Fixed Point Arithmetic in SHE Schemes

A. Costache[1], N.P. Smart[1], S. Vivek[1], and A. Waller[2]

[1] University of Bristol, UK
[2] Thales Research, Reading, UK.

Abstract. The purpose of this paper is to investigate fixed point arithmetic in ring-based Somewhat Homomorphic Encryption (SHE) schemes. We provide three main contributions: Firstly, we investigate the representation of fixed point numbers. We analyse the two representations from Dowlin et al, representing a fixed point number as a large integer (encoded as a scaled polynomial) versus a polynomial-based fractional representation. We show that these two are, in fact, isomorphic by presenting an explicit isomorphism between the two that enables us to map the parameters from one representation to another. Secondly, given a computation and a bound on the fixed point numbers used as inputs and scalars within the computation, we achieve a way of producing lower bounds on the plaintext modulus p and the degree of the ring d needed to support complex homomorphic operations. Finally, we investigate an application in homomorphic image processing. We have an image given in encrypted form and are required to perform the standard image processing pipeline of Fourier Transform–Hadamard Product–Inverse Fourier Transform. In particular we examine applications in which the specific matrices involved in the Hadamard multiplication are also encrypted. We propose a mixed Fourier Transform Algorithm which aims to strike a compromise between the number of homomorphic multiplications and the parameter sizes of the underlying SHE scheme.

1 Introduction

The efficiency of Somewhat Homomorphic Encryption (SHE) schemes has improved dramatically in the seven years since their discovery by Gentry in 2009 [7]. The main effort in research now is to obtain practical schemes for a given class of interesting functions; since practical Fully Homomorphic Encryption seems out of reach using existing techniques.

When proposing to use SHE schemes in an application a key issue is how to map the data types of the application to the supported data types of the SHE scheme. Most theoretical treatments consider SHE schemes which work over bits, and the application is assumed to be the evaluation of some binary circuit. In practice this is likely to be very costly, and so some authors have considered other scenarios in which the computations are performed over arithmetic circuits or polynomial rings [6, 8, 10].

At their heart almost all SHE schemes make use of a plaintext space R_p, which is the reduction modulo p of a polynomial ring over the integers R. We shall refer to p as the plaintext modulus, which is often selected to be a prime. The ring is frequently selected to be the ring of integers of a cyclotomic number field; i.e.

$$R = \mathbb{Z}[X]/\Phi_m(X).$$

In considering an application one has a number of factors to balance; first the SHE multiplicative depth of the functions which can be evaluated; secondly the plaintext modulus p and thirdly the security level required. These all imply bounds on the degree of the ring one is using; and hence the efficiency of the application[3]. Of importance in what follows is that a SHE scheme has a maximum multiplicative depth bounding what can be evaluated. In practice this consists of a number of levels, where each ciphertext is associated to a specific level. Multiplication of ciphertexts at levels i_0 and i_1 results in a ciphertext at level $\max(i_0, i_1) + 1$; whereas scalar multiplication is equivalent to the addition of roughly half a level. Once the maximum level is obtained, no further homomorphic operations are possible.

The first obvious method to move away from binary circuits is to consider plaintext moduli other than $p = 2$, and hence to evaluate arithmetic circuits. Indeed the first application of SHE schemes to obtain an efficiency improvement upon other technologies did precisely this; for example the use of large plaintext moduli p in the SPDZ protocol [5]. However, using arithmetic circuits is also limited. For example, suppose one wished to perform integer arithmetic. In that case, naively increasing p to a large enough value to cope with the largest integer the application could obtain would impose considerable performance penalties.

One can think of using a large plaintext modulus p as using a plaintext space which is long and thin. Some authors have tried to balance the choice of p and the degree d of the ring R to obtain more efficient representation of integers, akin to a more short and fat plaintext space [10]. A problem overlooked by many authors is to how to select p and d to enable such a plaintext representation of integer valued payloads; and in particular to bound p and d as a complex homomorphic operation is performed. This is the first problem we consider in this paper. Given a computation on integers, and a bound on the input integers, we are able to produce lower bounds on p and d needed to support such a homomorphic calculation. Our main general technical contribution is to derive such lower bounds on p and d.

Given an ability to process plaintext messages consisting of large integers the next task is to process fixed-point numbers. A number of authors have considered methodologies for this, most notably Dowlin et al [6]. Dowlin et al present two efficient methods to represent fixed-point numbers. In the first they encode a fixed point number as a scaled integer (which they then encode as a polynomial), whilst in the second they utilize a fractional representation (also based on polynomials). The advantage of the former method is that it is easier to analyse and it can be applied for any polynomial plaintext ring R_p. However, it also

[3] In this paper we will ignore issues such as SIMD operations obtained by selecting p and m in an special manner, see [12, 4, 8] for details

requires complex bookkeeping of the homomorphic ciphertexts during a calculation to ensure that the fixed-point numbers are correctly scaled. The fractional representation avoids such bookkeeping, but it appears harder to analyse so as to derive parameters which will support the homomorphic operations. Further, it requires R to be selected to be a cyclotomic ring $Z[X]/\Phi_m(X)$, where m is a power of two. We show show that the two representations are in fact isomorphic, when used with the same power of two cyclotomic ring; we present a concrete isomorphism between the two underlying rings and hence are able to map our parameters from the first representation to the second.

Our third contribution is to analyse a relatively complex but useful fixed-point algorithm namely the Fast Fourier Transform (FFT). This is needed to perform applications such as homomorphic image processing. When examining our fixed point algorithms for addition and multiplication it will be immediately seen that one needs to consider the homomorphic levels which a given calculation will consume. Additionally, one must also consider how much the fixed point calculation increases the demands on the plaintext space, with repeated scalar multiplication being particularly costly. This is particularly interesting for the FFT algorithm, since at its heart the FFT algorithm is a linear operation performed in a recursive manner (with an FFT of size n reduced to two FFTs of size $n/2$). This recursion decreases the *number* of scalar multiplications needed, but increases the *depth* of the scalar multiplications needed. The naive Fourier Transform is also a linear operation, but it consists of only scalar multiplications of depth one. Thus one has a trade off between reducing the number of operations against the required depth.

Previous authors have examined homomorphic evaluation of the Fourier transform [2, 3]. Indeed by exploiting the linear nature of the calculation they utilized an encoding of fixed point numbers via scaled integers. Then they used the additively homomorphic Paillier encryption algorithm to perform the homomorphic evaluation of the Fourier transform. This has a number of disadvantages. Firstly by encoding in a purely integer manner the Paillier plaintext modulus space N increases dramatically if one is to perform an FFT, followed by a linear map, followed by an inverse FFT. In addition it requires *all* homomorphic operations in an application to be linear.

We consider a standard image processing pipeline of FFT, followed by application of some operation in the Fourier domain, followed by an inverse FFT. The use of additively homomorphic encryption would imply that the operation in the Fourier domain would also need to be linear. Even the simplest form of such processing (Hadamard multiplication by a given matrix) would require that the matrix be public and not also encrypted.

Thus in the third part of this paper we consider the homomorphic evaluation of such an image processing pipeline using Hadamard multiplication by an encrypted matrix as our main processing step. We examine the resulting homomorphic algorithms, given bounds on the plaintext spaces derived from our earlier analysis, and present runtimes obtained from an implementation using the using the HELib library [9]. Whilst we are not able to process large images in the encrypted domain, one notes that processing of tiny (32×32 pixel) images have found application in some domains, e.g. [13]. In addition even when processing large images, they are often divided into smaller patches during the processing pipeline.

2 Integer Arithmetic

We first consider the simpler case of integer arithmetic; it will turn out that once this is solved fixed point arithmetic can be built on top of the integer arithmetic. We wish to process an arithmetic circuit *over the integers* where the input encrypted integers, and scalars, come from multiple ranges $[-L_i, \ldots, L_i]$. Allowing different ranges for different inputs and scalars will result more accurate bounds when we come to consider the FFT algorithm later. Clearly as the circuit is computed the bound on the size of the integers increases, and it is this growth in size which we need to deal with if we are to be able to cope with integers encrypted via our SHE scheme.

