Fully Homomorphic Encryption for Point Numbers

Seiko Arita and Shota Nakasato

Graduate School of Information Security,
Institute of Information Security, Japan
arita@iisec.ac.jp, nakasatoshota@gmail.com

Abstract. Based on the FV scheme, we construct at first fully homomorphic encryption scheme FX that can homomorphically compute addition and multiplication of encrypted fixed point numbers without knowing the secret key. Then, we show that in the FX 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 indicates $x > x'$ or not, given two ciphertexts c and c' of x and x', respectively, without knowing the secret key. Finally we show that these properties of the FX scheme enables us to construct a fully homomorphic encryption scheme FL that can homomorphically compute addition and multiplication of encrypted floating point numbers.
Keywords: Fully homomorphic encryption, FV scheme, Fixed/Floating point number, Greater-than bit.

1 Introduction

Fully Homomorphic Encryption (FHE) scheme enables us to homomorphically compute an encrypted XORed bit $\mathsf{Enc}(b_1 \,\mathsf{XOR}\, b_2)$ and 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 [9]. 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, 4], one can encrypt congruent integers (i.e., $x \bmod n$) and can homomorphically compute addition $\mathsf{Enc}(x_1 + x_2 \bmod n)$ and 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. Based on those schemes, various mining algorithms are experimentally and homomorphically evaluated against outsourced genomic, medical, or financial encrypted data [12, 6, 16, 14, 13].

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

FHE scheme FX for fixed point numbers. Our starting point is the FV scheme given by Fan and Vercauteren [8], 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 less than $n = \phi(m)$. ($\phi(\cdot)$ denotes the Euler function.) Two cyclotomic integer a and b are added through 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 $\Phi_m(X)$: $(a \cdot b)(X) = a(X)b(X) \bmod \Phi_m(X)$.

Cyclotomic integers are reduced modulo an integer q, resulting elements of $R_q = \mathbb{Z}_q[X]/(\Phi_m(X))$. A FV ciphertext is a pair $c = (c_0, c_1)$ ($\in R_q \times R_q$) of cyclotomic integers modulo ciphertext modulus q. A plaintext is a cyclotomic integer x ($\in R_t$) modulo plaintext modulus t. For simplicity we assume t divides q in this paper. A FV ciphertext $c = (c_0, c_1)$ of plaintext $x \in R_t$ satisfies the relation $c_0 + c_1 s = \frac{q}{t}x + v + q\alpha$ with some small noise v ($\in R_q$) and some cyclotomic integer $\alpha \in R$. 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 provided that coefficients of noise v are not too large relative to 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 $x = 2^m\tilde{x}$ is an integer in $\mathbb{Z}_{2^{m+l}}$. We suppose that integer x represents a constant polynomial x (i.e., a polynomial whose unique non-zero term is its constant term) in the cyclotomic ring R. To enable homomorphic computation of $\tilde{x}$, we will use the relation: $c_0 + c_1 s =$

$\frac{q}{t}2^{-m}x + v + q\alpha$ with $t = 2^{m+l}$. 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 $\tilde{x}' = 2^{-m}x'$. Product of fixed point numbers $\tilde{x}$ and $\tilde{x}'$ is defined as $\tilde{x}\tilde{x}' = 2^{-m}\big[\lfloor 2^{-m}xx' \rfloor\big]_t$. (Here, $\big[a\big]_t$ denotes the residue of a modulo t.) We want homomorphic version of this computation. By short calculation, we see that

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

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

$$\begin{aligned}\frac{t2^m}{q}&(c_0 + c_1 s)(c'_0 + c'_1 s)\\ &= \frac{q}{t}\Big\{2^{-m}\big[xx'\big]_{2^m} + \big[\lfloor 2^{-m}xx' \rfloor\big]_t + \big(\lfloor 2^{-m}xx' \rfloor - \big[\lfloor 2^{-m}xx' \rfloor\big]_t\big) + t(x'\alpha + x\alpha')\Big\}\\ &\quad + v'' + 2^m q\alpha''.\end{aligned}$$

Thus, 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

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

Here we see that w contains the wanted answer $2^m\tilde{x}\tilde{x}' = \big[\lfloor 2^{-m}xx' \rfloor\big]_t$ in the middle, but it also contains two annoying terms $LG = 2^{-m}\big[xx'\big]_{2^m}$ and $UG = \lfloor 2^{-m}xx' \rfloor - \big[\lfloor 2^{-m}xx' \rfloor\big]_t + t(x'\alpha + x\alpha')$. 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

$$\frac{t2^m}{q}(d_0 + d_1 s)(d'_0 + d'_1 s) = \frac{q}{t}\big[\lfloor 2^{-m}xx' \rfloor\big]_t + v'' + 2^m q\alpha''.$$

By dividing both sides by 2^m, we get

$$\frac{t}{q}(d_0 + d_1 s)(d'_0 + d'_1 s) = \frac{q}{t}2^{-m}\big[\lfloor 2^{-m}xx' \rfloor\big]_t + 2^{-m}v'' + q\alpha''$$

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 [10, 19]. Let $t = 2^{m+l}$ and let $w = x + 2^m z$ 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 $2^m z$. The key observation is the following simple fact [10, 19]: if w is equal to $b \in \{0, 1\} \bmod 2^i$ then w^2 is equal to the same $b \in \{0, 1\} \bmod 2^{i+1}$ 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 wanted $2^m z = w - \sum_{i=0}^{m-1} w_i$. UpperClear clears the upper garbage by a similar method.

