathsf{Dec}} \approx \frac{\sqrt{N} \cdot \eta}{2 \cdot \delta_\infty}.$$

For $F(x) = x^N + 1$ we thus obtain the estimate $\mathsf{r}_{\mathsf{Dec}} \approx \eta/(2 \cdot \sqrt{N})$. In the remainder of the paper we will also sometimes use $\mathsf{r}_{\mathsf{Enc}}$ instead of μ. Note that if one wants to compare with Gentry's scheme, one should take into account that our bounds are formulated for the ∞-norm, whereas Gentry works with the 2-norm.

Add and Mult algorithms: It is clear that both algorithms are correct. However, we need to consider how the error values propagate as we apply Add and Mult. In particular, decryption of $c = C(\alpha)$ will work for a polynomial $C(x)$ if $C(x) \in \mathcal{B}_{\infty,N}(\mathsf{r}_{\mathsf{Dec}})$. However, as we apply Add and Mult to a ciphertext the value of $C(x)$ starts to lie in balls of larger and larger radius. As soon as $C(x) \notin \mathcal{B}_{\infty,N}(\mathsf{r}_{\mathsf{Dec}})$, we are no longer guaranteed to be able to decrypt correctly. This is why our basic scheme is only somewhat homomorphic, since we are only able to apply Add and Mult a limited number of times.

Let c_1 and c_2 denote two ciphertexts, corresponding to two randomizations $C_1(x) = M_1 + N_1(x)$ and $C_2(x) = M_2 + N_2(x)$; where $M_i \in \{0, 1\}$ are the messages and $N_i(x) \in \mathcal{B}_{\infty,N}(r_i - 1)$ is the randomness, i.e. $C_i(x) \in \mathcal{B}_{\infty,N}(r_i)$. We let

$$C_3(x) = M_3 + N_3(x) = (M_1 + N_1(x)) + (M_2 + N_2(x)),$$
$$C_4(x) = M_4 + N_4(x) = (M_1 + N_1(x)) \cdot (M_2 + N_2(x)),$$

where $M_3, M_4 \in \{0, 1\}$. Then

$$C_3(x) \in \mathcal{B}_{\infty,N}(r_1 + r_2)$$

and

$$C_4(x) \in \mathcal{B}_{\infty,N}(\delta_\infty \cdot r_1 \cdot r_2).$$

Initially we start with a ciphertext with $C(x)$ lying in $\mathcal{B}_{\infty,N}(\mu+1)$. After executing a circuit with multiplicative depth d, we expect the ciphertext to correspond to a polynomial $C'(x)$ lying in a ball $\mathcal{B}_{\infty,N}(r)$ with

$$r \approx (\delta_\infty \cdot \mu)^{2^d}.$$

Thus we can only decrypt the output of such a circuit if $r \leq \mathsf{r}_{\mathsf{Dec}}$, i.e.

$$\begin{aligned} d \log 2 &\leq \log\log \mathsf{r}_{\mathsf{Dec}} - \log\log(\delta_\infty \cdot \mu) \\ &\approx \log\log\left(\frac{\sqrt{N} \cdot \eta}{2 \cdot \delta_\infty}\right) - \log\log(\delta_\infty \cdot \mu). \end{aligned}$$

4 Security Analysis

We consider three aspects of security; key recovery, onewayness of the encryption and semantic security. Whilst semantic security is based on what might at first appear a non-traditional problem, the other two notions of security are related to well studied problems in number theory. This is similar to other notions in cryptography; for example key recovery in ElGamal is related to the DLP problem, and semantic security to the relatively obscure (for mathematicians) DDH problem. However, we first show that our scheme is in some sense a specialisation and optimization of Gentry's scheme.

Link With Gentry's Scheme: To discuss the security in more detail, we first show that our scheme is a specialisation and simplification of the lattice based scheme of Gentry [7]. The generator γ in our scheme is equivalent to the private basis of the ideal J in Gentry's scheme, the public basis is then the two element representation $\langle p, \theta - \alpha \rangle$. The ideal I of Gentry's scheme is simply set to the principal ideal $\langle 2 \rangle$. Therefore, we see that KeyGen is a specialised form of KeyGen for Gentry's scheme: in particular we use the compact two element representation $\langle p, \alpha \rangle$ of the public basis, instead of the larger HNF representation as Gentry does.

We now turn to the encryption algorithm. The element $C(\theta) = M(\theta) + 2 \cdot R(\theta)$ is precisely the value of ψ' computed in Gentry's encryption algorithm, with a value of $\mathsf{r}_{\mathsf{Enc}}$ (in the 2-norm) equal to $\sqrt{N} \cdot \mu$. Gentry then produces his ciphertext ψ by reducing ψ' modulo the ideal J using the HNF basis. It is at this point that we seem to depart from Gentry's presentation: we actually compute the reduction of ψ' modulo $\mathfrak{p}$ using the public two element representation. Given ψ' as a polynomial in θ, this involves replacing θ by α and reducing the result modulo p. So given $C(x)$, we produce c by simply computing $c = \iota_{\mathfrak{p}}(C(\theta)) \in \mathbb{F}_p$. However, given our earlier discussion on the HNF of the ideal given by $\langle p, \theta - \alpha \rangle$ we see that the two reduction algorithms are equivalent when we are working in the equation order $\mathbb{Z}[\theta]$.

Hence, we conclude that our scheme is a specialisation of Gentry's scheme. For the given specialisation our key sizes are much smaller than Gentry's, whilst our ciphertexts are the same size. When compared to the full generality of Gentry's scheme our ciphertexts are also much smaller than Gentry's. The link between the two schemes, and the relative simplicity of our scheme, may help shed light on parameter choices in Gentry's original scheme.