As a warm up we consider the simpler case where we we wish to compute a "regular" integer circuit which consists of at most $A \geq 0$ additions at each "level" in the circuit, and then, at each level, a layer of multiplications are performed. The multiplicative depth of the circuit will be denoted by $M \geq 1$. In addition to simplify this initial discussion we assume all scalars and inputs are in the same range, i.e. we fix $L_i = L$ for all i. Clearly the output values from such a circuit will have absolute value bounded by

$$L_{max}^{A,M} := \left(2^{\sum_{i=1}^{M} 2^i \cdot A}\right) \cdot L^{2^M} = 2^{A\left(2^{M+1}-2\right)} \cdot L^{2^M}. \tag{1}$$

As explained in the introduction, natively the SHE scheme will encrypt polynomials modulo p, with degree bounded by d. The obvious natural encoding for integers is the *scalar encoding* method. In this encoding method an integer is encoded as the constant polynomial, then integer addition and multiplication become addition and multiplication modulo p. To ensure correctness we then require than $p > 2 \cdot L_{max}^{A,M}$, and hence p has to be very large indeed. This would make the SHE scheme highly inefficient, even for very low depth circuits.

2.1 Representing Integers As Polynomials

This led some authors, e.g. [10], to introduce the following method of encoding an integer, which we call the *non-balanced base-B encoding* method. We encode integers as an integer polynomial in base B, for some base value B to be determined. The polynomial will have negative coefficients for negative integers, and positive coefficients for positive integers. Thus we encode the integer as a polynomial with coefficients in the range $[-(B-1), \ldots, (B-1)]$. In particular this implies an integer in the range $[-L_i, \ldots, L_i]$ on input is encoded as a polynomial of degree at most

$$d_i^{\mathsf{non-Bal}} = \lfloor \log L_i / \log B \rfloor.$$

We are interested in how the infinity norm, and degree, of the polynomials increases as we pass through the circuit. Where for a polynomial $P(X) = p_0 + p_1 \cdot X + \cdots p_d \cdot X^d$ we have $\|P\|_\infty = \max_{i=0,\ldots,d} |p_i|$. Thus for this input/scalar integer at circuit level 0 the infinity norm of our polynomials is bounded by $B_{i,0}^{\mathsf{non-Bal}} = B - 1$.

Another method, considered in [6], is the *balanced base-B encoding*. The integer is now encoded as a polynomial with coefficients in the range $[-(B-1)/2, \ldots, (B-1)/2]$ for an odd integer $B \geq 3$. Any polynomial can now have both non-negative and negative coefficients. This method overcomes a limitation of the previous method that wasted part of the plaintext space by allowing only polynomials with coefficients of the same sign. At level 0, our integer is encoded as a polynomial of degree at most

$$d_i^{\mathsf{Bal}} = \lceil \log(2 \cdot L_i + 1)/\log B \rceil - 1. \tag{2}$$

the infinity norm of our input polynomials is bounded by $B_{i,0}^{\mathsf{Bal}} = (B-1)/2$.

In a later section we outline how to obtain bounds on the degree and infinity norm of the polynomials as we perform a calculation via an integer circuit. It will turn out that the optimal choice in the above two polynomial representations is to use the balanced base $B = 3$ representation, so in particular we select $B_{i,0}^{\mathsf{Bal}} = 1$ for the rest of this paper.

3 Fixed Point Arithmetic

We present two encoding methods for fixed point arithmetic, introduced in [6]. We then show that these two representations are isomorphic. To illustrate the techniques, we will use the two fixed numbers below throughout

$$y = 6.370370\ldots = \frac{172}{27} \text{ and } y' = 2.6666666\ldots = \frac{8}{3},$$

which in balanced base $B = 3$ representation are given by

$$y = 1\bar{1}0.101 \text{ and } y' = 10.\bar{1},$$

where $\bar{1} = -1$.

3.1 Balanced Base-B Encoding

Our first method represents a fixed point number as two integers, one representing the number and the one representing by which power of B one needs to decode. Thus this method requires a level of book keeping in order to keep track of the second integer. Let y be a real fixed-point number, and denote by $y = y^+.y^-$ its integer and fractional parts in balanced base-B representation. We then let I^+ be one less than the number of integer digits and I^- be equal to the number of fractional digits; thus we can write

$$y^+ = b_{I^+} \cdot B^{I^+} + b_{I^+-1} \cdot B^{I^+-1} + \cdots + b_1 \cdot B + b_0,$$
$$y^- = b_{-I^-} \cdot B^{-I^-} + b_{-I^-+1} \cdot B^{-I^-+1} + \cdots + b_{-2} \cdot B^{-2} + b_{-1} \cdot B^{-1}$$

where $b_i \in [-(B-1)/2, \ldots, (B-1)/2]$. Thus we can express y as

$$y = \sum_{i=-I^-}^{I^+} b_i \cdot B^i.$$

We then represent y as the pair of integers $(y \cdot B^{I^-}, I^-) = (\hat{y}, i)$. The integer $\hat{y}$ can then be represented by a polynomial $q(X)$, by replacing B in the above expression by X, to obtain the final representation (q, i). Thus we have

$$\begin{aligned} q_0(X) &= b_{I^+} \cdot X^{I^+} + b_{I^+-1} \cdot X^{I^+-1} + \cdots + b_1 \cdot X + b_0, \\ q_1(X) &= b_{-I^-} + b_{-I^-+1} \cdot X + \cdots + b_{-2} \cdot X^{I^--2} + b_{-1} \cdot X^{I^--1}, \\ q(X) &= q_0(X) \cdot X^i + q_1(X). \end{aligned}$$

The degree of the polynomial $q(X)$ above is $\deg(q) = I^- + I^+$, and to recover the fixed point number y from a pair (q, i) we compute

$$y = q(B) \cdot B^{-i}.$$

For our two example fixed point numbers above we have $y \equiv (q, i)$ and $y' \equiv (q', i')$ where $i = 3$ and $i' = 1$ and

$$\begin{aligned} q(X) &= (X^2 - X) \cdot X^3 + (X^2 + 1) = X^5 - X^4 + X^2 + 1, \\ q'(X) &= X \cdot X - 1 = X^2 - 1. \end{aligned}$$

Given this encoding we can now define how to perform basic arithmetic on the encoding.

Addition: Suppose we have two pairs (q, i) and (q', i') encoding the fixed point numbers, y and y' respectively. Write them as above, namely $q(X) = q_0(X) \cdot X^i + q_1(X)$ and similarly for $q'(X)$. Now if $i \neq i'$, this means that the encodings are not at the same "fixed-point level"[4] and thus the numbers they represent are expressed with a different number of significant digits. Thus, before adding two encodings we must ensure that they are at the same level, by multiplying one by a suitable power of X. Thus if we let $I = \max(i, i')$, we have that $(q, i) + (q', i') = (Q, I)$, where

$$(Q, I) = \begin{cases} (q + q' \cdot X^{I-i'}, i) & \text{if } i > i' \\ (q' + q \cdot X^{I-i}, i') & \text{if } i' \geq i. \end{cases}$$

[4] Note to be confused with the associated level in the SHE scheme once we encrypt the polynomial.

To see this indeed corresponds to fixed point multiplication, notice that, assuming $i \leq i'$, that

$$\begin{aligned} Q(B) \cdot B^{-I} &= (q + q' \cdot B^{I-i'}) \cdot B^{-I} \\ &= q \cdot B^{-I} + q' \cdot B^{I-i'} \cdot B^{-I} \\ &= q \cdot B^{-i} + q' \cdot B^{-i'} \\ &= y + y'. \end{aligned}$$

For our two example numbers we have, $i = 3 > i' = 1$, that

$$Q = q + q' \cdot X^2 = (X^5 - X^4 + X^2 + 1) + (X^2 - 1) \cdot X^2 \qquad = X^5 + 1,$$

and $I = \max(3, 1) = 3$. To check correctness, notice that $Q(B) \cdot B^{-3} = B^2 + B^-3 = 9 + 1/27 = 9.037037$ as required.

