Fully Homomorphic Encryption for Point Numbers

Seiko Arita and Shota Nakasato

2016.04.22
Graduate School of Information Security,
Institute of Information Security, Japan

Abstract. In this paper, based on the FV scheme, we construct a first fully homomorphic encryption scheme FHE4FX that can homomorphically compute addition and/or multiplication of encrypted fixed point numbers without knowing the secret key. Then, we show that in the FHE4FX scheme one can efficiently and homomorphically compare magnitude of two encrypted numbers. That is, one can compute an encryption of the greater-than bit that represents whether or not $x > x'$ given two ciphertexts c and c' (of x and x', respectively) without knowing the secret key. Finally we show that these properties of the FHE4FX scheme enables us to construct a fully homomorphic encryption scheme FHE4FL that can homomorphically compute addition and/or multiplication of encrypted floating point numbers.

Keywords: Fully homomorphic encryption, FV scheme, Fixed point number, Floating point number, Greater-than bit.

1 Introduction

Using Fully Homomorphic Encryption (FHE) scheme, one can homomorphically compute an encrypted XORed bit $\mathsf{Enc}(b_1 \,\mathsf{XOR}\, b_2)$ and/or encrypted AND bit $\mathsf{Enc}(b_1 \,\mathsf{AND}\, b_2)$ of given encrypted bits $\mathsf{Enc}(b_1)$ and $\mathsf{Enc}(b_2)$ without knowing the secret key [19, 8]. Since any function can be written using XOR and AND gates, this means that one can homomorphically compute any function of encrypted bits without knowing the secret key.

Practically, computation over bitwise encryptions is not efficient. It is a kind of "1-bit" processor. In schemes such as [3, 5], one can encrypt congruent integers (i.e., $x \bmod n$) and can homomorphically compute addition $\mathsf{Enc}(x_1 + x_2 \bmod n)$ and/or multiplication $\mathsf{Enc}(x_1 \times x_2 \bmod n)$ of encrypted congruent integers $\mathsf{Enc}(x_1)$ and $\mathsf{Enc}(x_2)$ without knowing the secret key. Nowadays based on those schemes, various mining algorithms are experimentally and homomorphically evaluated against outsourced genomic, medical, or financial encrypted data [17, 11, 14, 6, 15, 13, 12].

However, the real world is not comprised of congruent integers. Real numbers have greater-than relation $x < y$, which requires computation of the most significant bit of $x - y$. Is it possible to efficiently compute $\mathsf{Enc}(\mathsf{MSb}(x-y))$ given $\mathsf{Enc}(x)$ and $\mathsf{Enc}(y)$ without knowing the secret key? Moreover, to compute real numbers, we must depend on some precision control mechanism, that enables computation of real numbers as fixed or floating point number computation. Is it possible to realize precision control against x only given its encryption $\mathsf{Enc}(x)$ without knowing the secret key? To make the theoretical universality of FHE schemes be of more practical interest, we need to resolve such problems. As we will see later, the two problems are tightly related to each other.

1.1 Our Contribution

First, we construct a fully homomorphic encryption scheme for fixed point numbers.

FHE scheme for fixed point numbers. Our starting point is the FV scheme given by Fan and Vercauteren [7], which is an FHE scheme for congruent integers, instantiating the FHE scheme by Brakerski [2] based on the Ring LWE problem [18].

Let $R = \mathbb{Z}[X]/(\Phi_m(X))$ be the m-th cyclotomic ring, where $\Phi_m(X)$ denotes the m-th cyclotomic polynomial. Elements a of cyclotomic ring R are called cyclotomic integers and represented by integer coefficient polynomials $a(X) = \sum_{i=0}^{n-1} a_i X^i$ of degree $< n := \phi(m)$. ($\phi(\cdot)$ denotes the Euler function.) Two cyclotomic integer a and b are added through ordinal polynomial addition: $(a+b)(X) = \sum_{i=0}^{n-1}(a_i+b_i)X^i$. Product of cyclotomic integer a and b is computed as polynomial multiplication followed by reduction via $\Phi_m(X)$: $(a \cdot b)(X) = a(X)b(X) \bmod \Phi_m(X)$.

We can reduce cyclotomic integers $a(X) = \sum_{i=0}^{n-1} a_i X^i$ modulo any positive integer q, by reducing each coefficient a_i modulo q, resulting elements of residue ring R_q defined as $R_q = \mathbb{Z}_q[X]/(\Phi_m(X))$. Elements of the residue ring R_q are represented by mod-q integer coefficient polynomials.

In the FV scheme, each ciphertext is a pair $c = (c_0, c_1)$ $(\in R_q \times R_q)$ of cyclotomic integers modulo some ciphertext modulus q. A plaintext is a cyclotomic integer x $(\in R_t)$ modulo some plaintext modulus t. The ciphertext modulus q must be sufficiently larger than the plaintext modulus t to afford noises occurred among computations. For simplicity we assume t divides q in this paper.

In the FV scheme, a ciphertext $c = (c_0, c_1)$ for some plaintext $x \in R_t$ will be created to satisfy following relation with some small noise term v $(\in R_q)$ and some cyclotomic integer $\alpha \in R$:

$$c_0 + c_1 s = \frac{q}{t}x + v + q\alpha.$$

If one knows the secret key $s \in R$, by computing $\left\lfloor \frac{t}{q}(c_0 + c_1 s) \right\rceil \bmod t$, one can recover the plaintext x from the ciphertext $c = (c_0, c_1)$ provided that coefficients of noise v are not too large relative to the modulus ratio $\frac{q}{t}$.

We modify the FV scheme so that we can treat fixed point numbers $\tilde{x} = 2^{-m}x$ $(x \in \mathbb{Z}_{2^{m+l}})$. Here, m is bit-length after point and l is bit-length before point of $\tilde{x}$ and so $x = 2^m\tilde{x}$ becomes an integer in $\mathbb{Z}_{2^{m+l}}$. We suppose that integer x represents the constant polynomial x (i.e., a polynomial that has only constant term x as nonzero term) in the cyclotomic ring R. To enable homomorphic computation of $\tilde{x}$ via homomorphic computation of x, we will use the following relation:

$$c_0 + c_1 s = \frac{q}{t}2^{-m}x + v + q\alpha$$

with $t = 2^{m+l}$. We note that this is nothing but the relation for FV scheme with enlarged plaintext modulus $t' = t2^m$.

Let $c' = (c'_0, c'_1)$ be another ciphertext encrypting another fixed point number $\tilde{x}' = 2^{-m}x'$ $(x' \in \mathbb{Z}_{2^{m+l}})$. Product of fixed point numbers $\tilde{x}$ and $\tilde{x}'$ is defined to be $\tilde{x}\tilde{x}' = 2^{-m}\left[\left\lfloor 2^{-m}xx' \right\rfloor\right]_t$. (Here, $\left[a\right]_t$ denotes the residue of a modulo t.) We want homomorphic version of this computation.

Short calculation shows us that

$$\frac{t2^m}{q}(c_0 + c_1 s)(c'_0 + c'_1 s) = \frac{q}{t}(2^{-m}xx' + t(x'\alpha + x\alpha')) + v'' + 2^m q\alpha'' \tag{1}$$

for some small noise v'' and cyclotomic integer α''. By principle of division, we have $xx' = \left\lfloor 2^{-m}xx' \right\rfloor 2^m + \left[xx'\right]_{2^m}$. Substituting $2^{-m}xx' = \left\lfloor 2^{-m}xx' \right\rfloor + 2^{-m}\left[xx'\right]_{2^m}$ into Equation (1), we

have

$$\begin{aligned}
&\frac{t2^m}{q}(c_0+c_1s)(c_0'+c_1's)\\
&\quad=\frac{q}{t}\Big\{2^{-m}\big[xx'\big]_{2^m}+\big\lfloor 2^{-m}xx'\big\rfloor+t(x'\alpha+x\alpha')\Big\}+v''+2^mq\alpha''\\
&\quad=\frac{q}{t}\Big\{2^{-m}\big[xx'\big]_{2^m}+\big[\big\lfloor 2^{-m}xx'\big\rfloor\big]_t+\big(\big\lfloor 2^{-m}xx'\big\rfloor-\big[\big\lfloor 2^{-m}xx'\big\rfloor\big]_t\big)+t(x'\alpha+x\alpha')\Big\}\\
&\quad\quad+v''+2^mq\alpha''.
\end{aligned}$$

Thus, we see that the product of two ciphertexts c and c', as ciphertexts of the FV scheme with plaintext modulus $t'=t2^m$ (and ciphertext modulus q), is an encryption of

$$w=2^{-m}\big[xx'\big]_{2^m}+\big[\big\lfloor 2^{-m}xx'\big\rfloor\big]_t+\big(\big\lfloor 2^{-m}xx'\big\rfloor-\big[\big\lfloor 2^{-m}xx'\big\rfloor\big]_t\big)+t(x'\alpha+x\alpha').$$

Here we see that w contains the wanted answer $2^m\tilde{x}\tilde{x}'=\big[\big\lfloor 2^{-m}xx'\big\rfloor\big]_t$ in the middle, but it also contains two annoying terms $LG=2^{-m}\big[xx'\big]_{2^m}$ and $UG=\big\lfloor 2^{-m}xx'\big\rfloor-\big[\big\lfloor 2^{-m}xx'\big\rfloor\big]_t+t(x'\alpha+x\alpha')$. We call the former *lower garbage* and the latter *upper garbage*, since LG is the least significant m bits of w and UG is the most significant m bits of w. In order to realize homomorphic multiplication of encrypted fixed point numbers, we will implement some clearing methods of such two types of garbage LG and UG. Suppose here we had cleared LG and UG from (c_0,c_1) to get a new ciphertext (d_0,d_1), which will satisfy

$$\frac{t2^m}{q}(d_0+d_1s)(d_0'+d_1's)=\frac{q}{t}\big[\big\lfloor 2^{-m}xx'\big\rfloor\big]_t+v''+2^mq\alpha''.$$

By dividing both sides by 2^m, we get

$$\frac{t}{q}(d_0+d_1s)(d_0'+d_1's)=\frac{q}{t}2^{-m}\big[\big\lfloor 2^{-m}xx'\big\rfloor\big]_t+2^{-m}v''+q\alpha''$$

as desired.