Key Recovery: Recall the public key in our scheme consists of a principal degree one prime ideal in two element representation, whilst the private key consists of the inverse of a small generator of this principal prime ideal. To see that the generator γ is small, notice that the polynomial $G(x)$ has an ∞-norm given roughly by η, whereas the size of p is roughly $\sqrt{N}^N \eta^N \cdot \|F\|_2^{N-1}$. Recovering the private key given the public key is therefore an instance of the small principal ideal problem:

Definition 1 (Small Principal Ideal Problem (SPIP)) *Given a principal ideal $\mathfrak{a}$ in either two element or HNF representation compute a "small" generator of the ideal.*

This is one of the core problems in computational number theory and has formed the basis of previous cryptographic proposals, see for example [3]. There are currently two approaches to the above problem. The first approach is a deterministic method based on the Baby-Step/Giant-Step method attributed to [1]. This takes time

$$N^{O(N)} \cdot \sqrt{\min(A, R)} \cdot |\Delta|^{o(1)},$$

where Δ is the discriminant of $\mathbb{Z}[\theta]$, R is the regulator and $A = \min_{i=1}^{N} \log |\gamma^{(i)}|$ is the mimimal logarithmic embedding of γ. Clearly A can itself be bounded by η, a minor detail which we leave to the reader.

The second approach to this problem is via Buchmann's sub-exponential algorithm for units and class groups which is described in [2] and [4][Chapter 6]. This method has complexity

$$\exp\left(O(N \log N) \cdot \sqrt{\log(\Delta) \cdot \log\log(\Delta)}\right)$$

where again Δ is the discriminant of the order $\mathbb{Z}[\theta]$. However, this method is likely to produce a generator of large height, i.e. with large coefficients. Indeed so large, that writing the obtained generator down as a polynomial in θ may take exponential time.

In conclusion determining the private key given only the public key is an instance of a classical and well studied problem in algorithmic number theory. In particular there are no efficient solutions for this problem, and the only sub-exponential method does not find a solution which is equivalent to our private key.

Onewayness of Encryption: In this section we consider the problem of recovering a message given a ciphertext element. It is readily seen that this is equivalent to solving the following problem: Given p and $\alpha, c \in \mathbb{F}_p$ find x_i for $i = 0, \ldots, N-1$, such that

$$\sum_{i=0}^{N-1} x_i \cdot \alpha^i = c - k \cdot p,$$

where $|x_i| \leq \mathsf{r}_{\mathsf{Enc}}$, for some integer value of k.

To recast this as a lattice problem, consider the lattice generated by the rows of the matrix H given earlier. Consider the lattice vector

$$(k, -x_1, \ldots, -x_n) \cdot H = (c - x_0, -x_1, \ldots, -x_n).$$

This is a lattice vector which is very close (within $\mathsf{r}_{\mathsf{Enc}}$ in the ∞-norm, or $\sqrt{N} \cdot \mathsf{r}_{\mathsf{Enc}}$ in the 2-norm) to the non-lattice vector $(c, 0, \ldots, 0)$. Hence, determining the underlying plaintext given the ciphertext is an instance of the closest vector problem.

However, the underlying lattice is a well-studied lattice in algorithmic number theory, see for example the applications of LLL described in [12, 13, 15]. A lattice generated by a matrix such as H, namely a matrix in Hermite Normal Form in which all but one diagonal entry is equal to one, is probably the most studied lattice problem from the computational perspective in number theory. Thus whilst we are unable to make use of modern worst-case/average-case reductions for our scheme, the underlying lattice problem is well studied.

However, for later use, we will recap on the analysis Gentry has given for this problem. Although one should bear in mind that Gentry's analysis is for a

general lattice arising from the HNF of an ideal and not for the specific one in our scheme. The best known attack on Gentry's scheme is one of lattice reduction, related to the bounded distance decoding problem (BDDP). In particular it is related to finding short/closest vectors within a multiplicative factor of $\mathsf{r}_{\mathsf{Dec}}/\mathsf{r}_{\mathsf{Enc}}$ in a lattice of dimension N. If we set

$$2^{\epsilon} = \frac{\mathsf{r}_{\mathsf{Dec}}}{\mathsf{r}_{\mathsf{Enc}}} = \frac{\sqrt{N} \cdot \eta}{2 \cdot \delta_{\infty} \cdot \mu},$$

then it is believed that solving BDDP has difficulty $2^{N/\epsilon}$ (see [8][Section 7.7]). We shall refer to the value $2^{N/\epsilon}$ as the security level of our somewhat homomorphic scheme.

Semantic Security: Finally we discuss the semantic security of our somewhat homomorphic encryption scheme. Consider the following distinguishing problem:

Definition 2 (Polynomial Coset Problem (PCP)) *The challenger first selects* $b \leftarrow_R \{0,1\}$ *and runs* KeyGen *as above to obtain a value of* α *and* p*. If* $b = 0$ *then the challenger performs*

- $R(x) \leftarrow_R \mathcal{B}_{\infty,N}(\mathsf{r}_{\mathsf{Enc}})$.
- $r \leftarrow R(\alpha) \pmod{p}$.

Whilst if $b = 1$ *the challenger performs*

- $r \leftarrow_R \mathbb{F}_p$.

Given (r, PK) *the problem is to guess whether* $b = 0$ *or* $b = 1$.

We call the problem the Polynomial Coset Problem as it is akin to Gentry's Ideal Coset Problem from [7]. The problem basically says one cannot determine whether r is the evaluation of some small polynomial at α or is a random value modulo p. Note that the size of the space $\mathcal{B}_{\infty,N}(\mathsf{r}_{\mathsf{Enc}})$ is roughly ${\mathsf{r}_{\mathsf{Enc}}}^N$, whereas $\mathbb{F}_p$ has size η^N. So if $\mathsf{r}_{\mathsf{Enc}}$ is much smaller than η, we are trying to distinguish a relatively small space within a larger one. Note, in the case where $b = 0$ we generate the value $R(x)$ from $\mathcal{B}_{\infty,N}(\mathsf{r}_{\mathsf{Enc}})$ as opposed to $\mathcal{B}_{\infty,N}(\mathsf{r}_{\mathsf{Dec}})$, since we are interested in arguing about semantic security for what are the simplest ciphertexts to break.