Multiplication: Multiplication is more straightforward, we simply perform

$$(q, i) \cdot (q', i') = (q \cdot q', i + i') = (Q, I),$$

with correctness being obvious. For our two example fixed point numbers we have the product representation being given by

$$\begin{aligned} Q &= (X^5 - X^4 + X^2 + 1) \cdot (X^2 - 1) \\ &= X^7 - X^6 - X^5 + 2 \cdot X^4 - 1 \end{aligned}$$

and $I = i + i' = 3 + 1 = 4$. To check correctness we not that $Q(B) \cdot B^{-4} = 1376/3^4 = 16.987654$ as required.

The ring $\mathfrak{R}_1$: We now define a ring $\mathfrak{R}_1$ out of the above operations. For reasons which will become apparent later we define the ring as pairs (q, i) where $q \in \mathbb{Z}[X]/\Phi_m(X)$ and $i \in \mathbb{Z}/\phi(m)\mathbb{Z}$. In practice we will take m to be a power of two. We define addition and multiplication as above, but now take the resulting pair modulo $\Phi_m(X)$ and $\phi(m)$. The proof of the following theorem is given in Section A.1 of the Supplementary Material.

Theorem 1. *With the above definitions $\mathfrak{R}_1$ is a ring.*

This representation of fixed point numbers in the ring $\mathfrak{R}_1$ enables us to bound the degree of the polynomial and the coefficients, after a number of homomorphic operations relatively easy, using the techniques in the next section. Of course it also implies that if we perform too many operations the degree of q will become too large and the polynomial will wrap around modulo $\Phi_m(X)$. Thus the complexity of the operations one performs not only provides a lower bound on p, i.e. an upper bound on the polynomial coefficients, but also a lower bound on the ring degree. These bounds enable us to set parameters for the SHE scheme relatively easy. However, in performing homomorphic operations we not only need for each pair (q, i) to keep a homomorphic encryption of the plaintext q, we also need to keep track (albeit in the clear) the value i.

3.2 Fractional Encoding

Our second representation of fixed point numbers dispenses with the need to keep the second component i of our first representation. On the other hand it requires us work in the cyclotomic ring $R = \mathbb{Z}[X]/(X^n + 1)$, where n is a power of two. Again we let $y = y^+.y^-$ denote the fixed point number as above, written in balanced base-B representation with $I^+ + 1$ digits in y^+ and I^- digits in y^-. We again write

$$y^+ = b_{I^+} \cdot B^{I^+} + b_{I^+-1} \cdot B^{I^+-1} + \cdots + b_1 \cdot B + b_0,$$
$$y^- = b_{-I^-} \cdot B^{-I^-} + b_{-I^-+1} \cdot B^{-I^-+1} + \cdots + b_{-2} \cdot B^{-2} + b_{-1} \cdot B^{-1}$$

where $b_i \in [-(B-1)/2, \ldots, (B-1)/2]$. We then encode the fixed point number y in the ring R by the polynomial

$$\begin{aligned} p &= \sum_{i \leq I^+} X^i b_i - \sum_{0<i\leq I^-} X^{n-i} b_{-i} \\ &= p_0(X) + p_1(X) \cdot X^{n-\mathfrak{d}_1} \end{aligned}$$

where $p_0(X) = \sum_{i\leq I^+} X^i b_i$ and $p_1(X) = -\sum_{0<i\leq I^-} b_{-i} \cdot X^{I^- - i}$, with $\mathfrak{d}_0$ and $\mathfrak{d}_1$ being the degrees of $p_0(X)$ and $p_1(X)$ respectively Thus $\mathfrak{d}_0 = I^+$ is one less than the number of digits in the integer part y^+ and $\mathfrak{d}_1 = I^-$ is the number of digits in the fractional part y^-.

Given a polynomial $q(X)$ of this form we recover the fixed point number it represents. We will need to know an upper bound for our calculation on $p_0(X)$, which can be easily calculated from the formulae below. We then take $p(X)$ and split it into p_0 and p_1 as above (using the upper bound on the degree of $p(X)$), and we recover y by setting

$$y = p_0(B) - p_1(B) \cdot B^{-\mathfrak{d}_1}$$

where we utilize the ring equation $X^n + 1 = 0$.

For our two example numbers $y = 6.370370\ldots$ and $y' = 2.666666\ldots$ we have $y \equiv p$ and $y' \equiv p'$ where

$$\begin{aligned} p &= (X^2 - X) - (X^2 + 1) \cdot X^{n-3} \\ p' &= X - (-1) \cdot X^{n-1}. \end{aligned}$$

Notice that we have in both cases that in terms of our prior representation of $(q = q_0 \cdot X^i + q_1, i)$ that

$$p_0 = q_0 \text{ and } p_1 = q_1.$$

We have $\mathfrak{d}_0 = 2$, $\mathfrak{d}'_0 = 1$, $\mathfrak{d}_1 = 3$ and $\mathfrak{d}'_1 = 1$.

Our second ring $\mathfrak{R}_2$ is the representation above, i.e. the set of polynomials modulo $X^n + 1$, which is trivially a ring. We now show that addition and multiplication in this ring corresponds to addition and multiplication of the encoded fixed point values.

Addition: Let $p(X) = p_0(X) + p_1(X) \cdot X^{n-\mathfrak{d}_1}$ and $p'(X) = p'_0(X) + p'_1(X) \cdot X^{n-\mathfrak{d}'_1}$ be two elements such as described above, encoding y and y', respectively. To perform the addition we simply add the associated polynomials as follows, without loss of generality, assume that $\mathfrak{d}_1 \leq \mathfrak{d}'_1$,

$$\begin{aligned} p + p' &= (p_0 + p_1 \cdot X^{n-\mathfrak{d}_1}) + (p'_0 + p'_1 \cdot X^{n-\mathfrak{d}'_1}) \\ &= (p_0 + p'_0) + P_1 \cdot X^{n-\mathfrak{d}_1} \\ &= P_0 + P_1 \cdot X^{n-\mathfrak{d}_1}. \end{aligned}$$

where P_0 has degree $\max(\mathfrak{d}_0, \mathfrak{d}'_0)$ and P_1 has degree $\max(\mathfrak{d}_1, \mathfrak{d}'_1)$. The polynomial P_1 will in fact be $P_1 = p_1 + p'_1 \cdot X^{\mathfrak{d}_1 - \mathfrak{d}'_1}$.

For our two example numbers, their addition therefore has the encoding

$$\begin{aligned} p + p' &= \big((X^2 - X) - (X^2 + 1) \cdot X^{n-3}\big) + \big(X - (-1) \cdot X^{n-1}\big) \\ &= X^2 - X^{n-1} - X^{n-3} + X^{n-1} \\ &= X^2 - X^{n-3}, \end{aligned}$$

which agrees with the numerical value of their sum.

Multiplication: Let $p(X) = p_0(X) + p_1(X) \cdot X^{n-\mathfrak{d}_1}$ and $p'(X) = p'_0(X) + p'_1(X) \cdot X^{n-\mathfrak{d}'_1}$ be as above. We write

$$p_0 \cdot p'_1 = r_0 + r_1 \cdot X^{\mathfrak{d}'_1}$$

and

$$p'_0 \cdot p_1 = r'_0 + r'_1 \cdot X^{\mathfrak{d}_1},$$

where $\deg(r_0) \leq \mathfrak{d}'_1 - 1$, $\deg(r_1) \leq \mathfrak{d}_0 + \mathfrak{d}'_1 - \mathfrak{d}'_1 = \mathfrak{d}_0$, $\deg(r'_0) \leq \mathfrak{d}_1 - 1$, and $\deg(r'_1) \leq \mathfrak{d}'_0 + \mathfrak{d}_1 - \mathfrak{d}_1 = \mathfrak{d}'_0$, Then the product $y \cdot y'$ is encoded by the product of the two polynomials modulo $X^n + 1$,