As clearing methods of the lower and upper garbage, we introduce LowerClear and UpperClear algorithms. The algorithm LowerClear is a variant of arithmetic procedure for computing $\mathrm{msb}_q:\mathbb{Z}_q\to\mathbb{Z}_2$ of [9, 20]. Let $t=2^{m+l}$ and let $w=x+2^mz$ be an element of $\mathbb{Z}_{2^{2m+l}}$ with $x\in\mathbb{Z}_{2^m}$, $z\in\mathbb{Z}_t$. That is, x is the least significant m bits of $(2m+l)$-bit w (lower garbage). We want to clear $x\in\mathbb{Z}_{2^m}$ from w to get 2^mz. The key observation is the following simple fact[9, 20]: if integer w is equal to $b\in\{0,1\}$ mod 2^i then w^2 is equal to the same $b\in\{0,1\}$ mod 2^{i+1} for any integer $i\geq 1$. So, if w has bit decomposition $(b_{2m+l-1},\ldots,b_0)_2$ then by repeating squaring $(2m+l-1)$ times against w, we get an integer w_0 with bit decomposition $(0,\cdots,0,b_0)_2$. LowerClear (w) repeats in this way to extract all lower m bits $b_0,b_1,\cdots,b_{m-1}$ of w in the form of integers $w_0=(0,\cdots,0,b_0)_2, w_1=(0,\cdots,0,b_1,0)_2,\ldots,w_{m-1}=(0,\cdots,0,b_{m-1},0,\cdots,0)_2$ and gets the pure lower part $2^mz=w-\sum_{i=0}^{m-1}w_i$. UpperClear clears the upper garbage by a similar method.

Summarizing, we use the FV scheme with plaintext modulus $t'=2^mt$ with $t=2^{m+l}$ to enable homomorphic evaluation on ciphertexts encrypting fixed point numbers $\tilde{x}=2^{-m}x$ $(x\in Z_{2^{m+l}})$. To clear lower and upper garbage involved in homomorphic multiplication of fixed point numbers, we use LowerClear and UpperClear arithmetic procedures homomorphically against multiplied FV ciphertexts. We call our FHE scheme for fixed point numbers built in this way FHE4FX scheme. Since the FV scheme is semantically secure and fully homomorphic, our FHE4FX (which ciphertext is nothing but a ciphertext of FV scheme with enlarged plaintext modulus) is also semantically secure and fully homomorphic.

Greater-than bit extraction. As an application of the FHE4FX scheme, we treat the problem of comparison of magnitude of two encrypted numbers. Suppose we have two encrypted numbers $\mathsf{Enc}(x_1)$ and $\mathsf{Enc}(x_2)$. Define a bit b to be 1 if $x_1 > x_2$ and to be 0 otherwise. We want to compute an encryption $\mathsf{Enc}(b)$ of the bit b given only ciphertexts $\mathsf{Enc}(x_1)$ and $\mathsf{Enc}(x_2)$ without knowing the secret key. In the literature [13, 12] such problem is tackled by Greater-Than protocol based on (such as) the one given by Golle [10]. Let $D \subset \mathbb{Z}$ be a range that possible x_i's belong to. Their protocol is based on the fact that if $x_1 > x_2$, there exists a positive integer i such that $x_1 = x_2 + i$. To establish security, the protocol needs $O(\left|D\right|)$ encryptions and $O(\left|D\right|)$ homomorphic additions among them and needs an interaction with secret key holder. By using the FHE4FX scheme, we show that one can compute the greater-than bit encryption $\mathsf{Enc}(b)$ given only $\mathsf{Enc}(x_1)$ and $\mathsf{Enc}(x_2)$ in polylogarithmic complexity of $\left|D\right|$, neither knowing the secret key nor interaction with secret key holder.

FHE scheme for floating point numbers. Using the method of greater-than bit extraction by FHE4FX, we will construct a fully homomorphic encryption scheme for floating point numbers, FHE4FL. The floating point number N is described as $N = (-1)^s f 2^e$, where $s \in \{0, 1\}$ is the sign, $f \in [1, 2)$ is the significant and e is the exponent of N. We will use three different (but related) FHE4FX schemes to encrypt each part of s, f and e into a ciphertext $([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}})$.

In computation of floating point numbers $N = (-1)^s f 2^e$, different parts of s, f, e have influence to each other. For example, to add two floating point numbers $N = (-1)^s f 2^e$ and $N' = (-1)^{s'} f' 2^{e'}$, we need to compare e and e' to decide $e > e'$ or not. If so, we will compute $f'' = f + (0.5)^{e-e'} f'$ and if not we will compute $f'' = f' + (0.5)^{e'-e} f$. Since we can homomorphically compute a greater-than bit $e > e'$ as seen above, it is not too difficult to evaluate such process homomorphically, given two encryptions $([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}})$ and $([|s'|]_{\mathsf{s}}, [|f'|]_{\mathsf{f}}, [|e'|]_{\mathsf{e}})$ without knowing the secret key.

Since the FHE4FX scheme is semantically secure and fully homomorphic, the FHE4FL scheme for floating point numbers is also semantically secure and fully homomorphic.

Organization. In Section 2 we recall the notion of FHE and construction of the FV scheme as well as its some basic properties. We construct the FHE4FX scheme for fixed point numbers in Section 3. Then, Section 4 treats the problem of greater-than bit extraction. Finally we construct the FHE4FL scheme for floating point numbers in Section 5.

2 Preliminaries

2.1 Homomorphic Encryption

A homomorphic encryption scheme is a quadruple $\mathsf{HE} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ of probabilistic polynomial time algorithms. Given a security parameter 1^n, Keygen algorithm generates a triple of a public key pk, a secret key sk and an evaluation key evk: $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{Keygen}(1^n)$. Enc algorithm encrypts a plaintext $x \in \{0, 1\}$ into a ciphertext c under a public key pk: $c \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$. Decrypt algorithm decrypts a given ciphertext c into a plaintext x using the secret key sk: $x \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$. Eval algorithm applies a function $f : \{0, 1\}^l \to \{0, 1\}$ to given ciphertexts $c_1, \ldots, c_l$ and outputs a ciphertext c_f using the evaluation key evk : $c_f \leftarrow \mathsf{Eval}(\mathsf{evk}, f, c_1, \ldots, c_l)$.

A homomorphic encryption scheme HE is called *L-homomorphic* for $L = L(n)$ if for any function $f : \{0, 1\}^l \to \{0, 1\}$ given as a circuit of depth L, and for any l bits $x_1, \ldots, x_l$, it holds that

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_{\mathsf{evk}}(f, c_1, \ldots, c_l)) \;=\; f(x_1, \ldots, x_l)$$

for $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x_i)$ $(i = 1, \ldots, l)$ except with a negligible probability.

A homomorphic encryption scheme HE is called *fully homomorphic encryption* (FHE) scheme if it is L-homomorphic for any polynomial function $L = poly(n)$.

2.2 The FV Scheme

Here we briefly recall the FHE scheme by Fan and Vercauteren [7, 16] (FV scheme). The FV scheme is an FHE scheme for congruent integers, instantiating the FHE scheme by Brakerski [2] based on the Ring LWE problem [18].

Let m be a positive integer and let $\Phi_m(X) \in \mathbb{Z}[X]$ be the m-th cyclotomic polynomial. The ring $R = \mathbb{Z}[X]/\Phi_m(X)$ is called the m-th cyclotomic ring. Elements a of cyclotomic ring R are called cyclotomic integers and represented by integer coefficient polynomials $a(X) = \sum_{i=0}^{n-1} a_i X^i$ of degree $< n := \phi(m)$. ($\phi(\cdot)$ denotes the Euler function.) Two cyclotomic integer a and b are added through ordinal polynomial addition: $(a + b)(X) = \sum_{i=0}^{n-1}(a_i + b_i)X^i$. Product of cyclotomic integer a and b is computed as polynomial multiplication followed by reduction via $\Phi_m(X)$: $(a \cdot b)(X) = a(X)b(X) \bmod \Phi_m(X)$.

For cyclotomic integer $a = \sum_{i=0}^{n-1} a_i X^i \in R$, let $\left\|a\right\|_\infty = \max\{\left|a_i\right| : 0 \le i < n\}$ be the infinity norm of a. Let δ be the expansion factor of R, i.e., $\delta = \sup\{\left\|ab\right\|_\infty / (\left\|a\right\|_\infty \left\|b\right\|_\infty) : a, b \in R\}$. It is known that there are cases where δ can be taken to be small polynomials of n. For example, we can take $\delta = n$ for a power of two m. The symbol $\lfloor a \rceil$ denotes the nearest cyclotomic integer (or the (coefficient wise) nearest integer coefficient polynomial) of a.

A ciphertext of FV scheme is a pair of cyclotomic integers in $R_q = \mathbb{Z}_q[X]/\Phi_m(X)$ for ciphertext modulus q and a plaintext is a cyclotomic integer in $R_t = \mathbb{Z}_t[X]/\Phi_m(X)$ for plaintext modulus t. Denote by $\left[\cdot\right]_q$ reduction modulo q into the interval $(-q/2, q/2]$.

We fix an integer base w and let $l_w = \left\lfloor \log_w(q) \right\rfloor + 1$. Any cyclotomic integer $a \in R_q$ can be written as $a = \sum_{i=0}^{l_w-1} a_i w^i$ where $a_i \in R$ has coefficients in the interval $(-w/2, w/2]$. Define $\mathsf{WD}(a) = \left(\left[a_i\right]_w\right)_{i=0}^{l_w-1} \in R^{l_w}$ and $\mathsf{PO}(a) = \left(\left[aw^i\right]_q\right)_{i=0}^{l_w-1} \in R^{l_w}$. As easily verified, $\left\langle \mathsf{WD}(a), \mathsf{PO}(b) \right\rangle \equiv ab \pmod q$, where $\left\langle \boldsymbol{x}, \boldsymbol{y} \right\rangle = \sum_{i=1}^{l} x_i y_i$ denotes the inner product of vectors $\boldsymbol{x} = (x_i)$ and $\boldsymbol{y} = (y_i)$.

Let χ_{key} and χ_{err} be two discrete, bounded probability distributions on R. Constants B_{key} and B_{err} denote the corresponding bounds, respectively: $\chi_{key} < B_{key}$, $\chi_{err} < B_{err}$. The symbol $x \leftarrow \chi$ denotes a random sampling of x according to distribution χ. For a finite set X, the symbol $x \xleftarrow{\mathsf{u}} X$ denotes a uniformly random sampling of x from X.

Parameters. We parameterize FV schemes by two parameter q and t. The parameter q denotes ciphertext modulus. The parameter t denotes plaintext modulus. In this paper we assume q is a power of two and t is a divisor of q for simplicity. Let integer $\Delta = q/t$ be a quotient of q by t.

Scheme Description. The FV scheme consists of the following algorithms.

- Keygen $()$:
  $s \leftarrow \chi_{key}$, $e \leftarrow \chi_{err}$
  $a \xleftarrow{\mathsf{u}} R_q$, $b = \left[-(as + e)\right]_q$
  $\boldsymbol{a} \xleftarrow{\mathsf{u}} R_q^{l_w}$, $\boldsymbol{e} \leftarrow \chi_{err}^{l_w}$, $\boldsymbol{b} = \left[\mathsf{PO}(s^2) - (\boldsymbol{a}s + \boldsymbol{e})\right]_q$
  return $\mathsf{sk} = s$, $\mathsf{pk} = (a, b)$, $\mathsf{evk} = (\boldsymbol{a}, \boldsymbol{b})$.