The proof of the following theorem closely follows the proof of Theorem 7 of [7], but we include it here for completeness.

Theorem 1. *Suppose there is an algorithm* $\mathcal{A}$ *which breaks the semantic security of our somewhat homomorphic scheme with advantage* ϵ*. Then there is an algorithm* $\mathcal{B}$*, running in about the same time as* $\mathcal{A}$*, which solves the PCP with advantage* $\epsilon/2$.

Proof: The algorithm $\mathcal{B}$ creates a challenge ciphertext for algorithm $\mathcal{A}$ from its own challenge (r, PK) by setting

$$c \leftarrow (M_{\beta}(\alpha) + 2 \cdot r) \pmod{p},$$

where M_0 and M_1 are $\mathcal{A}$'s two challenge messages and $\beta \leftarrow_R \{0,1\}$, is $\mathcal{B}$'s choice of a challenge bit. $\mathcal{A}$ sends back a guess β' for β and $\mathcal{B}$ returns $\beta \oplus \beta'$.

When $b = 0$ in the PCP problem, it is clear that the challenge ciphertext c has the correct distribution, so $\mathcal{B}$ obtains the same advantage as $\mathcal{A}$, namely ϵ. When $b = 1$, r is uniformly random modulo p and since p is odd, $2r$ is uniformly random modulo p and therefore so is c. Hence, the advantage of $\mathcal{A}$ is 0, which implies that $\mathcal{B}$'s overall advantage is $\epsilon/2$. □

5 A Fully Homomorphic Scheme

We now proceed to turning the somewhat homomorphic scheme into a fully homomorphic scheme. Since we have shown that our scheme is a specialisation of Gentry's scheme, we only need to recast Gentry's method for our parameters. Indeed we can simplify the method somewhat, since our ciphertext is an integer rather than a vector. We assume that our scheme is secure under key dependent encryptions, purely to keep the notation simpler; to deal with the more general case is immediate from our discussion.

At a high level we need to define a new algorithm called $\mathsf{Recrypt}$, which takes a ciphertext c and re-encrypts it to c_{new}, whilst at the same time removing some of the errors in c. Intuitively this takes a "dirty ciphertext" c and "cleans it" to obtain the ciphertext c_{new}.

To do this we augment the encryption key with some additional information, by extending the algorithm KeyGen with the following additional operations, based on two integer parameters s_1 and s_2. We make use of the fact that we are only interested in the coefficients of $Z(x)$ modulo $2p$.

- Generate s_1 uniformly random integers B_i in $[-p, \ldots, p]$ such that there exists a subset S of s_2 elements with
$$\sum_{j \in S} B_j = B$$
over the integers.
- Define $\mathsf{sk}_i = 1$ if $i \in S$ and 0 otherwise. Notice that only s_2 of the bits $\{\mathsf{sk}_i\}$ are set to one.
- Encrypt the bits sk_i under the somewhat homomorphic scheme to obtain $\mathfrak{c}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}_i, \mathsf{PK})$.
- The public key now consists of
$$\mathsf{PK} = (p, \alpha, s_1, s_2, \{\mathfrak{c}_i, B_i\}_{i=1}^{s_1}) .$$

We can now describe the re-encryption operation.

$\mathsf{Recrypt}(c, \mathsf{PK})$: This algorithm takes as input a "dirty" ciphertext c, and then produces a "cleaner" ciphertext c_{new} of the same message, but with less "errors" in its randomization vector. The re-encryption works by performing a homomorphic decryption on an encryption of the ciphertexts bits. In the Appendix we

explain the Recrypt algorithm in detail and analyse precisely how complicated it is for possible real life values.

Note that we have

$$B = \sum_{i=1}^{s_1} \mathsf{sk}_i \cdot B_i,$$

hence we will now require that this additional information in the public key does not compromise the security of the scheme. Gentry reduces this security issue to the decisional version of the sparse subset-sum problem (SSSP), and hence the same assumption needs to be made in our situation. The SSSP problem is believed to take at least $\sqrt{\binom{s_1}{s_2}} > (s_1/s_2)^{s_2/2}$ steps to solve, assuming we are not in a low density subset sum, i.e. $s_1/\log p > 1$. If we take s_1 to be slightly greater than $\log p$, then we need to select s_2 such that

$$\sqrt{\binom{s_1}{s_2}} > 2^{N/\epsilon},$$

so as to ensure that the SSSP difficulty is at least as difficult as the difficulty of the BDDP underlying the somewhat homomorphic scheme.

6 Extension To Large Message Space

We now show that our scheme provides for a more powerful fully homomorphic scheme than that of Gentry. In [7] the fully homomorphic property can only be applied to single bit messages, since the Recrypt algorithm for full size messages is relatively complicated. We shall show we can obtain fully homomorphic encryption on N-bit messages and then discuss what this actually means.

First return to our basic scheme. We alter the KeyGen algorithm to output the whole polynomial $Z(x) = \sum_{i=0}^{N-1} z_i x^i$ modulo $2p$ as the secret key as opposed to the single term B. Let the resulting polynomial be denoted $B(x) = \sum_{i=0}^{N-1} b_i x^i$. Encryption is now modified to take any message from the space $\mathcal{B}^+_{\infty,N}(2)$, i.e. any binary polynomial of degree less than N. Decryption is then performed coefficient wise, namely each coefficient m_i of M is recovered by computing

$$m_i \leftarrow (c - \lfloor c \cdot b_i/p \rceil) \pmod 2.$$

It is easily seen that this modification results in a somewhat homomorphic scheme with the same multiplicative depth as the original scheme.