$$\begin{aligned} p \cdot p' &= \big(p_0 + p_1 \cdot X^{n-\mathfrak{d}_1}\big) \cdot \Big(p'_0 + p'_1 \cdot X^{n-\mathfrak{d}'_1}\Big) \\ &= p_0 \cdot p'_0 + p_0 \cdot p'_1 \cdot X^{n-\mathfrak{d}'_1} + p'_0 \cdot p_1 \cdot X^{n-\mathfrak{d}_1} + p_1 \cdot p'_1 \cdot X^{2n-\mathfrak{d}_1-\mathfrak{d}'_1} \\ &= p_0 \cdot p'_0 + p_1 \cdot p'_1 \cdot X^{2n-\mathfrak{d}_1-\mathfrak{d}'_1} \\ &\qquad + (r_0 + r_1 \cdot X^{\mathfrak{d}'_1}) \cdot X^{n-\mathfrak{d}'_1} + (r'_0 + r'_1 \cdot X^{\mathfrak{d}_1}) \cdot X^{n-\mathfrak{d}_1} \\ &= p_0 \cdot p'_0 + p_1 \cdot p'_1 \cdot X^{n-\mathfrak{d}_1-\mathfrak{d}'_1} \cdot X^n \\ &\qquad + r_0 \cdot X^{n-\mathfrak{d}'_1} + r_1 \cdot X^n + r'_0 \cdot X^{n-\mathfrak{d}_1} + r'_1 \cdot X^n \\ &= (p_0 \cdot p'_0 - r_1 - r'_1) + \Big(-p_1 \cdot p'_1 + r_0 \cdot X^{\mathfrak{d}_1} + r'_0 \cdot X^{\mathfrak{d}'_1}\Big) \cdot X^{n-\mathfrak{d}_1-\mathfrak{d}'_1} \\ &= P_0(X) + P_1(X) \cdot X^{n-\mathfrak{d}_2}, \end{aligned}$$

where $\deg(P_0) = \max(\deg(p_0 \cdot p'_0), \deg r_1, \deg r'_1) = \max(\mathfrak{d}_0 + \mathfrak{d}'_0, \mathfrak{d}_0, \mathfrak{d}'_0) = \mathfrak{d}_0 + \mathfrak{d}'_0$, and $\deg(P_1) \leq \mathfrak{d}_2 = \max(\deg(p_1 \cdot p'_1), \mathfrak{d}_1 + \deg r_0, \mathfrak{d}'_1 + \deg r_1) = \max(\mathfrak{d}_1 + \mathfrak{d}'_1, \mathfrak{d}_1 + \mathfrak{d}'_1, \mathfrak{d}'_1 + \mathfrak{d}_1) = \mathfrak{d}_1 + \mathfrak{d}'_1$.

For our two example numbers, we have

$$\begin{aligned}
p \cdot p' &= \big((X^2 - X) - (X^2 + 1) \cdot X^{n-3}\big) + \big(X - (-1) \cdot X^{n-1}\big) \\
&= (X^3 - X^2) + (X^2 - X) \cdot X^{n-1}(-X^3 - X) \cdot X^{n-3} + (-X^2 - 1) \cdot X^{2 \cdot n-4} \\
&= (X^3 - X^2) + (X^5 - X^4) \cdot X^{n-4}(-X^4 - X^2) \cdot X^{n-4} + (X^2 + 1) \cdot X^{n-4} \\
&= (X^3 - X^2) + (X - 1) \cdot X^n - X^n - X^2 \cdot X^{n-4} + (X^2 + 1) \cdot X^{n-4} \\
&= (X^3 - X^2 - X + 2) + X^{n-4} \\
&= P_0 + P_1 \cdot X^{n-\mathfrak{d}_2},
\end{aligned}$$

where $\mathfrak{d}_2 = \mathfrak{d}_1 + \mathfrak{d}'_1 = 3 + 1 = 4$. To check this gives the correct value we note that

$$P_0(3) - P_1(3) \cdot 3^{-4} = \frac{1376}{81}.$$

3.3 Relating $\mathfrak{R}_1$ to $\mathfrak{R}_2$

On one hand the ring representation of fixed point numbers in the ring $\mathfrak{R}_1$ allows us to bound the resulting degree and infinity norm of the associated polynomials encoding the fixed point numbers relatively easily (see the next section). In addition it allows a wide choice of underlying rings, which could enable SIMD computation of specific fixed point operations. However, it requires the "bookkeeping" of base power which is needed to map the encoded integer into a fixed point number.

The ring $\mathfrak{R}_2$ on the other hand requires no such bookkeeping, although limited book keeping is needed to ensure decoding after decryption works correctly. In addition it requires that we work in the ring defined by polynomial arithmetic modulo $X^n + 1$, where n is a power of two. A major drawback seems to be that one cannot derive obvious bounds on the degree and coefficients in the fractional representation, something which is crucial in order to set parameters of the SHE scheme. However, such bounds *can* be derived for the fractional representation, since this representation is *isomorphic* to the representation using the ring $\mathfrak{R}_1$.

Let ϕ be as follows,

$$\phi : \begin{cases} \quad\quad \mathfrak{R}_1 & \rightarrow \quad \mathfrak{R}_2 \\ (q = q_0 \cdot X^i + q_1, i) & \mapsto q_0 - q_1 \cdot X^{n-i} \end{cases}$$

Theorem 2. *If R is defined by $Z[X]/X^n + 1$ then ϕ is a ring isomorphism*

The proof of this theorem is given in Section A.1 of the Supplementary Material.

4 Bounds on Integer Arithmetic

Considering the balanced base B method for encoding integers as polynomials we need to estimate, for a given calculation, a lower bound on the p and d. This is to determine parameters our SHE scheme needs to support to enable a given calculation to be performed correctly. In previous works this problem was not addressed. In this section we provide a methodology to produce tight bounds on the size of p, for any given computation.

To perform our analysis, we first note that as we pass through a general integer circuit each encrypted polynomial expression we are processing will be of the form

$$\sum_{d=0}^{M}\left(\sum_{d_1<d_2<\ldots<d_t}\left(\sum_{e_1+e_2+\cdots+e_k=d}\left(c_*\prod_{i=1}^{t}p_{d_i,*}^{e_i}\right)\right)\right).$$

where t is the number of distinct ranges $[-L_i,\ldots,L_i]$ for input/scalar values. $p_{d_i,*}$ is a polynomial of degree d_i with infinity norm $B_{i,0}^{\mathsf{Bal}}=1$. The c_* are some constants and the value M is the maximal depth. Here we count scalar multiplication as consuming one level of depth. If we wish to determine the infinity norm of such a term we can simplify the discussion by just considering terms of the form

$$\prod_{i=1}^{t}(1+x+x^2+\ldots+x^{d_i})^{e_i}. \tag{3}$$

Indeed we define

$$c_{[(d_1,e_1),\ldots,(d_t,e_t)]}=\left\|\prod_{i=1}^{t}(1+x+x^2+\ldots+x^{d_i})^{e_i}\right\|_\infty.$$

In what follows, to ease discussion, the subscript indices are ordered such that

$$d_i\cdot e_i\leq(d_{i+1}\cdot e_{i+1})\text{ and in the case of equality }d_i<d_{i+1}.$$

For two terms of the form $c_{[(d_1,e_1),\ldots,(d_t,e_t)]}$ and $c_{[(d_1,e_1'),\ldots,(d_t,e_t')]}$ we define

$$c_{[(d_1,e_1),\ldots,(d_t,e_t)]}\otimes c_{[(d_1,e_1'),\ldots,(d_t,e_t')]}=c_{[(d_1,e_1+e_1'),\ldots,(d_t,e_t+e_t')]}.$$

We can now bound the infinity norm of any polynomial P obtained in evaluating the integer circuit by an expression of the form

$$L_P=\sum_{e_1,\ldots,e_t}a_{[(d_1,e_1),\ldots,(d_t,e_t)]}\cdot c_{[(d_1,e_1),\ldots,(d_t,e_t)]},$$

where $a_{[(d_1,e_1),\ldots,(d_t,e_t)]}$ are constants depending on the precise polynomial P, and we think of this (for now) as a formal sum in the variables $c_{[(d_1,e_1),\ldots,(d_t,e_t)]}$. For an input or scalar value from the range $[-L_i,\ldots,L_i]$ the infinity norm of the polynomial P_0 is bounded by

$$L_{P_0}=c_{[(d_1,0),\ldots,(d_{i-1},0),\ (d_i,1),\ (d_{i+1},0),\ldots,(d_t,0)]}.$$