– $\mathsf{Enc}\ ((a,b),\ x \in R_t)$:
  $u \leftarrow \chi_{key}$, $e_1, e_2 \leftarrow \chi_{err}$
  $c_0 = \big[\Delta x + bu + e_1\big]_q$, $c_1 = \big[au + e_2\big]_q$
  return $c = (c_0, c_1)$.

– $\mathsf{Dec}\ (s,\ c = (c_0, c_1))$:
  return $\big[\big\lfloor \frac{t}{q}\big[c_0 + c_1 s\big]_q \big\rceil\big]_t$.

– $\mathsf{Add}\ (c = (c_0, c_1),\ c' = (c'_0, c'_1))$:
  return $c_{add} = \big(\big[c_0 + c'_0\big]_q, \big[c_1 + c'_1\big]_q\big)$.

– $\mathsf{Mult}\ (c = (c_0, c_1),\ c' = (c'_0, c'_1),\ \gamma = (\boldsymbol{a}, \boldsymbol{b}))$:
  $d_0 = \big[\big\lfloor \frac{t}{q} c_0 c'_0 \big\rceil\big]_q$, $d_1 = \big[\big\lfloor \frac{t}{q}(c_0 c'_1 + c_1 c'_0)\big\rceil\big]_q$, $d_2 = \big[\big\lfloor \frac{t}{q} c_1 c'_1 \big\rceil\big]_q$
  return $c_{mult} = \big(\big[d_0 + \langle \mathsf{WD}(d_2), \boldsymbol{b}\rangle\big]_q, \big[d_1 + \langle \mathsf{WD}(d_2), \boldsymbol{a}\rangle\big]_q\big)$.

It is not difficult to see that the FV scheme is semantically secure under the ring-LWE assumption, that is, a pair $(a, b = -as + e) \in (R_q)^2$ sampled as in the Keygen algorithm is indistinguishable from uniformly randomly selected pair (a, b) from $(R_q)^2$. By standard hybrid argument a ciphertext (c_0, c_1) is also indistinguishable from uniformly random pair over $(R_q)^2$.

Definition 1. *The* inherent noise term v *of* $c = (c_0, c_1)$ *designed for* $x \in R_t$ *is an element* $v \in R$ *of smallest norm* $\big\|v\big\|_\infty$ *satisfying*

$$c_0 + c_1 s = \Delta x + v + q\alpha \tag{2}$$

for some $\alpha \in R$.

For an "initial" ciphertext (c_0, c_1) directly produced by $\mathsf{Enc}((a,b), x)$, we have $c_0 + c_1 s - \Delta x \equiv bu + e_1 + (au + e_2)s \equiv -eu + e_1 + e_2 \pmod q$. So, initial ciphertexts have inherent noise terms $v = -eu + e_1 + e_2$ bounded as

$$\big\|v\big\| \ \le\ V := B_{err}(1 + 2\delta B_{key}). \tag{3}$$

Let

$$V^{\mathsf{FV}}_{max} = \frac{1}{2}\Delta = \frac{1}{2}\Big(\frac{q}{t}\Big). \tag{4}$$

Following lemmas adapted from [7, 16, 4] show some basic properties of the FV scheme.

Lemma 1 (Correctness of FV scheme). *Let* v *be the inherent noise term of* $c = (c_0, c_1)$ *designed for* $x \in R_t$. *If* $\big\|v\big\|_\infty < V^{\mathsf{FV}}_{max}$, *then the decryption works correctly, i.e.,*

$$\mathsf{Dec}(s, c) = \big[x\big]_t = x.$$

Lemma 2 (Additive Noise of FV scheme). *Let* v *and* v' *be inherent noise terms of* c *and* c', *designed for* x *and* $x' \in R_t$, *respectively. Let* v_{add} *be the inherent noise term of* $c_{add} = \mathsf{Add}(c, c')$ *designed for* $\big[x + x'\big]_t \in R_t$. *Then, it is bounded as*

$$\big\|v_{add}\big\|_\infty \le \big\|v\big\|_\infty + \big\|v'\big\|_\infty.$$

Lemma 3 (Multiplicative Noise of FV scheme). *Let v and v' be inherent noise terms of c and c', designed for x and $x' \in R_t$, respectively. Suppose $\|v\|_\infty, \|v'\|_\infty < V$ for some V ($< V^{\mathsf{FV}}_{max}$). Let v_{mult} be the inherent noise term of $c_{mult} \leftarrow \mathsf{Mult}(c, c')$ designed for $[xx']_t \in R_t$. Then, it is bounded as*

$$\|v_{mult}\|_\infty \leq \delta t(2 + 4\delta B_{key})V + \delta^2 B_{key}(2B_{key} + 4t^2) + 2^{-1}\delta l_w w B_{err}.$$

As a corollary of Lemma 2 and Lemma 3, we have:

Corollary 1 (Homomorphic Noise of FV scheme). *Let f be an arithmetic circuit over R_t with L levels of multiplications. Let V be an upper bound of inherent noise terms of input ciphertexts c_i, designed for plaintexts x_i, for all i. Let v_f be the inherent noise term of homomorphically evaluated ciphertext $f(c_i)$ designed for $f(x_i) \in R_t$. Then, it is bounded as*

$$\|v_f\|_\infty \leq C_1^L V + L C_1^{L-1} C_2$$

where

$$C_1 \leq 2\delta t(1 + 2\delta B_{key})$$
$$C_2 \leq 2\delta^2 B_{key}(B_{key} + 2t^2) + 2^{-1}\delta l_w w B_{err}.$$

Let c be a ciphertext of FV scheme. Let L_{dec} be the level of some circuit that evaluates the decryption algorithm $\mathsf{Dec}(c, \cdot)$ with a ciphertext c built-in. By Lemma 1 and Corollary 1, if inequality

$$\Delta = \frac{q}{t} > 2(C_1^{L_{dec}} V + L_{dec} C_1^{L_{dec}-1} C_2) \tag{5}$$

holds, we can homomorphically evaluate the algorithm $\mathsf{Dec}(c, \cdot)$ using encrypted secret keys $\mathsf{Enc}(pk, s)$ and can recrypt the ciphertext c into a more noiseless new ciphertext (bootstrapping). By Lemma 4 of [2], we can implement the algorithm $\mathsf{Dec}(c, \cdot)$ by some circuit of level $L_{dec} = O(\log n)$ which is independent of c. Hence Inequality (5) can be satisfied by taking sufficiently large $q = O(n^{\log n})$ for any ciphertext c in cases where the expansion factor δ can be taken polynomial of n. Thus, the FV scheme will be fully homomorphic under circular security assumption (i.e., $\mathsf{Enc}(\mathsf{pk}, \mathsf{sk} = s)$ does not leak any information about s) by taking sufficiently large $q = O(n^{\log n})$ for cyclotomic ring R with polynomial δ.

3 The Proposed Scheme FHE4FX

In this section we construct an FHE scheme that can homomorphically compute fixed point numbers, using the FV scheme as building blocks.

Let $\tilde{x}$ be a fixed point number that has l bits before point and m bits after point. We encode $\tilde{x}$ by an integer $x \in \mathbb{Z}_{2^{m+l}}$ as usual: $\tilde{x} = 2^{-m}x$. Lt $t = 2^{m+l}$. Addition and multiplication of two fixed point numbers $\tilde{x} = 2^{-m}x$ and $\tilde{y} = 2^{-m}y$ are defined as

$$\tilde{x} + \tilde{y} = 2^{-m}[x + y]_t$$
$$\tilde{x} \cdot \tilde{y} = 2^{-m}\big([\lfloor 2^{-m}[x \cdot y]_{2^m t} \rfloor]_t\big).$$

Here, we see that the sum $\tilde{x} + \tilde{y}$ is encoded by integer $x + y \in \mathbb{Z}_t$. So, homomorphic addition of encrypted fixed point numbers is easy. It can be done merely by using homomorphic addition of underlying FV scheme.

However, the product $\tilde{x} \cdot \tilde{y}$ is more complicated. As seen above, it is encoded by integer $\left[\left\lfloor 2^{-m}\left[x \cdot y\right]_{2^m t}\right\rfloor\right]_t$, that results from m-bit right shift of integer product $\left[x \cdot y\right]_{2^m t}$. Now our problem is distinguished: How can we homomorphically compute m-bit right shift of given encrypted $(2m + l)$-bit integers?

Let $\mathsf{FV}(2^m t) = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{Mult})$ be the FV scheme of ciphertext modulus q and plaintext modulus $2^m t$. Using $\mathsf{FV}(2^m t)$, we construct a fully homomorphic encryption scheme for fixed point numbers, FHE4FX, that can homomorphically compute m-bit right shift of given encrypted encoding integers.

Parameters. The FHE4FX scheme is parameterized by three parameters q, m and l. The parameter q denotes ciphertext modulus. The parameter l denotes bit-length of fixed point number before point and the parameter m denotes bit-length of fixed point number after point. Let $t = 2^{m+l}$. We assume q, m and l are all powers of two and t is a divisor of q. Let integer $\Delta = q/t$ be a quotient of q by t.

Scheme Description. A fixed point number $\tilde{x} = 2^{-m}x$ to be encrypted is encoded by an integer $x \in \mathbb{Z}_t$ which we identify with a constant polynomial $x \in R_t$ in the cyclotomic ring. The FHE4FX scheme consists of the following algorithms.

- $\mathsf{Keygen}\ ()$:
  return $(\mathsf{sk} = s, \mathsf{pk} = (a, b), \mathsf{evk} = (\boldsymbol{a}, \boldsymbol{b})) \leftarrow \mathsf{FV}(2^m t).\mathsf{Keygen}()$

- $\mathsf{Enc}\ (\mathsf{pk} = (a, b),\ x \in R_t)$:
  return $c = (c_0, c_1) \leftarrow \mathsf{FV}(2^m t).\mathsf{Enc}((a, b), x)$.

- $\mathsf{Dec}\ (s,\ c)$:
  return $\left[\mathsf{FV}(2^m t).\mathsf{Dec}(s, c)\right]_t$.

- $\mathsf{Add}\ (c,\ c')$:
  return $c_{add} \leftarrow \mathsf{FV}(2^m t).\mathsf{Add}(c, c')$.