We now extend this somewhat homomorphic scheme to a fully homomorphic scheme. We write each coefficient of $B(x)$ as a different sum, over a different set of indices S_i,

$$\sum_{j \in S_i} B_{i,j} = b_i.$$

The secret key is now defined to be $\mathsf{sk}_{i,j} = 1$ if $j \in S_i$ and 0 otherwise. The Recrypt algorithm is then immediate. We first apply the Recrypt algorithm as

above, coefficient wise, to obtain new "cleaner" encryptions of each bit of the message, i.e. we obtain

$$c_{\mathsf{new}}^{(i)} = \mathsf{Encrypt}(m_i, \mathsf{PK}).$$

To obtain the encryption of the entire message we simply compute

$$c_{\mathsf{new}} = \mathsf{Encrypt}(m, \mathsf{PK}) = \sum_{i=0}^{N-1} c_{\mathsf{new}}^{(i)} \cdot \alpha^i \pmod{p}.$$

Note that recombining the different encryptions causes an extra increase in the error term with a factor of δ_∞. This increase in the error term is due to the multiplication, by α^i, of the error term underlying $c_{\mathsf{new}}^{(i)}$.

Hence, we can obtain fully homomorphic encryption with respect to the algebra $\mathbb{F}_2[x]/(F)$. To see the power of this we need to examine the algebra $\mathbb{F}_2[x]/(F)$. If $F(x)$ splits as $\prod_{i=1}^{t} f_i \pmod 2$ with f_i coprime and $\deg f_i = d_i$ then by the Chinese Remainder Theorem we have

$$F_2[x]/(F) \equiv \mathbb{F}_{2^{d_1}} \times \cdots \times \mathbb{F}_{2^{d_t}}.$$

By concentrating on a single component of the product on the right we therefore, by careful choice of F, obtain fully homomorphic encryption in any finite field of characteristic two of degree less than N. Furthermore, we could also obtain SIMD style homomorphic encryption in multiple finite fields of characteristic two at the same time.

7 Implementation Results

We now examine a practical instantiation of our scheme. We take the polynomial $F(x) = x^{2^n} + 1$, which is always irreducible over the integers. In particular our main parameter N is equal to 2^n, and we have $\delta_\infty = N$. We take $\eta = 2^{\sqrt{N}}$ and either $\mu = \sqrt{N}$ or $\mu = 2$. The case of $\eta = 2^{\sqrt{N}}$ and $\mu = \sqrt{N}$ are (for comparison) also the suggested parameter choices made in [7] (albeit in the 2-norm). The case of $\mu = 2$ is chosen to try to obtain as large a depth for the somewhat homomorphic scheme as possible.

Recall that if we write $\eta/(2 \cdot \sqrt{N} \cdot \mu) = 2^\epsilon$, then the security of our somewhat homomorphic scheme is assumed to be $2^{N/\epsilon}$. We then select $s_1 = \log p$ and s_2 to be such that

$$\sqrt{\binom{s_1}{s_2}} > 2^{N/\epsilon},$$

which ensures the difficulty of the SSSP is at least $2^{N/\epsilon}$. In addition, for our choice of $F(x)$, the expected multiplicative depth d for our somewhat homomorphic scheme, is estimated by

$$d \log 2 \leq \log \log \left(\frac{\eta}{2 \cdot \sqrt{N}} \right) - \log \log (N \cdot \mu).$$

We present the implications in the following table, for increasing values of n.

		$\mu = 2$			$\mu = \sqrt{N}$		
n	$\log_2 p$	$2^{N/\epsilon}$	s_2	d	$2^{N/\epsilon}$	s_2	d
8	4096	2^{25}	5	0.3	2^{36}	8	0.0
9	11585	2^{31}	6	0.8	2^{40}	7	0.3
10	32768	2^{41}	7	1.2	2^{48}	8	0.8
11	92681	2^{54}	8	1.7	2^{61}	9	1.2
12	262144	2^{73}	10	2.1	2^{80}	11	1.6
13	741455	2^{100}	12	2.5	2^{107}	13	2.1

In the Appendix we make a precise estimate for each value of s_2 what the corresponding Recrypt algorithm will produce in terms of the "dirtyness" of the ciphertext. This allows us to be able to estimate, for each value of s_2, the multiplicative depth $\hat{d}$ which would be required to obtain a fully homomorphic scheme. In the following table we present the value of $\hat{d}$ required. The given value includes the final and-gate to recombine two Recypted ciphertexts. We note that we only obtain a fully homomorphic scheme if $\hat{d} \leq d$, so we see that for practical values of n our scheme cannot be made fully homomorphic, although asymptotically it can be. In fact, in the Appendix we show that for $n \geq 27$ it is possible to obtain a fully homomorphic scheme. For a given fixed security level (and not the maximum possible for a given N and our choice of parameters), it should be possible to obtain a slightly lower n.

s_2	6	7	8	9	10	11	12	13
$\hat{d}$	7	7	7	8	8	8	8	8

The above estimates are very crude and we refer to the Appendix for a more detailed analysis.

Despite this problem with obtaining a fully homomorphic scheme, we timed the various algorithms for the somewhat homomorphic scheme on a desk-top machine using the NTL library: This was an x86-64 platform, and housed 2.4 GHz Intel Core2 (6600) processor cores and used the GCC 4.3.2 C compiler. We were unable to generate keys for the parameter size of $N = 2^{12}$, and smaller values of N key generation could take many hours. The problem with KeyGen being the need to compute many resultants and test the resulting number for primality. This is because the output of the resultant calculation will have $\log_2 p$ bits, so not only are we working with huge numbers; we also have little chance that this number is prime on any one iteration. A more general version of KeyGen would allow for non-prime, but squarefree resultants. But even in this case obtaining keys for say $n = 15$ seems daunting. We thus do not present times for the KeyGen algorithm. The times (in milli-seconds), and the actual value of d computed for the specific key, are presented in the following table:

				d	
n	Encrypt	Decrypt	Mult	$\mu = 2$	$\mu = \sqrt{N}$
8	4.2	0.2	0.2	1.0	0.0
9	38.8	0.3	0.2	1.5	1.0
10	386.4	0.6	0.4	2.0	1.0
11	3717.2	3.0	1.6	2.5	1.5

We see that in practice our scheme appears to obtain a better depth of decryption circuit than theory predicts, although still not deep enough to enable fully homomorphic encryption; at least at practical key sizes.