We can derive upper bounds on the infinity norm of the polynomials as we pass through the integer circuit using the following rules. Given upper bounds on the infinity norm of polynomials P and P' in this form given by

$$L_P = \sum_{e_1,\ldots,e_t} a_{[(d_1,e_1),\ldots,(d_t,e_t)]} \cdot c_{[(d_1,e_1),\ldots,(d_t,e_t)]},$$
$$L_{P'} = \sum_{e'_1,\ldots,e'_t} a_{[(d_1,e'_1),\ldots,(d_t,e'_t)]} \cdot c_{[(d_1,e'_1),\ldots,(d_t,e'_t)]},$$

we can derive upper bounds on the infinity norm of the sum and the product of these polynomials terms via the equations

$$\begin{aligned} L_{P+P'} &= L_P + L_{P'}, \\ L_{P\cdot P'} &= \sum_{e_1,\ldots,e_t,e'_1,\ldots,e'_t} \left(a_{[(d_1,e_1),\ldots,(d_t,e_t)]} \cdot a_{[(d_1,e'_1),\ldots,(d_t,e'_t)]}\right) \\ &\qquad \cdot \left(c_{[(d_1,e_1),\ldots,(d_t,e_t)]} \otimes c_{[(d_1,e'_1),\ldots,(d_t,e'_t)]}\right) \end{aligned}$$

Is it clear that the degree of the sum of two polynomials is the maximum of the degrees, and the degree of the product is the sum of the degrees.

4.1 Bounding $c_{[(d_1,e_1),\ldots,(d_t,e_t)]}$

To use these bounds we eventually obtain a formal expression for infinity norm of the output of the circuit consisting of a linear polynomial in the terms $c_{[(d_1,e_1),\ldots,(d_t,e_t)]}$. We thus are left with simply bounding $c_{[(d_1,e_1),\ldots,(d_t,e_t)]}$. We perform this bounding at the end, rather than as we go, as these leads to much tighter bounds on the infinity norm of the output polynomial.

We first present some basic facts on the case of a single pair of terms (d,e). Let $d,e \geq 0$ be integers, and define a_i for $0 \leq i \leq d \cdot e$ as

$$(1 + x + x^2 + \ldots + x^d)^e = \sum_{i=0}^{d\cdot e} a_i \cdot x^i. \tag{4}$$

We then define

$$c_{d,e} = \left\| (1 + x + x^2 + \ldots + x^d)^e \right\|_\infty = \max_{0<i<d\cdot e} a_i. \tag{5}$$

Naively we can obtain upper and lower bounds on $c_{d,e}$ as follows:

$$\frac{(d+1)^e}{d\cdot e+1} \leq c_{d,e} \leq (d+1)^e. \tag{6}$$

The upper bound is obtained by evaluating (4) at $x = 1$ and the lower bound is obtained from the upper bound by noting that there are only $d \cdot e + 1$ coefficients a_i in (4). We have the trivial bounds $c_{d,0} = c_{d,1} = 1$ and $c_{d,2} = (d+1)$.

The parameter $c_{d,e}$ is also of interest in probability theory and bounds on its value have been previously analysed [1, 11]. The following upper bound follows from the main theorem in [11] (see also [1] for a relation between the parameter $c_{m,n}$ and the main parameter studied in [11]).

Theorem 3. *If $e \neq 2$ or $d \in \{1, 2, 3\}$, then*

$$c_{d,e} < \sqrt{\frac{6}{\pi \cdot d \cdot e \cdot (d+2)}} \cdot (d+1)^e. \tag{7}$$

The above upper bound is optimal in the following sense [11, Remark (a)].

Corollary 1. $\lim_{e\to\infty} \frac{\sqrt{e}\cdot c_{d,e}}{(d+1)^e} = \sqrt{\frac{6}{\pi\cdot d\cdot(d+2)}}$.

Although it is unknown whether the above convergence is uniform as d varies as well.

Given this bound on terms $c_{d,e}$ we can now derive bounds on our terms $c_{[(d_1,e_1),\dots,(d_t,e_t)]}$ as follows. Recalling our ordering of the pairs in the subscript of $d_i \cdot e_i \leq (d_{i+1} \cdot e_{i+1})$ and in the case of equality, $d_i < d_{i+1}$. We (recursively) use the following bound, where d_k is the first value of d_i in the subscript for which the associated e_k value is non-zero,

$$c_{[(d_1,e_1),\dots,(d_t,e_t)]} \leq (d_k \cdot e_k + 1) \cdot c_{d_k,e_k} \cdot c_{[(d_1,e'_1),\dots,(d_t,e'_t)]}, \tag{8}$$

where $e'_i = e_i$ except that $e'_k = 0$.

4.2 Applying The Bounds

We can now estimate the size of p and d needed to ensure correctness when evaluating our example balanced integer circuit that consists of M levels and A additions per level. The infinity norm bound on our polynomials becomes

$$B_M = c_{d,2^M} \cdot 2^{A\left(2^{M+1}-2\right)},$$

assuming the input values are in the range $[-L, \dots, L]$ and using a balanced base-3 representation of the input values, so $d = d^{\mathsf{Bal}} = \lceil \log(2\cdot L+1)/\log 3\rceil - 1$. The degree bound for our circuit output value is $d_{\mathsf{out}} = 2^M \cdot d_0$. From Theorem 3, a sharp upper bound on B_M (for $M > 1$, or $d > 3$ if $M = 1$) is

$$B_M < \sqrt{\frac{6}{\pi \cdot 2^M \cdot d(d+2)}} \cdot (d+1)^{2^M} \cdot 2^{A\left(2^{M+1}-2\right)}.$$

To ensure correctness, when we encrypt and manipulate these polynomials homomorphically, we need to ensure that our SHE scheme supports a plaintext with

$$p > 2 \cdot B_M,$$
$$\deg(R) > d_M.$$

The most stringent constraint is that on p, and we give examples in Section A.2 of the Supplementary Material.

Of course given a specific circuit we could derive other values of d_M and B_M, the above are just examples in the case of our regular circuit with multiplicative depth M and A additions per level. See later for an application using the FFT where our more general analysis becomes applicable.

5 Homomorphic Image Processing via the Fourier Transform

A standard image processing pipeline is to take an image (consisting of n pixels), pass it into the frequency domain by applying the Fourier transform, apply an operation in the Fourier domain, and then map back to the image space by applying the inverse Fourier transform. The operation in the Fourier domain in its simplest form could be the Hadamard component wise multiplication of the data by a fixed matrix. For example this is used when applying Gabor filters, which feature prominently in applications that are motivated by biological vision.

In this section we examine the application of our fixed point analysis to the case of image processing in which the initial image *and* the Hadamard transformation data are both encrypted using a SHE scheme. It is well known that the Fourier transform is a linear operation, and hence only requires (in theory) an additively homomorphic encryption scheme to obtain an encrypted version. However, our requirement that the processing in the frequency domain is also unknown to the evaluator implies that our overall operation is non-linear. For means of comparison of parameters with prior work [2, 3], which used Paillier encryption and only processed a single FFT operation, we also provide a comparison of parameters in that case.

5.1 The Mixed Fourier Transform Algorithm

The standard method to apply the (radix-2) Fourier transform[5] is to use the Fast Fourier Transform (FFT) which is a recursive algorithm requiring $O(\log n)$ depth of scalar multiplications and a total of $O(n \cdot \log n)$ scalar multiplications in total. As we have seen the need to perform a large depth of scalar multiplications will imply a large plaintext modulus for our SHE scheme. The naive method of performing the Fourier transform is to simply apply a matrix-vector product. This requires only depth one of scalar multiplications but on the other hand requires $O(n^2)$ scalar multiplications. We will refer to this method as the Naive Fourier Transform (NFT).