Summarizing, we use the FV scheme with plaintext modulus $t' = 2^m t$ (with $t = 2^{m+l}$) to enable homomorphic evaluation on encrypted 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 the multiplied FV ciphertexts. We call our FHE scheme for fixed point numbers built in this way FX scheme. Since the FV scheme is semantically secure and fully homomorphic, our FX (which ciphertext is nothing but a FV ciphertext with enlarged plaintext modulus) is also semantically secure and fully homomorphic.

Greater-than bit extraction. As an application of the FX 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 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 [14, 13] such problem is tackled by Greater-Than protocol based on (such as) the one given by Golle [11]. Let $D \subset \mathbb{Z}$ be a range that possible x_i's belong to. The 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, it requires $O(\left|D\right|)$ encryptions and $O(\left|D\right|)$ homomorphic additions among them and needs interaction with secret key holder. By using the FX 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 FL for floating point numbers. Using the method of greater-than bit extraction by FX, we will construct a fully homomorphic encryption scheme for floating point numbers, FL. 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) FX 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 difficult to evaluate such process homomorphically, given 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 FX scheme is semantically secure and fully homomorphic, the FL scheme for floating point numbers is also semantically secure and fully homomorphic.

Related works. There are some different approaches to handle fixed-point arithmetic in homomorphic encryption schemes in the literatures. Costache, Smart, Vivek, and Waller [7] encodes fixed-point numbers as polynomials in cyclotomic rings, which enables a lower plaintext modulus. Although in contrast our method encodes point numbers only into constant terms of polynomials, our method can adapt to homomorphic SIMD operations using plaintext slots [20]. Chung and Kim [5] encodes rational numbers using continued fractions. They restrict their interest to linear multivariate polynomials.

Organization. Section 2 recalls the FV scheme as well as its basic properties. We construct the FX scheme for fixed point numbers in Section 3. After treating the problem of homomorphic greater-than bit extraction in Section 4, we construct the FL 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. Keygen generates 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 encrypts a plaintext $x \in \{0,1\}$ to a ciphertext c under a public key pk: $c \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$. $\mathsf{Decrypt}$ decrypts a ciphertext c to a plaintext x by the secret key sk: $x \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$. Eval 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 is called *fully homomorphic encryption* (FHE) scheme if it is L-homomorphic for any polynomial function $L = poly(n)$.

2.2 The FV Scheme

The FV scheme [8, 17] 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 $\Phi_m(X)$ be the m-th cyclotomic polynomial. The ring $R = \mathbb{Z}[X]/\Phi_m(X)$ is called the m-th cyclotomic ring. For cyclotomic integer $a = \sum_{i=0}^{n-1} a_i X^i \in R$, let $\|a\|_\infty = \max_{0 \le i < n}\{|a_i|\}$ be the infinity norm of a. Let δ be the expansion factor of R, i.e., $\delta = \sup_{a,b \in R}\{\|ab\|_\infty / (\|a\|_\infty \|b\|_\infty)\}$. 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 $\lceil \cdot \rfloor_q$ reduction modulo q into the interval $(-q/2, q/2]$. We fix an integer base w and let $l_w = \lfloor \log_w(q) \rfloor + 1$. 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 $\mathsf{WD}(a) = \left(\lceil a_i \rfloor_w\right)_{i=0}^{l_w-1} \in R^{l_w}$ and $\mathsf{PO}(a) = \left(\lceil a w^i \rfloor_q\right)_{i=0}^{l_w-1} \in R^{l_w}$. As easily verified, $\langle \mathsf{WD}(a), \mathsf{PO}(b) \rangle \equiv ab \pmod q$, where $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^{l_w-1} x_i y_i$.

Let χ_{key} and χ_{err} be two discrete, bounded probability distributions on R. Constants B_{key} and B_{err} denote the corresponding bounds: $\chi_{key} < B_{key}$, $\chi_{err} < B_{err}$. 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, denoting ciphertext modulus and plaintext modulus, respectively. 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. Figure 1 shows algorithms in the FV scheme. 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 random pair (a, b) in $(R_q)^2$. By standard hybrid argument a ciphertext (c_0, c_1) is also indistinguishable from uniformly random pair over $(R_q)^2$.