References

1. J. Buchmann. *Zur Komplexität der Berechungung von Einheiten und Klassenzahlen algebraischer Zahlkörper*, Habilitationsschrift, 1987.
2. J. Buchmann. A subexponential algorithm for the determination of class groups and regulators of algebraic number fields. *Séminaire de Théorie des Nombres – Paris 1988-89*, 27–41, 1990.
3. J. Buchmann, M. Maurer and B. Möller. Cryptography based on number fields with large regulator. *Journal de Théorie des Nombres de Bordeaux*, **12**, 293–307, 2000.
4. H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer GTM 138, 1993.
5. J. Ding and R. Lindner. Identifying ideal lattices. IACR eprint 2009/322.
6. J. Von Zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.
7. C. Gentry. Fully homomorphic encryption using ideal lattices. In *Symposium on Theory of Computing – STOC 2009*, ACM, 169–178, 2009.
8. C. Gentry. A fully homomorphic encryption scheme. Manuscript, 2009.
9. O. Goldreich, S. Goldwasser and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology – CRYPTO '97*, Springer-Verlag LNCS 1294, 112–131, 1997.
10. S. Hallgren. Fast quantum algorithms for computing the unit group and class group of a number field. In *Symposium on Theory of Computing – STOC 2005*, ACM 468–474, 2005.
11. J. Hoffstein, J. Pipher and J.H. Silverman. NTRU: a ring-based public key cryptosystem. *Algorithmic number theory – ANTS III*, Springer-Verlag LNCS 1423, 267–288, 1998.
12. A.K. Lenstra, H.W. Lenstra, Jr. and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Ann.*, **261**, 513-534, 1982.
13. P.Q. Nguyen and J. Stern The two faces of lattices in cryptology. In *Cryptography and Latticesm – CaLC 2001*, Springer-Verlag LNCS 2146, 146–180, 2001.
14. C. Thiel. *On the complexity of some problems in algorithmic algebraic number theory*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1995.
15. B.M.M. de Weger. *Algorithms for Diophantine Equations*. PhD thesis, University of Leiden, 1987.

A Analysis of the Recrypt Procedure

In this appendix we explain exactly how Gentry's re-encryption circuit is implemented in the context of our scheme. We first decrease the size of $\mathsf{r}_{\mathsf{Dec}}$ by a factor of two to ensure that the floating point number obtained in the decryption procedure is within $1/4$ of an integer, i.e. we know that

$$c \cdot B/p \in]x - 1/4, x + 1/4[\,,$$

for some integer x. Since we are only interested in the result modulo 2, we can actually compute

$$\left\lfloor \sum_{i=1}^{s_1} \mathsf{sk}_i (c \cdot B_i \bmod 2p)/p \right\rceil \pmod 2 .$$

As such we are adding up a subset of s_2 values out of s_1 floating point values, all of which lie in the range $[0, \ldots, 2)$.

To keep the Recryption method manageable we need to minimize the precision of the floating point numbers $(c \cdot B_i \bmod 2p)/p$ that we work with. First note that if we truncate these values to a fixed precision and make a maximum error of $\leq 1/2$, then we can still recover the correct result since the approximated sum will be in the interval $]x - 3/4, x + 1/4[$, and any number in this interval determines x uniquely. More precisely, if we obtain bits e_0, e_1 and e_2 such that the sum computed with fixed precision is given by

$$e_0 + 2^{-1} \cdot e_1 + 2^{-2} \cdot e_2 + \cdots ,$$

then the final output is given by

$$(e_0 + e_1 + e_2 + e_1 \cdot e_2) \pmod 2.$$

The above equation is derived from examining the four possible cases corresponding to the values of e_1 and e_2;

e_1	e_2	Output
0	0	e_0
1	0	$e_0 + 1 \pmod 2$
0	1	$e_0 + 1 \pmod 2$
1	1	$e_0 + 1 \pmod 2$

Assume we work with t bits of precision, i.e. each floating point number in $[0, \ldots, 2)$ is represented as $\sum_{i=0}^{t-1} e_i 2^{-i}$. Then since only s_2 numbers are non-zero, the maximum error in the total sum is given by

$$s_2 \sum_{i=t}^{\infty} 2^{-i} = s_2 2^{-t+1} .$$

As such we need to choose t such that $s_2 \cdot 2^{-t+1} \leq 1/2$ which implies that $t = \lceil \log_2 s_2 \rceil + 2$. We also define s to be the number of bits to represent all integers up to s_2, i.e. $s = \lfloor \log_2 s_2 \rfloor + 1$. To get some idea of the practical implications of these two values in what follows we give the following table:

s_2	s	t		s_2	s	t
5	3	5		10	4	6
6	3	5		11	4	6
7	3	5		12	4	6
8	4	5		13	4	6
9	4	6		14	4	6

So we see that the value of s is essentially either 3 or 4, and t is either 5 or 6.

The algorithm re-encryption takes as input a ciphertext c and a public key $\mathsf{PK} = (p, \alpha, s_1, s_2, \{\mathfrak{c}_i, B_i\}_{i=1}^{s_1})$ and consists of the following distinct phases:

1. Write down the first t bits of the s_1 floating point numbers $(c \cdot B_i \bmod 2p)/p$ as an $s_1 \times t$ matrix $(b_{i,j})$ for $i = 1, \ldots, s_1$ and $j = 1, \ldots, t$.
2. Encrypt each of the bits $b_{i,j}$ under the public key PK to obtain an $s_1 \times t$ matrix of clean ciphertexts $(c_{i,j})$.
3. Multiply each row of the matrix by the corresponding encryption $\mathfrak{c}_i$ of sk_i to obtain $(\mathfrak{c}_i \cdot c_{i,j}) \bmod p$. As such we obtain the encryption of a matrix with only s_2 non-zero rows.
4. Compute the sum of each column as the Hamming weight using symmetric polynomials and hence reduce the sum of s_1 floating point values to the sum of t floating point values of t bits of precision. More precisely, denote by $h_{i,j}$ the j-th bit of the Hamming weight of the i-th column for $i = 1, \ldots, t$ and $j = 1, \ldots, s$ and form the $t \times t$ matrix $(H_{i,j})$ with $H_{i,j} = h_{i,i-j+s}$ whenever the right hand side is defined and zero otherwise.
5. Merge rows of the matrix H, so as to obtain an $s \times t$ matrix H' such that the sum of the rows of H' equals the sum of the rows of H.
6. Apply carry-save-adders to progressively reduce the matrix to one with two rows. Each set of three rows is reduced to two, and then this procedure is repeated.
7. Perform the final addition, and output the encryption of a single bit.