There is an obvious balance to be struck here, which we present in Figure 1. This is an algorithm, which we dub the Mixed Fourier Transform (MFT) algorithm. It executes standard recursive FFT algorithm down to a given depth $\lfloor \log_2(\mathfrak{B}) \rfloor$, and then at this lower level executes the naive Fourier transform method.

When we execute $MFT(\mathbf{x}, n, 1)$ we perform the full traditional Fast Fourier Transform method, while when we execute $MFT(\mathbf{x}, n, n)$ we perform the naive Fourier transform method. All values of $\mathfrak{B}$ in between execute a hybrid approach. By varying $\mathfrak{B}$ we can trade a reduced depth of scalar multiplications for an increased total number of multiplications. It is obvious that the depth of scalar

[5] Other FFT's, e.g. the radix-4 method, can be analysed using similar techniques to those in this paper.

$\underline{MFT(\mathbf{x}, n, \mathfrak{B})}$

if $n \leq \mathfrak{B}$ **then**
  for $0 \leq k \leq n-1$ **do**
    $\mathbf{y}_k \leftarrow \sum_{j=0}^{n-1} x_j \cdot \exp(-2 \cdot \pi \cdot \sqrt{-1} \cdot j \cdot k/n)$.
  end for
else
  $m \leftarrow n/2$.
  $z_0, \cdots, z_{n/2-1} \leftarrow MFT((x_0, x_2, x_4, \ldots, x_{n-2}), m, \mathfrak{B})$.
  $z_{n/2}, \cdots, z_n \leftarrow MFT((x_1, x_3, x_5, \ldots, x_{n-1}), m, \mathfrak{B})$.
  for $0 \leq k \leq n/2-1$ **do**
    $s \leftarrow \exp(-2 \cdot \pi \cdot \sqrt{-1} \cdot k/n) \cdot z_{k+n/2}$.
    $t \leftarrow z_k$.
    $\mathbf{y}_k \leftarrow t + s$.
    $\mathbf{y}_{k+n/2} \leftarrow t - s$.
  end for
end if
return y

Fig. 1. The Mixed Fourier Transform Algorithm

multiplications required is given by

$$\mathsf{depth}(n, \mathfrak{B}) = \log_2(n) - \log_2(\mathfrak{B}) + 1.$$

Computing the total number of scalar multiplications requires a little more thought. For $n = 2^N$ and $\mathfrak{B} = 2^B$, the first level of the FFT operation has

$$\mathsf{mults}(n, \mathfrak{B}) = 2 \cdot \mathsf{mults}(n/2, \mathfrak{B}) + 2^{N-1}$$

multiplications. Doing FFT until we reach $\mathfrak{B}$ gives

$$\mathsf{mults}(n, \mathfrak{B}) = 2^{N-B} \cdot \mathsf{mults}(\mathfrak{B}, \mathfrak{B}) + (N - B) \cdot 2^{N-1}.$$

Solving this yields

$$\mathsf{mults}(n, \mathfrak{B}) = n \cdot B + (\log_2(n) - \log_2(\mathfrak{B})) \cdot \frac{n}{2}$$

as the number of multiplications performed in a MFT circuit.

5.2 Comparison With Prior Work

In [2, 3] the authors present work on implementing a radix-2 FFT in the encrypted domain using the Paillier encryption algorithm. As a means of comparison of their work with ours we examine how their Paillier parameters would compare to our Ring-LWE parameters in their setting. The first key aspect is the precision of the input values, the roots of unity and the output precision. Both [2, 3] and ourselves use a fixed point encoding in which precision is never lost. But if one implemented FFT on a machine with b bits of floating point precision one would loose precision as the calculation proceeds. This means that to obtain the same output as running in the clear on a standard machine using floating point arithmetic, we can adapt the precision of the roots of unity.

In particular, we let b_1 denote the bits of precision in the input data (which is typically eight), b_2 denote the bits of precision in the roots of unity and b denote the bits of equivalent output bits of precision in an in-the-clear implementation. Then [2, 3] show that for a single iteration of the FFT algorithm on data of size 2^v, one can take

$$b_2 = \left\lceil b - \frac{v}{2} + \frac{1}{2} \right\rceil.$$

Using this they are able to implement the FFT in the encrypted domain using a Paillier modulus of bit size

$$n_P \geq v + \alpha \cdot b_2 + b_1 + 4,$$

where $\alpha = 1$ for the naive Fourier transform, and $\alpha = v - 2$ for the full FFT; they do not consider a mixed Fourier Transform.

As a means of comparison we look at the same situation using our polynomial encoding for use in the Ring-LWE system. The degrees of the associated polynomails to encode the input data and the roots of unity, in balanced base 3 encoding, are

$$d_i = \lceil \log(2 \cdot 2^{b_i} + 1) / \log 3 \rceil - 1.$$

Applying the analysis from Section 4 to a single Fourier Transform execution, we can obtain formulae for the infinity norm of the resulting polynomials via a computer algebra system in the form of a linear sum of terms the following form

$$c_{[(d_1,1),(d_2,e_2)]},$$

where $0 \leq e_2 \leq \mathsf{depth}(n, \mathfrak{B})$. Note that $e_1 = 1$ as we are only executing a single FFT operation.

Then using equations 7 and 8 and the fact that $c_{d,1} = 1$ we can give an upper bound this quantity

$$c_{[(d_1,1),(d_2,e_2)]} \leq \left\{ c \cdot (d_1 + 1) \cdot (d_2 + 1)^{e_2} \quad \text{Otherwise,} \right.$$

where

$$c = \sqrt{\frac{6}{\pi \cdot d_2 \cdot e_2 \cdot (d_2 + 2)}}.$$

Hence, we can upper bound the linear sum and so lower bound the plaintext modulus p needed for the SHE scheme to ensure correctness. A similar method allows us to upper bound the degree of the resulting polynomials. This itself leads to a lower bound on the ring dimension $\deg(R)$ needed for the SHE scheme. We summarize the results in Table 1 for emulating $b = 32$ bits of floating point precision and $b_1 = 8$ bit inputs.

n	b_2	d_1	d_2	FFT $\log_2 p \geq$	FFT $\deg(R) \geq$	FFT $n_P \geq$	NFT $\log_2 p \geq$	NFT $\deg(R) \geq$	NFT $n_P \geq$
64	30	5	19	35	138	138	11	24	48
256	29	5	18	45	167	194	13	23	49
1024	28	5	18	56	203	246	15	23	50

Table 1. Comparing Paillier vs Ring-LWE encoding parameters for a single NFT/FFT execution for $b = 32$

5.3 FFT-Hadamard-iFFT Pipeline

We now turn to investigating the FFT-Hadamard-iFFT standard image processing pipeline. Since we apply two Fourier transforms the precision of the roots of unity we take to be

$$b_2 = \left\lceil b - v + \frac{1}{2} \right\rceil,$$

in order to retain the same precision as b bits of floating point precision on a standard machine.

Applying the analysis from Section 4 again, we obtain formulae for the infinity norm of the resulting polynomials in the form of a linear sum of terms of the following form

$$c_{[(d_1,2),(d_2,e_2)]},$$

where $0 \leq e_2 \leq \mathsf{depth}(n, \mathfrak{B})$. Then using equations 7 and 8, and the fact that $c_{d,2} = (d+1)$ we now upper bound this quantity via

$$c_{[(5,2),(10,e_2)]} \leq \begin{cases} 36 & \text{If } e_2 = 1, \\ c \cdot (2 \cdot 5 + 1) \cdot (5 + 1) \cdot (10 + 1)^{e_2} & \text{Otherwise,} \end{cases}$$

where

$$c = \begin{cases} \sqrt{\frac{6}{\pi \cdot 10 \cdot e_2 \cdot (10+2)}} & \text{If } e_2 > 2, \\ 1 & \text{Otherwise.} \end{cases}$$

Hence, we can upper bound the linear sum and so lower bound the plaintext modulus p needed for the SHE scheme to ensure correctness. This results in the parameters given in Table 2

We then took this bounds and instantiated an SHE system to evaluate the pipeline using the HELib library [9]. The HELib library implements the BGV [4, 8] Somewhat Homomorphic Encryption scheme, but restricts the plaintext modulus to be at most 64 bits in length. Hence, our experiments are limited to this reduced size of plaintext space.