- $\mathsf{Mult}\ (c,\ c',\ \mathsf{evk} = \gamma)$:
  $\tilde{c} \leftarrow \mathsf{FV}(2^m t).\mathsf{Mult}(c, c', \gamma)$
  $d = (d_0, d_1) \leftarrow \mathsf{LowerClear}(\tilde{c})$
  $e = \left(\left[\left\lfloor \frac{1}{2^m} d_0 \right\rceil\right]_q, \left[\left\lfloor \frac{1}{2^m} d_1 \right\rceil\right]_q\right)$
  return $c_{mult} \leftarrow \mathsf{UpperClear}(e)$.

- $\mathsf{LowerClear}\ (c)$:
  $d \leftarrow \mathsf{FV}(2^m t).\mathsf{Enc}_{\mathsf{pk}}(0)$
  For $i \in [1..m)$]:
    $d_i \leftarrow \mathsf{FV}(2^m t).\mathsf{Add}(c, -d)$
    For $j \in [1..(2m + l - i)]$:
      $d_i \leftarrow \mathsf{FV}(\frac{2^m t}{2^{i-1}}).\mathsf{Mult}(d_i, d_i)$
    $d \leftarrow \mathsf{FV}(2^m t).\mathsf{Add}(d, d_i)$
  return $\mathsf{FV}(2^m t).\mathsf{Add}(c, -d)$.

- $\mathsf{UpperClear}\ (c)$:

$d \leftarrow \mathsf{FV}(2^m t).\mathsf{Enc}_{\mathsf{pk}}(0)$
For $i \in [1..(m+l)]$:
  $d_i \leftarrow \mathsf{FV}(2^m t).\mathsf{Add}(c, -d)$
  For $j \in [1..(2m+l-i)]$:
    $d_i \leftarrow \mathsf{FV}(\frac{2^m t}{2^{i-1}}).\mathsf{Mult}(d_i, d_i)$
  $d \leftarrow \mathsf{FV}(2^m t).\mathsf{Add}(d, d_i)$
return d.

Note that ciphertexts of the proposed FHE4FX scheme of parameter (q, m, l) are nothing but the ciphertexts of FV scheme with parameter $(q, 2^m t)$ and encryption/decryption algorithms are the same as corresponding algorithms of $\mathsf{FV}(2^m t)$ scheme. (More precisely, the decryption algorithm is slightly lighter than the original one of $\mathsf{FV}(2^m t)$ scheme, since the $\mathsf{FHE4FX}(q, m, l)$ scheme only recovers $(l+m)$-bit integers rather than $(l+2m)$-bit integers in the $\mathsf{FV}(2^m t)$ scheme.) Especially, the FHE4FX scheme is also semantically secure and fully homomorphic (if parameters are suitably selected just as in the $\mathsf{FV}(2^m t)$ scheme) because the FV scheme is so. The difference is in the way of homomorphic evaluation.

Let c be a ciphertext of the proposed FHE4FX scheme of parameter (q, m, l). By the definition of Enc algorithm, initially it is produced as an encryption of some constant polynomial $x \in R_t$ that encodes fixed point number $\tilde{x} = 2^{-m}x$ using $\mathsf{FV}(2^m t)$ scheme. Here note that bit length of plaintext integer x is only $l + m$ (l bits before point and m bits after point), but $\mathsf{FV}(2^m t)$ scheme treats much longer plaintext integer of $l + 2m$ bits. In fact, Dec algorithm only returns the least $l + m$ bits of recovered integer by $\mathsf{FV}(2^m t)$. That is, the FHE4FX scheme has m bits more in plaintext space than finally required for decryption. As a distinctive feature, FHE4FX scheme uses this room of m bits in plaintext space in order to homomorphically compute m-bit right-shift of encrypted integer of $l + 2m$ bits.

To homomorphically compute such m-bit right-shift, algorithms LowerClear and UpperClear are important tools.

Suppose a ciphertext c is an $\mathsf{FV}(2^m t)$-encryption of some $l + 2m$-bit integer $w = x + 2^m z \in R_{2^m t}$ (x is the least m bits of w and z is the significant $(l + m)$-bit of w). Before m-bit right-shifting w homomorphically in the ciphertext c, we need to clear the least m-bit integer x of w, because without it the term $2^{-m}x$ should cause a significant noise in the resulting ciphertext. As verified in Proof of Lemma 5 in the below, the least m-bit x (lower garbage) of w is cleared by using the following LowerClearPlain algorithm.

– LowerClearPlain $(w = x + 2^m z)$:
  $g \leftarrow 0$
  For $i \in [1..m]$:
    $w_i \leftarrow w - g$
    $w_i = \left(\left(\frac{1}{2}\right)^{i-1} w_i \right)^{2^{2m+l-i}} \cdot 2^{i-1}$ $\{w_i$ is divisible by 2^{i-1} (proof of Lemma 6)$\}$
    $g = g + w_i$
  return $w - g$ $\{= 2^m z\}$

As directly verified, the above LowerClear (c) procedure (for c that encrypts $w = x + 2^m z$) is a homomorphic version of LowerClearPlain (w) procedure, that computes $2^m z$ given $w = x + 2^m z$ (Lemma 1). Here we add remark about this. Suppose a constant polynomial $x \in R_{2^m t}$ is divisible

by 2^i ($i \geq 0$). Then, its encryption $c = (c_0, c_1)$ by $\mathsf{FV}(2^m t)$ (=$\mathsf{FHE4FX}(q, m, l)$) scheme will satisfy

$$\begin{aligned} c_0 + c_1 s &= \frac{q}{2^m t} x + v + q\alpha \\ &= \frac{q}{2^{m-i} t} \frac{x}{2^i} + v + q\alpha \end{aligned}$$

with some small noise $v \in R$ and some $\alpha \in R$. This means that when plaintext integer x is divisible by 2^i, its ciphertext $c = (c_0, c_1)$ is nothing but the ciphertext of $\frac{x}{2^i}$ w.r.t. $\mathsf{FV}(2^{m-i}t)$. So the homomorphic version of the step

$$w_i = \left(\left(\tfrac{1}{2}\right)^{i-1} w_i \right)^{2^{2m+l-i}} \cdot 2^{i-1}$$

in $\mathsf{LowerClearPlain}$ corresponds to the step

For $j \in [1..(2m + l - i)]$:
  $d_i \leftarrow \mathsf{FV}(\frac{2^m t}{2^{i-1}}).\mathsf{Mult}(d_i, d_i)$

in $\mathsf{LowerClear}$. The resulting ciphertext d_i will be treated as ciphertext in $\mathsf{FV}(2^m t)$.

Now suppose we have cleared the lower garbage x from the ciphertext c that encrypts $w = x + 2^m z$, resulting a ciphertext d that encrypts $2^m z$. Then we simply divide each coefficients of d by 2^m and we get a ciphertext e of $u = z + ty$ with some integer y. (Recall that our plaintext space is $2m + l$ bits. So the term ty should appear with some y.) Note that y is the significant m bits of $z + ty$ (upper garbage). As in the case of $\mathsf{LowerClear}$, we use algorithm $\mathsf{UpperClear}$ to clear the upper garbage y from $u = z + ty$, that is a homomorphic version of the following $\mathsf{UpperClearPlain}$, as directly verified.

– $\mathsf{UpperClearPlain}$ $(u = z + ty)$:
  $r \leftarrow 0$
  For $i \in [1..(m + l)]$:
    $u_i \leftarrow u - r$
    $u_i = \left(\left(\tfrac{1}{2}\right)^{i-1} u_i \right)^{2^{2m+l-i}} \cdot 2^{i-1}$ $\{u_i$ is divisible by 2^{i-1} (proof of Lemma 6)$\}$
    $r = r + u_i$
  return r $\{= z\}$

Now we turn to formal treatment. For clarity, we re-define the inherent noise term:

Definition 2. *The* inherent noise term v *of* $c = (c_0, c_1)$ *designed for* $x \in R_{2^m t}$ *is the term* v *of smallest norm* $\|v\|_\infty$ *satisfying*

$$c_0 + c_1 s = \Delta 2^{-m} x + v + q\alpha \tag{6}$$

for some $\alpha \in R$.

This coincides to the original definition of the inherent noise term as ciphertexts of $\mathsf{FV}(2^m t)$ scheme.

Let

$$V_{max} = \frac{1}{2} \Delta 2^{-m} = \frac{1}{2} \Big(\frac{q}{2^m t} \Big). \tag{7}$$

By Lemma 1, we get immediately:

Lemma 4. *Let v be the inherent noise term of $c = (c_0, c_1)$ designed for $x \in R_t$. If $\|v\|_\infty < V_{max}$, then the decryption works correctly, i.e.,*

$$\mathsf{Dec}(s, c) = \left[x\right]_t = x.$$

Next we examine correctness of the LowerClear and UpperClear algorithms. For C_1 and C_2 given in Corollary 1, it holds that:

Lemma 5 (Lower Clear). *Let $\tilde{v}$ be the inherent noise term of some ciphertext $\tilde{c}$ designed for constant polynomial $w = x + 2^m z \in R_{2^m t}$ with $x \in R_{2^m}$ and $z \in R_t$. Let c be a ciphertext output by $c \leftarrow \mathsf{LowerClear}(\tilde{c})$. Let v be the inherent noise term of the ciphertext c as designed for constant polynomial $2^m z\,(= w - x = w - \left[w\right]_{2^m}) \in 2^m R_{2^m t}$ Then, it holds that*

$$\|v\|_\infty \;\le\; \|\tilde{v}\|_\infty + V_{\mathsf{LC}},$$

where $V_{\mathsf{LC}} = C_1^{2m+l-1} V + L C_1^{2m+l-2} C_2$.

Proof. Since algorithm LowerClear is a homomorphic version of algorithm LowerClearPlain, first, we show that LowerClearPlain computes $2^m z$ given $w = x + 2^m z$, that is, it clears up the lower garbage x of w.

The key observation is the following simple fact [9, 20]: if integer w is equal to a bit $b \in \{0, 1\}$ mod 2^i then w^2 is equal to the same bit b mod 2^{i+1} for any integer $i \ge 1$. So, if w has bit decomposition $(b_{2m+l-1}, \ldots, b_0)_2$ then by repeating squaring $(2m + l - 1)$ times against it, we get an integer w_0 with bit decomposition $(0, \cdots, 0, b_0)_2$. By repeating the procedure for $w - w_0 = (b_{2m+l-1}, \cdots, b_1, 0)_2$, we get an integer w_1 with bit decomposition $(0, \cdots, 0, b_1, 0)_2$: Multiply $w - w_0$ by $\frac{1}{2}$ to get $(0, b_{2m+l-1}, \cdots, b_2, b_1)_2$, repeat squaring $(2m + l - 2)$ times against it to get $(0, \cdots, 0, b_1)_2$, and multiply back it by 2 to get $(0, \cdots, 0, b_1, 0)_2$. LowerClearPlain (w) repeats in this way to extract all least m bits $b_0, b_1, \cdots, b_{m-1}$ of w in the form of integers $w_0 = (0, \cdots, 0, b_0)_2, w_1 = (0, \cdots, 0, b_1, 0)_2, \ldots, w_{m-1} = (0, \cdots, 0, b_{m-1}, 0, \cdots, 0)_2$ and gets the least m-bit x of w as $x = \sum_{i=0}^{m-1} w_i$. Then we get desired $2^m z = w - x$.