It is perhaps worth recalling that for our scheme we have $s_1 \approx \log_2 p$ and s_2 is chosen so that $\sqrt{\binom{s_1}{s_2}} \geq 2^\tau$ for our required security level τ, which itself defines the parameter N. The value p is approximately equal to $2^{N \cdot \sqrt{N}}$, thus s_1 is very large indeed. Ciphertexts are integers modulo p and each valid decryptable ciphertext can be considered to lie in ball of a given radius. A "clean" ciphertext lies in a ball of radius $\mu + 1$ and as we add/multiply ciphertexts this radius increases, with the ciphertext becoming increasingly "dirtier". Recall that we have the following behaviour: Let c_1 and c_2 denote two ciphertexts, corresponding to two randomizations $C_1(x) = M_1 + N_1(x)$ and $C_2(x) = M_2 + N_2(x)$; where $M_i \in \{0, 1\}$ are the messages and $N_i(x) \in \mathcal{B}_{\infty,N}(r_i - 1)$ is the randomness, i.e. $C_i(x) \in \mathcal{B}_{\infty,N}(r_i)$. For a ciphertext c, denote with $\mathsf{rad}(c)$ the radius of the ball containing the corresponding polynomial $C(x)$, i.e. we have $C(x) \in \mathcal{B}_{\infty,N}(\mathsf{rad}(c))$. Then

$$\begin{aligned} \mathsf{rad}(c_1 + c_2) &= \mathsf{rad}(c_1) + \mathsf{rad}(c_2), \\ \mathsf{rad}(c_1 \cdot c_2) &= \delta_\infty \cdot \mathsf{rad}(c_1) \cdot \mathsf{rad}(c_2). \end{aligned}$$

We will now analyze the growth of the error terms during each of the phases of the re-encryption. Recall that for our choice of parameters we have $\mu = \sqrt{N}$, $\delta_\infty = N$ and $s_1 = N\sqrt{N}$. Therefore define $\rho = \sqrt{N}$, then we will compute explicit expressions for the radii of the ciphertexts as a function of ρ. Recall that

the notation $f \sim g$ means that $\lim_{\rho\to\infty} f/g = 1$. If A is a matrix of ciphertexts we let $\mathsf{rad}(A)$ denote the matrix obtained by applying rad to each entry in A.

Stages 1 and 2

The result of the first two stages is that we obtain an $s_1 \times t$ matrix A_2 containing clean ciphertexts $c_{i,j}$ with $\mathsf{rad}(c_{i,j}) = \mu + 1 := r_2 \sim \rho$.

Stage 3

Here we take the clean encryptions of the values of sk_i and multiply through each corresponding row to obtain a matrix A_3, with $(\mathsf{rad}(A_3))_{i,j} = \delta_\infty \cdot r_2^2 := r_3 \sim \rho^4$ for all i, j.

Stage 4

In this stage, we need to compute the encryption of the sum of the (plaintext) bits $\mathsf{sk}_i \cdot b_{i,j}$ in each of the columns seperately. Note that the sum is simply the Hamming weight of the column, so it suffices to compute the bits of the Hamming weight. Furthermore, since only s_2 entries in each column are one, the number of bits in the Hamming weight is bounded by s. We let $\mathrm{SymPol}_i(x_1, \ldots, x_k)$ denote the i-th symmetric polynomial on the variables $x_1, \ldots, x_k$. Then the bits of the Hamming weight of the bit vector $(b_1, \ldots, b_k)$ is given by

$$(\mathrm{SymPol}_{2^{s-1}}(b_1, \ldots, b_k) \pmod 2,\ \ldots,\ \mathrm{SymPol}_{2^0}(b_1, \ldots, b_k) \pmod 2)\,.$$

So for each column of our matrix A_3 we need to compute all the symmetric polynomials up to $S = 2^{s-1}$. To compute the S symmetric polynomials on (the encryptions of) s_1 bits we proceed in a recursive manner, essentially using Horner's Rule to compute the last $S+1$ coefficients of the product $\prod_{i=1}^{s_1}(b_i \cdot x + 1)$.

For each column of A_3 we execute the following function to compute the (encryptions) of the bits of the Hamming weight of the j-th column for $j = 1, \ldots, t$:

- Set $(\mathfrak{s}_1, \ldots, \mathfrak{s}_S) \leftarrow (0, \ldots, 0)$.
- For $i = 1, \ldots, s_1$ do
  - For $k = \min(i, S), \ldots, 3, 2$ do
    - $\mathfrak{s}_k \leftarrow \mathfrak{s}_k + \mathfrak{s}_{k-1} \cdot A_3(i, j) \pmod p$
  - $\mathfrak{s}_1 \leftarrow \mathfrak{s}_1 + A_3(i, j) \pmod p$.
- Return $(\mathfrak{s}_1, \ldots, \mathfrak{s}_S)$.