In this scheme a plaintext $m \in R_p$ is encrypted as a pair of elements in $(c_0, c_1) \in R_q^2$, such that

$$c_0 - \mathfrak{sk} \cdot c_1 = m + p \cdot \epsilon \pmod{q}$$

n	b_2	d_1	d_2	FFT $\mathfrak{B}=1$ $\log_2 p \geq$	FFT $\mathfrak{B}=1$ $\deg(R) \geq$	$\mathfrak{B}=\sqrt{n}$ $\log_2 p \geq$	$\mathfrak{B}=\sqrt{n}$ $\deg(R) \geq$	NFT $\mathfrak{B}=n$ $\log_2 p \geq$	NFT $\mathfrak{B}=n$ $\deg(R) \geq$
16	29	5	18	54	190	37	118	25	46
64	27	5	17	74	248	49	146	29	44
256	25	5	16	93	298	61	170	33	42
1024	23	5	15	112	340	72	190	37	40

Table 2. Parameters for the FFT-Hadamard-iFFT pipeline

where $\mathfrak{sk}$ is the secret key (a short element in R_q) and ϵ is a short "noise" element in R_q. As homomorphic operations progress the value q of the ciphertext is reduced, until it can be reduced no more. At this point, operations cease to be possible. The reduction in q enables the noise value to be controlled, and each reduction in q is said to consume a homomorphic "level". Note, that the HELib library due to its choice of moduli for each level actually consumes multiple "internal levels" for each of these "external levels".

In Table 3 we present our implementation results using the HELib. In each case we used the plaintext modulus size derived from the Table 2, we note that in all cases HELib selects a ring dimension for security reasons which is much larger than we need for our application. This last fact means that by careful choice of the plaintext modulus one can process many such operations in parallel using standard SIMD tricks; with the amortization constant being (roughly) the actual degree of R divided by the lower bound from 2. We note that we cannot obtain results for the larger plaintext spaces as HELib has a restriction of 60 bits on the plaintext modulus. In future work we aim to remove this restriction by utilizing a different SHE library. All run times measure the time in seconds to evaluate the FFT-Hadamard-iFFT pipeline in the homomorphic domain, and they are obtained on a machine with six Intel Xeon E5 2.7GHz processors, and with 64 GB RAM.

n	$\mathfrak{B}$	$\deg(R)$	$\log_2 q$	HELib Levels	Amortization Amount	CPU Time	Amortized Time
16	1	32768	710	33	172	188	1.09
16	4	32768	451	19	277	147	0.53
16	16	16384	192	9	356	106	0.3
64	8	32768	622	30	224	1500	6.69
64	64	16384	192	10	372	1582	4.25
256	256	16384	278	11	390	34876	89.4

Table 3. Results For Homomorphically Evaluating A Full Image Processing Pipeline

Acknowledgements

This work has been supported in part by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO and by the European Union's H2020 Programme under grant agreement number ICT-644209 (HEAT). The authors would like to thank Carl Ek for input on image processing algorithms and Daniel P. Martin for valuable inputs throughout.

References

1. Hecène Belbachir. Determining the mode for convolution powers of discrete uniform distribution. *Probability in the Engineering and Informational Sciences*, 25:469–475, 10 2011.
2. Tiziano Bianchi, Alessandro Piva, and Mauro Barni. Comparison of different FFT implementations in the encrypted domain. In *2008 16th European Signal Processing Conference, EUSIPCO 2008, Lausanne, Switzerland, August 25-29, 2008*, pages 1–5. IEEE, 2008.
3. Tiziano Bianchi, Alessandro Piva, and Mauro Barni. On the implementation of the discrete fourier transform in the encrypted domain. *IEEE Transactions on Information Forensics and Security*, 4(1):86–97, 2009.
4. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS*, pages 309–325. ACM, 2012.
5. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.
6. Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Manual for using homomorphic encryption for bioinformatics, 2015. Available at http://research.microsoft.com/pubs/258435/ManualHE.pdf. Last accessed on November 20, 2015 at 17:45.
7. Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.
8. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.
9. Shai Halevi and Victor Shoup. Algorithms in helib. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2014.
10. Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? *IACR Cryptology ePrint Archive*, 2011:405, 2011.
11. Lutz Mattner and Bero Roos. Maximal probabilities of convolution powers of discrete uniform distributions. *Statistics & Probability Letters*, 78(17):2992 – 2996, 2008.

12. Nigel P. Smart and Frederik Vercauteren. Fully homomorphic simd operations. *IACR Cryptology ePrint Archive*, 2011:133, 2011.
13. Antonio Torralba, Robert Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, 2008.

A Supplementary Material

A.1 Proofs of Theorems

Proof of Theorem 1

Proof. The additive identity in $\mathfrak{R}_1$ is the pair $(0,0)$, which corresponds to the fixed point number 0. The additive inverse of any element $(q,i) \in \mathfrak{R}_1$ is $(-q,i)$. It is clear that these two elements sum up to $(0,0)$. Thus $\mathfrak{R}_1$ is an additive group; the fact that it is abelian is immediate.

The multiplicative identity is $(1,0)$, corresponding to the fixed point number 1. The associativity of the multiplication is trivially implied by associativity of (modular) polynomial multiplication and (modular) integer addition. We show that distributivity of multiplication over addition holds, thus completing the proof.

Let $(q_1,i_1),(q_2,i_2)$ and (q_3,i_3) be three elements of $\mathfrak{R}_1$. Without loss of generality, assume that $i_2 \geq i_3$, then

$$\begin{aligned}
\big(q_1,i_1\big)\cdot\big((q_2,i_2)+(q_3,i_3)\big) &= (q_1,i_1)\cdot(q_2+q_3\cdot X^{i_2-i_3},i_2)\\
&= (q_1\cdot q_2+q_1\cdot q_3\cdot X^{i_2-i_3},i_1+i_2)\\
&= \big(q_1\cdot q_2+q_1\cdot q_3\cdot X^{i_1+i_2-i_1-i_3},\max(i_1+i_2,i_1+i_3)\big)\\
&= (q_1\cdot q_2,i_1+i_2)+(q_1\cdot q_3,i_1+i_3)\\
&= (q_1,i_1)\cdot(q_2,i_2)+(q_1,i_1)\cdot(q_3,i_3).
\end{aligned}$$

□

Proof of Theorem 2

Proof. First note that

1. $\phi(1_{\mathfrak{R}_1}) = \phi(1,0) = \phi(1\cdot X^0+0) = 1-0\cdot X^0 = 1 = 1_{\mathfrak{R}_2}$.
2. Let (q,i) and (q',i') in $\mathfrak{R}_1$; without loss of generality assume $i \geq i'$. Then
$$(q,i)+(q',i') = q+q'\cdot X^{i-i'} = (Q,i).$$
   Then
$$\begin{aligned}
\phi(Q,i) &= \phi(q+q'\cdot X^{i-i'},i)\\
&= \phi\big(X^i(q_0+q_0')+(q_1+q_1'),i\big)\\
&= (q_0+q_0')-(q_1+q_1')\cdot X^{n-i}\\
&= (q_0-q_1\cdot X^{n-i})+(q_0'-q_1'\cdot X^{n-i})\\
&= \phi(q,i)+\phi(q',i').
\end{aligned}$$

Notice that in the above, we have implicitly made use of addition properties of $\mathfrak{R}_2$.