Next we estimate strength of noise occurred by homomorphic evaluation of LowerClearPlain algorithm, i.e., LowerClear algorithm. Since LowerClear has $2m + l - 1$ levels of nested multiplications, by Lemma 1 we get the claimed bound V_{LC}. □

Lemma 6 (Upper Clear). *Let $\tilde{v}$ be the inherent noise term of some ciphertext $\tilde{c}$ designed for constant polynomial $u = z + ty \in R_{2^m t}$ with $z \in R_t$ and $y \in R_{2^m}$. Let c be a ciphertext resulting by $c \leftarrow \mathsf{UpperClear}(\tilde{c})$. Let v be the inherent noise term of the ciphertext c as designed for constant polynomial $z = \left[u\right]_t \in R_t$. Then, it holds that*

$$\|v\|_\infty \;\le\; \|\tilde{v}\|_\infty + V_{\mathsf{UC}},$$

where $V_{\mathsf{UC}} = C_1^{2m+l-1} V + L C_1^{2m+l-2} C_2$.

Proof. Similar as the proof of Lemma 5. Note that UpperClear algorithm also has $2m + l - 1$ levels of nested multiplications as LowerClear algorithm. So the resulting bound V_{UC} is the same as V_{LC}. □

Proposition 1 (Additive Noise of FHE4FX scheme). *Let v and v' be inherent noise terms of c and c', designed for x and $x' \in R_t$, respectively. Let v_{add} be the inherent noise term of $c_{add} = \mathsf{Add}(c, c')$ designed for $\left[x + x'\right]_t \in R_t$. Then, it is bounded as*

$$\|v_{add}\|_\infty \le \|v\|_\infty + \|v'\|_\infty.$$

Proof. This is a restatement of Lemma 2.□

Proposition 2 (Multiplicative Noise of FHE4FX scheme). *Let v and v' be inherent noise terms of c and c', designed for x and $x' \in R_{2^{m+l}}$, respectively. Suppose $\|v\|_\infty, \|v'\|_\infty < V$. Let v_{mult} be the inherent noise term of $c_{mult} \leftarrow \mathsf{Mult}(c, c', \mathsf{evk} = \gamma)$ designed for $\big[\lfloor 2^{-m}\big[xx'\big]_{2^m t}\rfloor\big]_t \in R_t$. Then, it is bounded as*

$$\|v_{mult}\|_\infty \le \delta t(2 + 4\delta B_{key})V + \delta^2 B_{key}(B_{key} + 4 \cdot 2^m t^2) + l_w \delta w B_{err} + 2^{-m} V_{\mathsf{LC}} + V_{\mathsf{UC}}.$$

Proof. We use the notation used in $\mathsf{Mult}(c, c', \mathsf{evk} = \gamma)$ algorithm. For $\tilde{c} = \mathsf{FV}(2^m t).\mathsf{Mult}(c, c', \gamma)$, by Lemma 3, we have

$$\tilde{c}_0 + \tilde{c}_1 s = \Delta 2^{-m}\big[xx'\big]_{2^m t} + v + q\alpha$$

with

$$\|v\|_\infty \le \delta 2^m t(2 + 4\delta B_{key})V + \delta^2 B_{key}(2B_{key} + 4 \cdot 2^{2m} t^2) + 2^{-1}\delta l_w w B_{err}$$

and with some $\alpha \in R$.

Then, by Lemma 5, $d = (d_0, d_1) \leftarrow \mathsf{LowerClear}(\tilde{c})$ satisfies

$$d_0 + d_1 s = \Delta 2^{-m}\big(\lfloor \frac{\big[xx'\big]_{2^m t}}{2^m} \rfloor 2^m\big) + v + w + q\alpha'$$

with

$$\|w\|_\infty \le V_{\mathsf{LC}}$$

and with some $\alpha' \in R$.

Then, dividing by 2^m, we get

$$\begin{aligned}\frac{1}{2^m}d_0 + \frac{1}{2^m}d_1 s &= \Delta 2^{-m}\Big(\lfloor \frac{\big[xx'\big]_{2^m t}}{2^m} \rfloor\Big) + \frac{v}{2^m} + \frac{w}{2^m} + \frac{q}{2^m}\alpha' \\ &= \Delta 2^{-m}\Big(\lfloor \frac{\big[xx'\big]_{2^m t}}{2^m} \rfloor + t\alpha'\Big) + \frac{v}{2^m} + \frac{w}{2^m}\end{aligned}$$

So, the rounded $e = \Big(\big[\big\lfloor \frac{1}{2^m} d_0 \big\rceil\big]_q, \big[\big\lfloor \frac{1}{2^m} d_1 \big\rceil\big]_q\Big)$ satisfies:

$$e_0 + e_1 s = \Delta 2^{-m}\Big(\lfloor \frac{\big[xx'\big]_{2^m t}}{2^m} \rfloor + t\alpha'\Big) + \frac{v}{2^m} + \frac{w}{2^m} + w'$$

with

$$\|w'\|_\infty \le \frac{1}{2}(1 + \delta B_{key}).$$

Finally, applying Lemma 6, we see $c_{mult} = \mathsf{UpperClear}(e)$ satisfies

$$c_{mult,0} + c_{mult,1} s = \Delta 2^{-m} \cdot \big[\lfloor \frac{\big[xx'\big]_{2^m t}}{2^m} \rfloor\big]_t + \frac{v}{2^m} + \frac{w}{2^m} + w' + w'' + q\beta$$

with

$$\|w''\|_\infty \le V_{\mathsf{UC}}$$

and with some $\beta \in R$.

The accumulated noise $z = \frac{v}{2^m} + \frac{w}{2^m} + w' + w''$ is bounded as:

$$\begin{aligned}\|z\|_\infty &\le 2^{-m}\big(\delta 2^m t(2+4\delta B_{key})V + \delta^2 B_{key}(2B_{key} + 4\cdot 2^{2m}t^2) + 2^{-1}\delta l_w w B_{err}\big) \\ &\quad + 2^{-m}V_{\mathsf{LC}} + 2^{-1}(1+\delta B_{key}) + V_{\mathsf{UC}} \\ &\le \delta t(2+4\delta B_{key})V + \delta^2 B_{key}(B_{key} + 4\cdot 2^m t^2) + l_w \delta w B_{err} + 2^{-m}V_{\mathsf{LC}} + V_{\mathsf{UC}}.\end{aligned}$$

□

By Proposition 1, 2 we have

Theorem 1. *The* FHE4FX *scheme of parameter q, l, m can fully homomorphically compute additions and multiplications of encrypted fixed point numbers $\tilde{x} = 2^{-m}x$ for $x \in \mathbb{Z}_{2^{m+l}}$ with suitable choice of parameters that makes the underlying* FV *scheme (of parameter $q, 2^{2m+l}$) fully homomorphic.*

Here, addition of fixed point numbers $\tilde{x} = 2^{-m}x$ and $\tilde{y} = 2^{-m}y$ is such that

$$\tilde{x} + \tilde{y} = 2^{-m}\big[x + y\big]_{2^{m+l}}$$

and their multiplication is such that

$$\tilde{x}\cdot\tilde{y} = 2^{-m}\big(\big[\lfloor 2^{-m}\big[x\cdot y\big]_{2^{2m+l}}\rfloor\big]_{2^{m+l}}\big).$$

Efficiency. We estimate efficiency of homomorphic operations of FHE4FX scheme of parameter q, l, m. Obviously, addition $\mathsf{Add}(c, c')$ is performed only one addition of $\mathsf{FV}(q, 2^m t)$ scheme.

The multiplication $\mathsf{Mult}(c, c', \gamma)$ is much more heavy. We estimate its complexity in terms of "multiplicative depth" and "multiplicative number". Here, the multiplicative depth means the required depth of nested multiplications of the underlying FV scheme to perform the target operation. It determines noise growth occurred by doing the target operation and larger noise requires larger ciphertext (since it requires larger ciphertext modulus). Thus, the multiplicative depth dominates space complexity of the target operation. On a while, the multiplicative number means the required total number of multiplications of the underlying FV scheme to perform the target operation. It dominates time complexity of the target operation.

By inspection it is clear that complexity of $\mathsf{Mult}(c, c', \gamma)$ operation is dominated by $\mathsf{UpperClear}(e)$. The multiplicative depth of UpperClear is $2m+l-1$ and the multiplicative number of UpperClear is $(2m+l-1) + \cdots + (m+l) = \frac{1}{2}(3m+2l-1)(m+l)$.

Thus, roughly estimated, the space and time complexity of multiplication $\mathsf{Mult}(c, c', \gamma)$ is linear and quadratic to the logarithmic of precision of fixed point numbers, respectively.

4 Greater-Than Bit Extraction

Here, as an application of our FHE4FX scheme, we consider the problem of comparison of magnitude of two encrypted numbers. Suppose we have two encrypted numbers $\mathsf{Enc}(x_1)$ and $\mathsf{Enc}(x_2)$. Define a bit b to be 1 if $x_1 > x_2$ and to be 0 otherwise. We want to compute an encryption $\mathsf{Enc}(b)$ of the bit b given only ciphertexts $\mathsf{Enc}(x_1)$ and $\mathsf{Enc}(x_2)$ without knowing the secret key. In the literature [13, 12] such problem is tackled by Greater-Than protocol based on (such as) the one given by Golle [10]. Let $D \subset \mathbb{Z}$ be a range that possible x_i's belong to. Their protocol is based on the fact that if $x_1 > x_2$, there exists a positive integer i such that $x_1 = x_2 + i$. To

establish security, the protocol needs $O(|D|)$ encryptions and $O(|D|)$ homomorphic additions among them and needs an interaction with secret key holder. By using the FHE4FX scheme, we can compute the greater-than bit encryption $\mathsf{Enc}(b)$ given only $\mathsf{Enc}(x_1)$ and $\mathsf{Enc}(x_2)$ in poly-logarithmic complexity of $|D|$, neither knowing the secret key nor interaction with secret key holder.

First, we show a homomorphic procedure $\mathsf{MSb}\ (c)$ for computing an encryption of the most significant bit $\mathsf{MSb}(x)$ given only an encryption $c = \mathsf{Enc}(x)$ of x under FHE4FX scheme. Here we note that our $\mathsf{MSb}(x)$ is integer 0 or 1 in the multi-bit plaintext space $R_{2^{m+l}}$ (different from msb procedure of [9, 20] that outputs a bit in R_2).