We can also see by analysing the above algorithm how dirty the ciphertexts will become. To produce $\mathfrak{s}_i$ we need to sum $\binom{s_1}{i}$ terms which consist of the multiplication of i of the ciphertexts in A_3 together. If $c_1, \ldots, c_8$ are eight ciphertexts given by entries in A_3 then we have

$$\mathsf{rad}(c_1) \sim \rho^4, \quad \mathsf{rad}(c_1 \cdot c_2) \sim \rho^{10}, \quad \mathsf{rad}(c_1 \cdots c_4) \sim \rho^{22}, \quad \mathsf{rad}(c_1 \cdots c_8) \sim \rho^{46}.$$

Thus we have

$$\mathsf{rad}(\mathfrak{s}_1) \sim s_1 \cdot \rho^4 \sim \rho^7, \qquad \mathsf{rad}(\mathfrak{s}_2) \sim \binom{s_1}{2} \cdot \rho^{10} \sim \rho^{16}/2,$$
$$\mathsf{rad}(\mathfrak{s}_4) \sim \binom{s_1}{4} \cdot \rho^{22} \sim \rho^{34}/4!, \qquad \mathsf{rad}(\mathfrak{s}_8) \sim \binom{s_1}{8} \cdot \rho^{46} \sim \rho^{70}/8!.$$

Given the bits $h_{i,j}$ for $i = 1, \ldots, t$ and $j = 1, \ldots, s$ of the t Hamming weights of the t columns, the sum of the resulting floating point numbers represented by the rows of A_3 is given by

$$\sum_{i=1}^{t} \sum_{j=1}^{s} 2^{j-i} h_{i,j} \,.$$

Since we are only interested in the sum modulo 2, we can see that the above sum modulo 2 corresponds to the sum of the rows of the $t \times t$ matrix H with $H_{i,j} = h_{i,i-j+s}$ whenever the right hand side is defined and zero otherwise. We therefore obtain the following matrices H depending on the combination of s and t.

Case $(s,t) = (3,5)$:
We find

$$\mathsf{rad}(H) \sim \begin{pmatrix} \rho^7 & 0 & 0 & 0 & 0 \\ \rho^{16}/2 & \rho^7 & 0 & 0 & 0 \\ \rho^{34}/4! & \rho^{16}/2 & \rho^7 & 0 & 0 \\ 0 & \rho^{34}/4! & \rho^{16}/2 & \rho^7 & 0 \\ 0 & 0 & \rho^{34}/4! & \rho^{16}/2 & \rho^7 \end{pmatrix}$$

Case $(s,t) = (4,5)$:
We find

$$\mathsf{rad}(H) \sim \begin{pmatrix} \rho^7 & 0 & 0 & 0 & 0 \\ \rho^{16}/2 & \rho^7 & 0 & 0 & 0 \\ \rho^{34}/4! & \rho^{16}/2 & \rho^7 & 0 & 0 \\ \rho^{70}/8! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 & 0 \\ 0 & \rho^{70}/8! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 \end{pmatrix}$$

Case $(s,t) = (4,6)$:
We find

$$\mathsf{rad}(H) \sim \begin{pmatrix} \rho^7 & 0 & 0 & 0 & 0 & 0 \\ \rho^{16}/2 & \rho^7 & 0 & 0 & 0 & 0 \\ \rho^{34}/4! & \rho^{16}/2 & \rho^7 & 0 & 0 & 0 \\ \rho^{70}/8! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 & 0 & 0 \\ 0 & \rho^{70}/8! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 & 0 \\ 0 & 0 & \rho^{70}/8! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 \end{pmatrix}$$

Stage 5

We now notice that the entries in each column can be permuted around independently. It turns out that it makes more sense, due to the way we will add up the columns in Stage 6, to order the column entries so that the dirtyness increases as you descend a column. This also allows us to delete rows which consist entirely of zeros. We notice that the resulting matrix H' will be of size $s \times t$. In the three examples we do not give the precise permutation of the columns, as this can be deduced from the implied permutation of the d values.

Case $(s,t) = (3,5)$:
We find

$$\mathsf{rad}(H') \sim \begin{pmatrix} \rho^7 & \rho^7 & \rho^7 & 0 & 0 \\ \rho^{16}/2 & \rho^{16}/2 & \rho^{16}/2 & \rho^7 & 0 \\ \rho^{34}/4! & \rho^{34}/4! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 \end{pmatrix}$$

Case $(s,t) = (4,5)$:
We find

$$\mathsf{rad}(H') \sim \begin{pmatrix} \rho^7 & \rho^7 & 0 & 0 & 0 \\ \rho^{16}/2 & \rho^{16}/2 & \rho^7 & 0 & 0 \\ \rho^{34}/4! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 & 0 \\ \rho^{70}/8! & \rho^{70}/8! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 \end{pmatrix}$$

Case $(s,t) = (4,6)$:
We find

$$\mathsf{rad}(H') \sim \begin{pmatrix} \rho^7 & \rho^7 & \rho^7 & 0 & 0 & 0 \\ \rho^{16}/2 & \rho^{16}/2 & \rho^{16}/2 & \rho^7 & 0 & 0 \\ \rho^{34}/4! & \rho^{34}/4! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 & 0 \\ \rho^{70}/8! & \rho^{70}/8! & \rho^{70}/8! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 \end{pmatrix}$$

Stage 6

We now apply a sequence of carry-save-adders to reduce the number of rows down to two. We first apply a single chain of carry-save-adders to add the bits in the first three rows of matrix H' which produces two rows as output, where the first row simply contains the exor of the three rows and the second row contains the sum of all products of two out of three rows. Note, we ignore any overflow into the bit position corresponding to the binary weight 2^1 and above. If H' has four rows we then append the fourth row to the result and apply another chain of carry-save-adders. At the end of this stage we have a matrix A_5 of dimension $2 \times t$.