3. Let q, q' as above.

$$\begin{aligned}\phi(q,i)\cdot\phi(q',i') &= (q_0 - q_1\cdot X^{n-i})\cdot(q_0' - q_1'\cdot X^{n-i'})\\ &= q_0\cdot q_0' - q_0\cdot q_1'\cdot X^{n-i'}\\ &- q_0'\cdot q_1\cdot X^{n-i} + q_1\cdot q_1'\cdot X^{n-I},\end{aligned}$$

Where $I = i + i'$. Now computing $(q,i)\cdot(q',i')$ first,

$$\begin{aligned}q\cdot q' &= q_0\cdot q_0'\cdot X^I + q_0\cdot q_1'\cdot X^i\\ &+ q_0'\cdot q_1\cdot X^{i'} + q_1\cdot q_1'.\end{aligned}$$

Now, viewing this as the pair $(Q = q\cdot q' \mod X^n+1, i+i' \mod n) = \big((q_0\cdot q_0' + q_1\cdot q_1'\cdot X^{n-i-i'})\cdot X^{i+i'}) + (q_0\cdot q_1'\cdot X^i + q_0'\cdot q_1\cdot X^{i'}), i+i'\big)$, we get the following.

$$\begin{aligned}\phi(q\cdot q', I) &= \phi\big(q_0\cdot q_0' + q_1\cdot q_1'\cdot X^{n-i-i'})\cdot X^{i+i'}) + (q_0\cdot q_1'\cdot X^i + q_0'\cdot q_1\cdot X^{i'}), I\big)\\ &= q_0\cdot q_0' + q_1\cdot q_1'\cdot X^{n-i-i'}\\ &- (q_0\cdot q_1'\cdot X^i - q_0'\cdot q_1\cdot X^{i'})\cdot X^{n-I}\\ &= q_0\cdot q_0' - q_0\cdot q_1'\cdot X^{n-i'}\\ &- q_0'\cdot q_1\cdot X^{n-i} + q_1\cdot q_1'\cdot X^{n-I}\\ &= \phi(q,i)\cdot\phi(q',i'),\end{aligned}$$

so that ϕ is indeed a homomorphism between $\mathfrak{R}_1$ and $\mathfrak{R}_2$.

To finish the proof, we show that ϕ is bijective. For any $y = q_0 + q_1\cdot X^{n-\mathfrak{d}_1}$ in $\mathfrak{R}_2$, we have that $(q,\mathfrak{d}_1) = (q_0\cdot X^{\mathfrak{d}_1} + q_1, \mathfrak{d}_1)$ maps to y so that the mapping is surjective. To see that it is injective, suppose for $p, p' \in \mathfrak{R}_1$ we have that $\phi(p) = \phi(p') = z \in \mathfrak{R}_2$. Remember both rings contain encoding of fractional numbers written in balanced base B. Recall also that we recover the integers by simply evaluating (in our case) $z(B) = a \in \mathbb{Q}$, and since this is well-defined, a is unique. Now encode a in the ring $\mathfrak{R}_1$; the encoding operation (for both rings) is well-defined, therefore a will have a unique image in the ring $\mathfrak{R}_1$ and thus $p = p'$. It follows that ϕ is an isomorphism. □

A.2 Lower Bounds On p For Regular Circuits

Tables 5, 6, 7 lists the size in bits of the smallest prime satisfying the above bounds and also lists the degree bound $d_M = 2^M\cdot d_0$ for small values of A and M for balanced base encoding with $B = 3$, 5 and 7 and $L = 2^{19}$. For the sake of comparison, we give also give Table 4 that suggests the size of the primes for the non-balanced base encoding for $B = 2$ and $L = 2^{19}$. It is clearly evident that using balanced base encoding with $B = 3$ yields the smallest primes, although large multiplicative depth is hard to support in any method.

It should be noted that with current SHE schemes a ciphertext modulus over 256 bits in length seems currently infeasible for moderately sized circuits to be evaluated. Thus it is clear that if anything but small values of M are to be considered one needs a different way of encoding fixed point numbers. One such possibility is via multiple encryptions using different plaintext moduli, and then to use the Chinese Remainder Theorem to recover the final plaintext polynomial.

M	1	2	3	4	5	6	7	8	9	10
$A = 0$	6	14	31	65	133	271	547	1100	2206	4418
$A = 1$	8	20	45	95	195	397	801	1610	3228	6464
$A = 2$	10	26	59	125	257	523	1055	2120	4250	8510
$A = 3$	12	32	73	155	319	649	1309	2630	5272	10556
$A = 4$	14	38	87	185	381	775	1563	3140	6294	12602
$A = 5$	16	44	101	215	443	901	1817	3650	7316	14648
$A = 6$	18	50	115	245	505	1027	2071	4160	8338	16694
$A = 7$	20	56	129	275	567	1153	2325	4670	9360	18740
$A = 8$	22	62	143	305	629	1279	2579	5180	10382	20786
$A = 9$	24	68	157	335	691	1405	2833	5690	11404	22832
$A = 10$	26	74	171	365	753	1531	3087	6200	12426	24878
d_M	38	76	152	304	608	1216	2432	4864	9728	19456

Table 4. Size (in bits) of the smallest p and the degree bounds for *non-balanced* encoding with $B = 2$ and $L = 2^{19}$.

M	1	2	3	4	5	6	7	8	9	10
$A = 0$	5	12	26	55	114	232	468	942	1888	3783
$A = 1$	7	18	40	85	176	358	722	1452	2910	5829
$A = 2$	9	24	54	115	238	484	976	1962	3932	7875
$A = 3$	11	30	68	145	300	610	1230	2472	4954	9921
$A = 4$	13	36	82	175	362	736	1484	2982	5976	11967
$A = 5$	15	42	96	205	424	862	1738	3492	6998	14013
$A = 6$	17	48	110	235	486	988	1992	4002	8020	16059
$A = 7$	19	54	124	265	548	1114	2246	4512	9042	18105
$A = 8$	21	60	138	295	610	1240	2500	5022	10064	20151
$A = 9$	23	66	152	325	672	1366	2754	5532	11086	22197
$A = 10$	25	72	166	355	734	1492	3008	6042	12108	24243
d_M	24	48	96	192	384	768	1536	3072	6144	12288

Table 5. Size (in bits) of the smallest p and the degree bounds for *balanced* encoding with $B = 3$ and $L = 2^{19}$.

M	1	2	3	4	5	6	7	8	9	10
$A = 0$	7	14	31	64	130	263	529	1062	2129	4264
$A = 1$	9	20	45	94	192	389	783	1572	3151	6310
$A = 2$	11	26	59	124	254	515	1037	2082	4173	8356
$A = 3$	13	32	73	154	316	641	1291	2592	5195	10402
$A = 4$	15	38	87	184	378	767	1545	3102	6217	12448
$A = 5$	17	44	101	214	440	893	1799	3612	7239	14494
$A = 6$	19	50	115	244	502	1019	2053	4122	8261	16540
$A = 7$	21	56	129	274	564	1145	2307	4632	9283	18586
$A = 8$	23	62	143	304	626	1271	2561	5142	10305	20632
$A = 9$	25	68	157	334	688	1397	2815	5652	11327	22678
$A = 10$	27	74	171	364	750	1523	3069	6162	12349	24724
d_M	16	32	64	128	256	512	1024	2048	4096	8192

Table 6. Size (in bits) of the smallest p and the degree bounds for *balanced* encoding with $B = 5$ and $L = 2^{19}$.

M	1	2	3	4	5	6	7	8	9	10
$A = 0$	8	16	34	70	143	289	582	1169	2342	4689
$A = 1$	10	22	48	100	205	415	836	1679	3364	6735
$A = 2$	12	28	62	130	267	541	1090	2189	4386	8781
$A = 3$	14	34	76	160	329	667	1344	2699	5408	10827
$A = 4$	16	40	90	190	391	793	1598	3209	6430	12873
$A = 5$	18	46	104	220	453	919	1852	3719	7452	14919
$A = 6$	20	52	118	250	515	1045	2106	4229	8474	16965
$A = 7$	22	58	132	280	577	1171	2360	4739	9496	19011
$A = 8$	24	64	146	310	639	1297	2614	5249	10518	21057
$A = 9$	26	70	160	340	701	1423	2868	5759	11540	23103
$A = 10$	28	76	174	370	763	1549	3122	6269	12562	25149
d_M	14	28	56	112	224	448	896	1792	3584	7168

Table 7. Size (in bits) of the smallest p and the degree bounds for *balanced* encoding with $B = 7$ and $L = 2^{19}$.