Let c be an encryption of some point number $\tilde{x} = 2^{-m}x$ by the $\mathsf{FHE4FX}(q, l, m)$ scheme. Given c, we want to compute an encryption of the most significant bit of $x \in R_t$ $(t = 2^{m+l})$ without knowing the secret key. The point is that one can multiply any fixed point numbers by 0.5 in $\mathsf{FHE4FX}(q, l, m)$ scheme. First it homomorphically extracts the msb b of x in the form $y = (b, 0, \ldots, 0)_2$ using similar method as $\mathsf{UpperClear}(c)$. Then, it repeats taking half of y in $\mathsf{Enc}(y)$ homomorphically, i.e., computing $0.5 \times \mathsf{Enc}(y)$, until we get an encryption of $\mathsf{MSb}(x) = (0.5)^{l-1}y = (0, \ldots, 0, b, 0, \ldots, 0)_2$ (here, 0 repeat m times after b) that encodes a number $b.0$ In the following $\mathsf{FHE4FX.Add}(c, c')$ (or $\mathsf{FHE4FX.Mult}(c, c')$) is simply written as $c + c'$ (or $c \cdot c'$, respectively).

- $\mathsf{MSb}\ (c, k = 0)$:
  {returns the encryption of $b.0$ where b is the $(l + m - k)$-th bit of x. It must be $0 \le k < l$.}
  $d \leftarrow \mathsf{FV}(2^m t).\mathsf{Enc}_{\mathsf{pk}}(0)$
  For $i \in [1..(m + l - k)]$:
    $d_i \leftarrow \mathsf{FV}(2^m t).\mathsf{Add}(c, -d)$
    For $j \in [1..(2m + l - i)]$:
      $d_i \leftarrow \mathsf{FV}(\frac{2^m t}{2^{i-1}}).\mathsf{Mult}(d_i, d_i)$
    $d \leftarrow \mathsf{FV}(2^m t).\mathsf{Add}(d, d_i)$
  $d \leftarrow d_{m+l-k}$
  $h \leftarrow \mathsf{FHE4FX.Enc}(2^{m-1})$ {h encrypts "0.5"}
  Repeat $l - 1 - k$ times:
    $d \leftarrow h \cdot d$ {taking half of d}
  return d.

Using the homomorphic procedure $\mathsf{MSb}(\cdot)$, homomorphic computation $\mathsf{GTb}\ (c_1, c_2)$ of an encryption of greater-than bit b of given $c_1 = \mathsf{Enc}(x_1)$ and $c_2 = \mathsf{Enc}(x_2)$ is straightforward. Basically it simply compute $\mathsf{MSb}(c_2 - c_1)$. If x_1 and x_2 represent signed numbers, we need also to take care of their signs. If x_1 and x_2 have the same sign, the greater-than bit b is equal to $\mathsf{MSb}(x_2 - x_1)$. Otherwise, $b = \mathsf{MSb}(x_2)$. In the following we describe $\mathsf{GTb}\ (c, c_1)$ for signed numbers x_1, x_2 that will be needed in Section 5.

- $\mathsf{GTb}\ (c_1, c_2)$: {returns an encryption of $b.0$ where a bit b is 1 if $x_1 > x_2$ or 0 otherwise}
  $one \leftarrow \mathsf{FHE4FX.Enc}(2^m)$ {ciphertext *one* encrypts "1.0"}
  $d_1 \leftarrow \mathsf{MSb}(c_2 - c_1)$, $d_2 \leftarrow \mathsf{MSb}(c_2)$
  $s_0 \leftarrow \mathsf{MSb}(c_1), s_1 = d_2$
  $same\_sign \leftarrow s_0 \cdot s_1 + (one - s_0) \cdot (one - s_1)$ {*same_sign* encrypts 1.0 if x_1 and x_2 have the same sign, or encrypts 0.0 otherwise}
  $d \leftarrow same\_sign \cdot d_1 + (one - same\_sign) \cdot d_2$
  return d.

To compute $\mathsf{MSb}(c)$ we need $(m+l)^2$ Mult operations of the underlying FV scheme. So to compute $\mathsf{GTb}(c_1, c_2)$ we need roughly $3(m+l)^2$ Mult operations of the FV scheme, that is polylogarithmic complexity polylog($\left|D\right|$) of the number of possible plaintexts $\left|D\right| = 2^{m+l}$.

Here, we remark that using the $\mathsf{GTb}(c_1, c_2)$ procedure, one can homomorphically compare magnitude of two encrypted fixed point numbers $c_1 \leftarrow \mathsf{Enc}(x_1)$ and $c_2 \leftarrow \mathsf{Enc}(x_2)$, and output a ciphertext c that encrypts the greater one.

- GT (c_1, c_2) : {returns an encryption of x_1 if $x_1 > x_2$ or an encryption of x_2 otherwise}
  $one \leftarrow \mathsf{FHE4FX.Enc}(2^m)$ {ciphertext *one* encrypts "1.0"}
  $b \leftarrow \mathsf{GTb}(c_1, c_2)$
  $c \leftarrow b \cdot c_1 + (one - b) \cdot c_2$
  return c.

Note that even if one sees the three ciphertexts c_1, c_2 and c, he/she cannot know which plaintext x_i c is actually encrypting, without knowing the secret key.

5 The Proposed Scheme FHE4FL

In this section, using the method of greater-than bit extraction by $\mathsf{FHE4FX}$, we construct a fully homomorphic encryption scheme for floating point numbers, $\mathsf{FHE4FL}$.

A floating point number N is written as $N = (-1)^s f 2^e$, where $s \in \{0, 1\}$ is the sign, $f \in [1, 2)$ is the significand and $e \in \mathbb{Z}$ is the exponent of N.

Product $N'' = (-1)^{s''} f'' 2^{e''}$ of two floating point numbers $N = (-1)^s f 2^e$ and $N' = (-1)^{s'} f' 2^{e'}$ is computed as follows. The new sign s'' is XOR of signs s and s': $s'' = (1-s)s' + (1-s')s$. Significands are multiplied and exponents are added, respectively: $f'' = ff'$ and $e'' = e + e'$. If $f'' > 2$ then normalize the result as $f'' := f''/2$ and $e'' = e'' + 1$.

Computation of sum $N'' = (-1)^{s''} f'' 2^{e''}$ of two floating point numbers $N = (-1)^s f 2^e$ and $N' = (-1)^{s'} f' 2^{e'}$ is more complicated since we need to adjust the point position and to consider several cases as follows.

- When $s = s'$ and $e > e'$, let $s'' = s$, $f'' = f + 2^{e'-e} f'$, $e'' = e$. If $f'' > 2$ normalize it as $f'' := f''/2$ and $e'' = e'' + 1$.
- When $s = s'$ and $e \leq e'$, let $s'' = s$, $f'' = 2^{e-e'} f + f'$, $e'' = e'$. If $f'' > 2$ normalize it as $f'' := f''/2$ and $e'' = e'' + 1$.
- When $s \neq s'$ and $e > e'$, let $s'' = s$, $f'' = f - 2^{e'-e} f'$, $e'' = e$, While $f'' < 1$, do $f'' := 2f''$, $e'' = e'' - 1$.
- When $s \neq s'$ and $e < e'$, let $s'' = s'$, $f'' = f' - 2^{e-e'} f$, $e'' = e'$, While $f'' < 1$, do $f'' := 2f''$, $e'' = e'' - 1$.
- When $s \neq s'$, $e = e'$ and $f > f'$, let $s'' = s$, $f'' = f - f'$, $e'' = e$, While $f'' < 1$, do $f'' := 2f''$, $e'' = e'' - 1$.
- When $s \neq s'$, $e = e'$ and $f < f'$, let $s'' = s'$, $f'' = f' - f$, $e'' = e'$, While $f'' < 1$, do $f'' := 2f''$, $e'' = e'' - 1$.
- When $s \neq s'$, $e = e'$ and $f = f'$, let $s'' = 0$, $f'' = 1.0$, $e'' = 0$.

5.1 The scheme FHE4FL

Parameters. Recall a floating point number N is written as $N = (-1)^s f 2^e$, where $s \in \{0, 1\}$ is the sign, $f \in [1, 2)$ is the significand and $e \in \mathbb{Z}$ is the exponent of N. The $\mathsf{FHE4FL}$ scheme

is parameterized by three parameters q, m and l. The parameter q is the ciphertext modulus. Parameters m and l are such that $1+m$ is the bit-length of significand f and l is the bit-length of exponent e (excluding the sign-bit).

Building blocks. The scheme $\mathsf{FHE4FL}$ encrypts each part of sign s, significand f and exponent e of a floating point number N into a triple of ciphertexts $([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}})$, using three $\mathsf{FHE4FX}$ schemes FXs, FXf and FXe that share a same ciphertext modulus q and a same key pair $(\mathsf{pk}, \mathsf{sk})$.

- $\mathsf{FXs} : \mathsf{FHE4FX}(q, 1, 0)$ scheme for the sign s
- $\mathsf{FXf} : \mathsf{FHE4FX}(q, 2, m)$ scheme for the significand f
- $\mathsf{FXe} : \mathsf{FHE4FX}(q, l+1, 1)$ scheme for the exponent e

Although bit-length of the significand f is $1+m$, we use $\mathsf{FHE4FX}(q, 2, m)$ scheme for f to take care of carries that occurs among addition $f + f'$. Similarly, we use $\mathsf{FHE4FX}(q, l+1, 1)$ scheme for the exponent e of bit-length l, taking care of its sign.

Scheme Description.

- Keygen () :
  return $(\mathsf{sk} = s, \mathsf{pk} = (a, b), \mathsf{evk} = (\boldsymbol{a}, \boldsymbol{b})) \leftarrow \mathsf{FXs.Keygen}()$.
  { The key $(\mathsf{sk}, \mathsf{pk}, \mathsf{evk})$ will be shared among the three schemes FXs, FXf and FXe. }

- Enc $(\mathsf{pk},\ N = (s, f, e) \in R_2 \times R_{2^{2+m}} \times R_{2^{l+2}})$:
  $[|s|]_{\mathsf{s}} \leftarrow \mathsf{FXs.Enc}(\mathsf{pk}, s), \quad [|f|]_{\mathsf{f}} \leftarrow \mathsf{FXf.Enc}(\mathsf{pk}, f), \quad [|e|]_{\mathsf{e}} \leftarrow \mathsf{FXe.Enc}(\mathsf{pk}, e)$
  return $c = ([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}})$.

- Dec $(\mathsf{sk},\ c = ([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}}))$:
  $s \leftarrow \mathsf{FXs.Dec}(\mathsf{sk}, [|s|]_{\mathsf{s}}), \quad f \leftarrow \mathsf{FXf.Dec}(\mathsf{sk}, [|f|]_{\mathsf{f}}), \quad e \leftarrow \mathsf{FXe.Dec}(\mathsf{sk}, [|e|]_{\mathsf{e}})$
  return $N = (s, f, e)$.