From the above estimates for $\mathsf{rad}(H')$ we can then derive the following estimates for $\mathsf{rad}(A_5)$:

Case $(s,t) = (3,5)$:
We find

$$\mathsf{rad}(A_5) \sim \begin{pmatrix} \rho^{34}/4! & \rho^{34}/4! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 \\ \rho^{52}/48 & \rho^{52}/48 & \rho^{25}/2 & 0 & 0 \end{pmatrix}.$$

Case $(s,t) = (4,5)$:
We find

$$\mathsf{rad}(A_5) \sim \begin{pmatrix} \rho^{70}/8! & \rho^{70}/8! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 \\ \rho^{106}/4! \cdot 8! & \rho^{52}/2 \cdot 4! & \rho^{25}/2 & 0 & 0 \end{pmatrix}.$$

Case $(s,t) = (4,6)$:
We find

$$\mathsf{rad}(A_5) \sim \begin{pmatrix} \rho^{70}/8! & \rho^{70}/8! & \rho^{70}/8! & \rho^{34}/4! & \rho^{16}/2 & \rho^7 \\ \rho^{124}/2 \cdot 4! \cdot 8! & \rho^{106}/4! \cdot 8! & \rho^{52}/2 \cdot 4! & \rho^{25}/2 & 0 & 0 \end{pmatrix}.$$

A.1 Stage 7

The final stage is to add the two remaining rows together, and then use the analysis from earlier. More precisely, as before we write the final output as

$$e_0 + 2^{-1} \cdot e_1 + 2^{-2} \cdot e_2 + \cdots,$$

where, for $i = t-1, \ldots, 0$,

$$\begin{aligned} e_i &= (A_5)_{1,i} + (A_5)_{2,i} + c_{i+1}, \\ c_i &= ((A_5)_{1,i} + (A_5)_{2,i}) \cdot c_{i+1} + (A_5)_{1,i} \cdot (A_5)_{2,i}. \end{aligned}$$

where $c_t = 0$. Note that the last two elements on the final row of the above matrices are equal to zero, so there are no carries to worry about from these columns. This simplifies the above expressions a little. We then obtain in each of our cases:

- **Case** $(s,t) = (3,5)$:

$$\mathsf{rad}(e_0, e_1, e_2, e_3, e_4) = \left(\frac{\rho^{115}}{(2 \cdot 4!)^2}, \frac{\rho^{61}}{2 \cdot 4!}, \frac{\rho^{34}}{4!}, \frac{\rho^{16}}{2}, \rho^7 \right)$$

- **Case** $(s,t) = (4,5)$:

$$\mathsf{rad}(e_0, e_1, e_2, e_3, e_4) = \left(\frac{\rho^{133}}{(2 \cdot 4! \cdot 8!)}, \frac{\rho^{70}}{8!}, \frac{\rho^{34}}{4!}, \frac{\rho^{16}}{2}, \rho^7 \right)$$

- **Case** $(s,t) = (4,6)$:

$$\mathsf{rad}(e_0, e_1, e_2, e_3, e_5) = \left(\frac{\rho^{241}}{2(4! \cdot 8!)^2}, \frac{\rho^{133}}{(2 \cdot 4! \cdot 8!)}, \frac{\rho^{70}}{8!}, \frac{\rho^{34}}{4!}, \frac{\rho^{16}}{2}, \rho^7 \right)$$

As discussed earlier we can obtain the result Res of the Recryption procedure from the values of e_0, e_1 and e_2, by computing the expression $(e_0 + e_1 + e_2 + e_1 \cdot e_2) \pmod 2$. This enables us to determine the value of $\mathsf{rad}(\mathrm{Res})$ in our three cases as follows:

(s,t)	$(3,5)$	$(4,5)$	$(4,6)$
$\mathsf{rad}(\mathrm{Res})$	$\dfrac{\rho^{115}}{(2\cdot 4!)^2}$	$\dfrac{\rho^{133}}{(2\cdot 4!\cdot 8!)}$	$\dfrac{\rho^{241}}{2(4!\cdot 8!)^2}$

So using the above method we can Recrypt a ciphertext to obtain a new ciphertext whose dirtyness measure is bounded by the radius in the above table. We then operate on this ciphertext by applying an addition or a multiplication with another similar ciphertext so as to produce a new ciphertext which we then apply the Recrypt procedure to again. For this to work we require that the ciphertext before Recryption can itself be validly decrypted. This means that we need to be able to decrypt a ciphertext with dirtyness measure given by $\delta_\infty \cdot \mathsf{rad}(\mathrm{Res})^2$.

In the following table we present the final outcome. For a specific value of s_2 we give the values of (s,t) in the algorithm, then we give the value of $\mathsf{rad}(c)$ which needs to be able to be decrypted to obtain fully homomorphic encryption, and then the corresponding minimum value of the "depth" of the circuit. We note that this measure of depth is a very crude estimate since it measures the number of multiplications in a perfectly balanced circuit consisting solely of multiplications, whereas our measure $\mathsf{rad}(c)$ is much more precise.

s_2	(s,t)	$\mathsf{rad}(c)$	depth
$5,6,7$	$(3,5)$	$\dfrac{\rho^{232}}{(2\cdot 4!)^4}$	7
8	$(4,5)$	$\dfrac{\rho^{268}}{(2\cdot 4!\cdot 8!)^2}$	7
$9,10,11,12,13,14$	$(4,6)$	$\dfrac{\rho^{484}}{2^2(4!\cdot 8!)^4}$	8

Recall that the original $\mathsf{r}_{\mathsf{Dec}}$ is given by $2^{\sqrt{N}}/2\sqrt{N}$ and thus equal to $2^\rho/2\rho$. To obtain a fully homomorphic encryption scheme we therefore require that

$$\mathsf{rad}(c) < \frac{2^\rho}{4\rho}\,,$$

where the extra factor of 2 comes from the fact that we made $\mathsf{r}_{\mathsf{Dec}}$ smaller by a factor of 2. It is easy to see that this bound is not attained for the practical parameter sizes given in Section 7. A complete similar analysis for the case $(s,t) = (5,7)$ gives a radius $\mathsf{rad}(c)$ of

$$\mathsf{rad}(c) = \frac{\rho^{880}}{(8!)^2\cdot(4!\cdot 16!)^4}$$

which shows that for $\rho \geq 11680$ it is possible to obtain a fully homomorphic encryption scheme. This corresponds to $N \geq 136422400$ or thus $n \simeq 27$.