Conversion. First note that we can publicly and efficiently convert ciphertexts, keeping its underlying plaintext unchanged, between $\mathsf{FHE4FX}$ schemes that share a same ciphertext modulus q and a same key pair $(\mathsf{pk}, \mathsf{sk})$. In fact, suppose two schemes $\mathsf{FHE4FX}(q, l, m)$ and $\mathsf{FHE4FX}(q, l', m')$ share a same ciphertext modulus q and a same key pair $(\mathsf{pk}, \mathsf{sk})$. Let $c = (c_0, c_1)$ be a ciphertext in $\mathsf{FHE4FX}(q, l, m)$ scheme that encrypts $x \in R_t$:

$$c_0 + c_1 s \equiv \frac{q}{2^{2m+l}} x + v \pmod{q}.$$

Multiplying c by $2^{2(m-m')+(l-l')}$ homomorphically, we get a new ciphertext $d = (d_0, d_1)$ satisfying

$$d_0 + d_1 s \equiv \frac{q}{2^{2m'+l'}} x + v' \pmod{q}$$

that encrypts $x \in R_{t'}$ as a ciphertext in $\mathsf{FHE4FX}(q, l', m')$ scheme. Here note that we can multiply c by $2^{2(m-m')+(l-l')}$ homomorphically, even if $2(m-m')+(l-l')$ is a negative integer, since we are using the $\mathsf{FHE4FX}$ scheme that can treat an encryption of "0.5".

Now we describe operations for homomorphic multiplication and addition of two floating point numbers $N = (-1)^s f 2^e$ and $N' = (-1)^{s'} f' 2^{e'}$. These operations are direct results of applying homomorphic addition/multiplication algorithm of $\mathsf{FHE4FX}$ against the above procedure for multiplying and adding two floating point numbers.

In the following description, conversion between, say $[|b|]_{\mathsf{e}}$ and $[|b|]_{\mathsf{f}}$, is implicit. The evaluation key evk is also implicit. The symbol "1.0" means an encoding integer 2^m for $\mathsf{FHE4FX}(q, l, m)$ scheme. Similarly, "2.0" means an encoding integer 2^{m+1} and "0.5" means an encoding integer 2^{m-1}.

- Mult $(c = ([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}}),\ c' = ([|s'|]_{\mathsf{s}}, [|f'|]_{\mathsf{f}}, [|e'|]_{\mathsf{e}}))$:
  $[|s''|]_{\mathsf{s}} \leftarrow [|1.0 - s|]_{\mathsf{s}} \cdot [|s'|]_{\mathsf{s}} + [|s|]_{\mathsf{s}} \cdot [|1.0 - s'|]_{\mathsf{s}}$
  $[|f''|]_{\mathsf{f}} \leftarrow [|f|]_{\mathsf{f}} \cdot [|f'|]_{\mathsf{f}},\ [|e''|]_{\mathsf{e}} \leftarrow [|e|]_{\mathsf{e}} + [|e'|]_{\mathsf{e}}$
  $[|b|]_{\mathsf{f}} \leftarrow \mathsf{MSb}([|f''|]_{\mathsf{f}})$ $\quad \{$the m+2-th bit of $f''\}$
  $[|f''|]_{\mathsf{f}} \leftarrow [|b|]_{\mathsf{f}} \cdot [|0.5|]_{\mathsf{f}} \cdot [|f''|]_{\mathsf{f}} + [|1.0 - b|]_{\mathsf{f}} \cdot [|f''|]_{\mathsf{f}}$
  $[|e''|]_{\mathsf{e}} \leftarrow [|b|]_{\mathsf{e}} \cdot [|e'' + 1.0|]_{\mathsf{e}} + [|1.0 - b|]_{\mathsf{e}} \cdot [|e''|]_{\mathsf{e}}$
  return $c'' = ([|s''|]_{\mathsf{s}}, [|f''|]_{\mathsf{f}}, [|e''|]_{\mathsf{e}})$.

- Add $(c = ([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}}),\ c' = ([|s'|]_{\mathsf{s}}, [|f'|]_{\mathsf{f}}, [|e'|]_{\mathsf{e}}))$:
  $[|s''|]_{\mathsf{s}} \leftarrow [|s|]_{\mathsf{s}} \cdot [|s'|]_{\mathsf{s}} + [|1.0 - s|]_{\mathsf{s}} \cdot [|1.0 - s'|]_{\mathsf{s}}$
  return $\mathsf{IfThenElse}([|s''|], \mathsf{Add}_0(c, c'), \mathsf{Add}_1(c, c'))$

- Add_0 $(c = ([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}}),\ c' = ([|s'|]_{\mathsf{s}}, [|f'|]_{\mathsf{f}}, [|e'|]_{\mathsf{e}}))$: $\quad \{s = s'\}$
  $[|b|]_{\mathsf{e}} \leftarrow \mathsf{GTb}([|e|]_{\mathsf{e}}, [|e'|]_{\mathsf{e}})$
  $([|s''|]_{\mathsf{s}}, [|f''|]_{\mathsf{f}}, [|e''|]_{\mathsf{e}}) \leftarrow \mathsf{IfTenElse}([|b|], \mathsf{Add}_{00}(c, c'), \mathsf{Add}_{01}(c, c'))$
  $[|d|]_{\mathsf{f}} \leftarrow \mathsf{MSb}([|f''|]_{\mathsf{f}})$ $\quad \{$the m+2-th bit of $f''\}$
  $[|f''|]_{\mathsf{f}} \leftarrow [|d|]_{\mathsf{f}} \cdot [|0.5|]_{\mathsf{f}} \cdot [|f''|]_{\mathsf{f}} + [|1.0 - d|]_{\mathsf{f}} \cdot [|f''|]_{\mathsf{f}}$
  $[|e''|]_{\mathsf{e}} \leftarrow [|d|]_{\mathsf{e}} \cdot [|e'' + 1.0|]_{\mathsf{e}} + [|1.0 - d|]_{\mathsf{e}} \cdot [|e''|]_{\mathsf{e}}$
  return $c'' = ([|s''|]_{\mathsf{s}}, [|f''|]_{\mathsf{f}}, [|e''|]_{\mathsf{e}})$.

- Add_{00} $(c = ([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}}),\ c' = ([|s'|]_{\mathsf{s}}, [|f'|]_{\mathsf{f}}, [|e'|]_{\mathsf{e}}))$: $\quad \{s = s', e > e'\}$
  $[|s''|]_{\mathsf{s}} \leftarrow [|s|]_{\mathsf{s}},\ [|e''|]_{\mathsf{e}} \leftarrow [|e|]_{\mathsf{e}}$
  $[|f''|]_{\mathsf{f}} \leftarrow [|f|]_{\mathsf{f}} + \mathsf{RightShift}([|f'|]_{\mathsf{f}}, [|e - e'|]_{\mathsf{e}})$
  return $c'' = ([|s''|]_{\mathsf{s}}, [|f''|]_{\mathsf{f}}, [|e''|]_{\mathsf{e}})$.

- Add_{01} $(c = ([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}}),\ c' = ([|s'|]_{\mathsf{s}}, [|f'|]_{\mathsf{f}}, [|e'|]_{\mathsf{e}}))$: $\quad \{s = s', e \leq e'\}$
  $[|s''|]_{\mathsf{s}} \leftarrow [|s|]_{\mathsf{s}},\ [|e''|]_{\mathsf{e}} \leftarrow [|e'|]_{\mathsf{e}}$
  $[|f''|]_{\mathsf{f}} \leftarrow [|f'|]_{\mathsf{f}} + \mathsf{RightShift}([|f|]_{\mathsf{f}}, [|e' - e|]_{\mathsf{e}})$
  return $c'' = ([|s''|]_{\mathsf{s}}, [|f''|]_{\mathsf{f}}, [|e''|]_{\mathsf{e}})$.

- Add_1 $(c = ([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}}),\ c' = ([|s'|]_{\mathsf{s}}, [|f'|]_{\mathsf{f}}, [|e'|]_{\mathsf{e}}))$: $\quad \{s \neq s'\}$
  $[|b_e|]_{\mathsf{e}} \leftarrow \mathsf{GTb}([|e|]_{\mathsf{e}}, [|e'|]_{\mathsf{e}})$
  $c'' \leftarrow \mathsf{IfThenElse}([|b_e|], \mathsf{Add}_{11}(c, c'), \mathsf{Add}_{1a}(c, c'))$
  return $\mathsf{Normalize}(c'')$.

- Add_{11} $(c = ([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}}),\ c' = ([|s'|]_{\mathsf{s}}, [|f'|]_{\mathsf{f}}, [|e'|]_{\mathsf{e}}))$: $\quad \{s \neq s', e > e'\}$
  $[|s''|]_{\mathsf{s}} \leftarrow [|s|]_{\mathsf{s}},\ [|e''|]_{\mathsf{e}} \leftarrow [|e|]_{\mathsf{e}}$
  $[|f''|]_{\mathsf{f}} \leftarrow [|f|]_{\mathsf{f}} - \mathsf{RightShift}([|f'|]_{\mathsf{f}}, [|e - e'|]_{\mathsf{e}})$
  return $c'' = ([|s''|]_{\mathsf{s}}, [|f''|]_{\mathsf{f}}, [|e''|]_{\mathsf{e}})$.

- Add_{1a} $(c = ([|s|]_{\mathsf{s}}, [|f|]_{\mathsf{f}}, [|e|]_{\mathsf{e}}),\ c' = ([|s'|]_{\mathsf{s}}, [|f'|]_{\mathsf{f}}, [|e'|]_{\mathsf{e}}))$: $\quad \{s \neq s', e \leq e'\}$
  $[|b'_e|]_{\mathsf{e}} \leftarrow \mathsf{GTb}([|e'|]_{\mathsf{e}}, [|e|]_{\mathsf{e}})$

return $\mathsf{IfThenElse}([|b'_e|], \mathsf{Add}_{12}(c, c'), \mathsf{Add}_{10}(c, c'))$

– Add_{12} $(c = ([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e}))$, $c' = ([|s'|]_\mathsf{s}, [|f'|]_\mathsf{f}, [|e'|]_\mathsf{e}))$: $\quad \{s \neq s', e < e'\}$
$[|s''|]_\mathsf{s} \leftarrow [|s'|]_\mathsf{s}$, $[|e''|]_\mathsf{e} \leftarrow [|e'|]_\mathsf{e}$
$[|f''|]_\mathsf{f} \leftarrow [|f'|]_\mathsf{f} - \mathsf{RightShift}([|f|]_\mathsf{f}, [|e' - e|]_\mathsf{e})$
return $c'' = ([|s''|]_\mathsf{s}, [|f''|]_\mathsf{f}, [|e''|]_\mathsf{e})$.

– Add_{10} $(c = ([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e}))$, $c' = ([|s'|]_\mathsf{s}, [|f'|]_\mathsf{f}, [|e'|]_\mathsf{e}))$: $\quad \{s \neq s', e = e'\}$
$[|b_f|]_\mathsf{f} \leftarrow \mathsf{GTb}([|f|]_\mathsf{f}, [|f'|]_\mathsf{f})$
return $\mathsf{IfThenElse}([|b_f|], \mathsf{Add}_{101}(c, c'), \mathsf{Add}_{10a}(c, c'))$

– Add_{101} $(c = ([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e}))$, $c' = ([|s'|]_\mathsf{s}, [|f'|]_\mathsf{f}, [|e'|]_\mathsf{e}))$: $\quad \{s \neq s', e = e', f > f'\}$
$[|s''|]_\mathsf{s} \leftarrow [|s|]_\mathsf{s}$, $[|e''|]_\mathsf{e} \leftarrow [|e|]_\mathsf{e}$, $[|f''|]_\mathsf{f} \leftarrow [|f|]_\mathsf{f} - [|f'|]_\mathsf{f}$
return $c'' = ([|s''|]_\mathsf{s}, [|f''|]_\mathsf{f}, [|e''|]_\mathsf{e})$.

– Add_{10a} $(c = ([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e}))$, $c' = ([|s'|]_\mathsf{s}, [|f'|]_\mathsf{f}, [|e'|]_\mathsf{e}))$: $\quad \{s \neq s', e = e', f \leq f'\}$
$[|b'_f|]_\mathsf{f} \leftarrow \mathsf{GTb}([|f'|]_\mathsf{f}, [|f|]_\mathsf{f})$
return $\mathsf{IfThenElse}([|b'_f|], \mathsf{Add}_{102}(c, c'), \mathsf{Add}_{100}(c, c'))$

– Add_{102} $(c = ([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e}))$, $c' = ([|s'|]_\mathsf{s}, [|f'|]_\mathsf{f}, [|e'|]_\mathsf{e}))$: $\quad \{s \neq s', e = e', f < f'\}$
$[|s''|]_\mathsf{s} \leftarrow [|s'|]_\mathsf{s}$, $[|e''|]_\mathsf{e} \leftarrow [|e'|]_\mathsf{e}$, $[|f''|]_\mathsf{f} \leftarrow [|f'|]_\mathsf{f} - [|f|]_\mathsf{f}$
return $c'' = ([|s''|]_\mathsf{s}, [|f''|]_\mathsf{f}, [|e''|]_\mathsf{e})$.

– Add_{100} $(c,\ c')$: return $c'' = ([|0|]_\mathsf{s}, [|1.0|]_\mathsf{f}, [|0.0|]_\mathsf{e})$. $\quad \{s \neq s', e = e', f' = f\}$

– $\mathsf{Normalize}$ $([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e})$:
Repeat m times:
$[|b|]_\mathsf{f} \leftarrow \mathsf{MSb}([|f|]_\mathsf{f}, 1)$ $\quad \{$the m+1-th bit of $f''\}$
$[|f|]_\mathsf{f} \leftarrow [|b|]_\mathsf{f} \cdot [|f|]_\mathsf{f} + [|1.0 - b|]_\mathsf{f} \cdot [|2.0|]_\mathsf{f} \cdot [|f|]_\mathsf{f}$
$[|e|]_\mathsf{e} \leftarrow [|b|]_\mathsf{e} \cdot [|e|]_\mathsf{e} + [|1.0 - b|]_\mathsf{e} \cdot [|e - 1.0|]_\mathsf{e}$
return $([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e})$.

– $\mathsf{RightShift}([|a|]_\mathsf{f}, [|e|]_\mathsf{e})$: $\quad \{e > 0\}$
$r \leftarrow [|1.0|]_\mathsf{f}$
For i in $[1..(l-1)]$: $\quad \{l$ is the bit-length of $e\}$
$r \leftarrow r \cdot r$
$[|b|]_\mathsf{e} \leftarrow \mathsf{MSb}([|e|]_\mathsf{e}, i)$
$r \leftarrow [|b|]_\mathsf{f} \cdot [|0.5|]_\mathsf{f} \cdot r + [|1.0 - b|]_\mathsf{f} \cdot r$
return $r \cdot [|a|]_\mathsf{f}$.

– $\mathsf{IfThenElse}([|b|], ([|s|]_\mathsf{s}, [|e|]_\mathsf{e}, [|f|]_\mathsf{f}), ([|s'|]_\mathsf{s}, [|e'|]_\mathsf{e}, [|f'|]_\mathsf{f}))$:
$[|s''|]_\mathsf{s} \leftarrow [|b|]_\mathsf{s} \cdot [|s|]_\mathsf{s} + [|1.0 - b|]_\mathsf{s} \cdot [|s'|]_\mathsf{s}$
$[|e''|]_\mathsf{e} \leftarrow [|b|]_\mathsf{e} \cdot [|e|]_\mathsf{e} + [|1.0 - b|]_\mathsf{e} \cdot [|e'|]_\mathsf{e}$
$[|f''|]_\mathsf{f} \leftarrow [|b|]_\mathsf{f} \cdot [|f|]_\mathsf{f} + [|1.0 - b|]_\mathsf{f} \cdot [|f'|]_\mathsf{f}$
return $([|s''|]_\mathsf{s}, [|e''|]_\mathsf{e}, [|f''|]_\mathsf{f})$.

A ciphertext of the FHE4FL scheme of parameter (q, m, l) is just a triple of ciphertexts of three FHE4FX schemes that share a same ciphertext modulus q and a same key pair $(\mathsf{pk}, \mathsf{sk})$. Recall among those three FHE4FX schemes, ciphertexts in one scheme can be publicly converted into another scheme ciphertext, keeping its underlying plaintext unchanged. So, we can view the triple of ciphertexts just a set of independent three ciphertexts under a single FXE4FX scheme. Especially, the FHE4FL scheme is also semantically secure and fully homomorphic (if parameters are suitably selected just as in the FHE4FX(q, m, l) scheme) because the FHE4FX scheme is so.

By Theorem 1, we have:

Theorem 2. *The* FHE4FL *scheme of parameter q, l, m can fully homomorphically compute additions and multiplications of encrypted floating point numbers $N = (-1)^s f 2^e$ with suitable choice of parameters that makes the underlying* FHE4FX *scheme (of parameter q, l, m) fully homomorphic.*

Efficiency. We estimate efficiency of homomorphic operations of FHE4FL scheme of parameter q, l, m in terms of multiplicative depth which means depth of nested multiplications of the underlying FHE4FX scheme required for target operation.

It is obvious that multiplicative depth of multiplication $\mathsf{Mult}(c, c')$ is constant. (The complexity of MSb is subsumed by the one of one multiplication in FHE4FX scheme.)

Multiplicative depth of addition $\mathsf{Add}(c, c')$ is dominated by depth of Normalize and RightShift, which are $O(ml)$ and $O(l^2)$, respectively. Hence multiplicative depth of addition $\mathsf{Add}(c, c')$ is $O(ml + l^2)$.

Acknowledgements

This work was supported by CREST, JST.

References

1. Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig, Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. M. Stam (Ed.): IMACC 2013, LNCS 8308, pp. 4564, 2013.
2. Zvika Brakerski, Fully homomorphic encryption without modulus switching from classical GapSVP. In Advances in Cryptology - Crypto 2012, LNCS 7417, pages 868886. Springer, 2012.
3. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, ITCS, pages 309325. ACM, 2012.
4. Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig, Improved security for a ring-based fully homomorphic encryption scheme. Martijn Stam, editor. Cryptography and Coding - 14th IMA International Conference, IMACC 2013, LNCS 8308, Springer, 2013.
5. Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun, Batch fully homomorphic encryption over the integers. In Thomas Johansson and Phong Q. Nguyen, editors, EUROCRYPT 2013, LNCS 7881, pages 315335. Springer, 2013.
6. Jung Hee Cheon, Miran Kim, Kristin Lauter, Homomorphic Computation of Edit Distance. Financial Cryptography and Data Security 2015, LNCS 8976, pp 194-212, 2015.
7. Junfeng Fan and Frederik Vercauteren, Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, 2012-144, 2012.
8. Craig Gentry, Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, STOC, pages 169178. ACM, 2009.
9. C. Gentry, S. Halevi, and N. P. Smart, Better bootstrapping in fully homomorphic encryption. In Public Key Cryptography, pages 1-16. 2012.
10. Philippe Golle, A private stable matching algorithm. In Financial Cryptography and Data Security, pages 6580. Springer, 2006.

11. Thore Graepel, Kristin Lauter, Michael Naehrig, ML Confidential: Machine Learning on Encrypted Data. International Conference on Information Security and Cryptology ICISC 2012, LNCS 7839, Springer-Verlag (2013), pp 1-21.
12. Yu Ishimaki, Kana Shimizu, Koji Nuida, Hayato Yamana, Faster privacy-preserving search for genome sequences using fully homomorphic encryption, SCIS'16, 2016, Kumamoto, Japan.
13. Wen-jie Lu, Shohei Kawasaki, Jun Sakuma, Cryptographically-secure Outsourcing of statistical Data Analysis I: Descriptive Statistics. CSS'15, 2015, Nagasaki, Japan.
14. Kristin Lauter, Adriana Lpez-Alt, Michael Naehrig, Private Computation on Encrypted Genomic Data. Progress in Cryptology LATINCRYPT 2014, LNCS 8895, Springer-Verlag (2014), pp 327.
15. Junqiang Liu, Jiuyong Li, Shijian Xu, and Benjamin C.M. Fung, Secure Outsourced Frequent Pattern Mining by Fully Homomorphic Encryption. S. Madria and T. Hara (Eds.): DaWaK 2015, LNCS 9263, pp. 7081, 2015.
16. Tancrède Lepoint, Michael Naehrig, A Comparison of the Homomorphic Encryption Schemes FV and YASHE. Progress in Cryptology AFRICACRYPT 2014, LNCS 8469, Springer-Verlag (2014), pp 318335.
17. Kristin Lauter, Michael Naehrig, Vinod Vaikuntanathan, Can Homomorphic Encryption be Practical? CCSW ' 11, 2011, Chicago, Illinois, USA.
18. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. EUROCRYPT 2010, LNCS 6110, pp. 1-23, 2010.
19. R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, pages 169-177. Academic Press, 1978.
20. Jacob Alperin-Sheriff and Chris Peikert, Practical Bootstrapping in Quasilinear Time. R. Canetti and J.A. Garay (Eds.): CRYPTO 2013, Part I, LNCS 8042, pp. 120, 2013.