Keygen () : $s \leftarrow \chi_{key}$, $e \leftarrow \chi_{err}$, $a \xleftarrow{\mathsf{u}} R_q$, $b = \left[-(as+e)\right]_q$ $\boldsymbol{a} \xleftarrow{\mathsf{u}} R_q^{lw}$, $\boldsymbol{e} \leftarrow \chi_{err}^{lw}$, $\boldsymbol{b} = \left[\mathsf{PO}(s^2) - (\boldsymbol{a}s + \boldsymbol{e})\right]_q$ return $\mathsf{sk} = s$, $\mathsf{pk} = (a, b)$, $\mathsf{evk} = (\boldsymbol{a}, \boldsymbol{b})$.
Enc $((a, b),\ x \in R_t)$: $u \leftarrow \chi_{key}$, $e_1, e_2 \leftarrow \chi_{err}$, return $c = (c_0 = \left[\Delta x + bu + e_1\right]_q, c_1 = \left[au + e_2\right]_q)$
Dec $(s,\ c = (c_0, c_1))$: return $\left[\left\lfloor \frac{t}{q}\left[c_0 + c_1 s\right]_q \right\rceil\right]_t$.
Add $(c = (c_0, c_1),\ c' = (c_0', c_1'))$: return $c_{add} = \left(\left[c_0 + c_0'\right]_q, \left[c_1 + c_1'\right]_q\right)$.
Mult $(c = (c_0, c_1),\ c' = (c_0', c_1'),\ \gamma = (\boldsymbol{a}, \boldsymbol{b}))$: $d_0 = \left[\left\lfloor \frac{t}{q} c_0 c_0' \right\rceil\right]_q$, $d_1 = \left[\left\lfloor \frac{t}{q}(c_0 c_1' + c_1 c_0') \right\rceil\right]_q$, $d_2 = \left[\left\lfloor \frac{t}{q} c_1 c_1' \right\rceil\right]_q$ return $c_{mult} = \left(\left[d_0 + \langle \mathsf{WD}(d_2), \boldsymbol{b}\rangle\right]_q, \left[d_1 + \langle \mathsf{WD}(d_2), \boldsymbol{a}\rangle\right]_q\right)$.

Fig. 1. The FV scheme

Definition 1. *The* inherent noise term v *of FV ciphertext* $c = (c_0, c_1)$ *designed for* $x \in R_t$ *is an element* $v \in R$ *of smallest norm* $\left\|v\right\|_\infty$ *satisfying* $c_0 + c_1 s = \Delta x + v + q\alpha$ *for some* $\alpha \in R$.

A fresh ciphertext (c_0, c_1) directly produced by $\mathsf{Enc}((a, b), x)$ satisfies $c_0 + c_1 s - \Delta x \equiv bu + e_1 + (au + e_2)s \equiv -eu + e_1 + e_2 \pmod q$. So, fresh ciphertexts have inherent noise terms $v = -eu + e_1 + e_2$ bounded as $\left\|v\right\|_\infty \le V := B_{err}(1 + 2\delta B_{key})$.

Let $V_{max}^{\mathsf{FV}} = \frac{1}{2}\Delta = \frac{1}{2}\left(\frac{q}{t}\right)$. Following lemmas adapted from [8, 17, 1] show some basic properties of the FV scheme.

Lemma 1 (Correctness of FV scheme). *Let v be the inherent noise term of FV ciphertext c designed for $x \in R_t$. If $\|v\|_\infty < V_{max}^{\mathsf{FV}}$, then decryption works correctly, i.e., $\mathsf{Dec}(s, c) = [x]_t = x$.*

Lemma 2 (Additive Noise of FV scheme). *Let v and v' be inherent noise terms of FV ciphertexts 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 $[x + x']_t \in R_t$. Then, $\|v_{add}\|_\infty \leq \|v\|_\infty + \|v'\|_\infty$.*

Lemma 3 (Multiplicative Noise of FV scheme). *Let v and v' be inherent noise terms of FV ciphertexts c and c', designed for x and $x' \in R_t$, respectively. Suppose $\|v\|_\infty, \|v'\|_\infty < V$ for some V $(< V_{max}^{\mathsf{FV}})$. Let v_{mult} be the inherent noise term of $c_{mult} \leftarrow \mathsf{Mult}(c, c')$ designed for $[xx']_t \in R_t$. Then, $\|v_{mult}\|_\infty \leq \delta t(2+4\delta B_{key})V + \delta^2 B_{key}(2B_{key} + 4t^2) + 2^{-1}\delta l_w w B_{err}$.*

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 FV 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, $\|v_f\|_\infty \leq C_1^L V + LC_1^{L-1}C_2$ 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 L_{dec} be the level of some circuit that evaluates decryption algorithm $\mathsf{Dec}(c, \cdot)$ with a FV ciphertext c built-in. By Lemma 1 and Corollary 1, if inequality $\Delta(= q/t) > 2(C_1^{L_{dec}}V + L_{dec}C_1^{L_{dec}-1}C_2)$ holds, we can homomorphically evaluate 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 algorithm $\mathsf{Dec}(c, \cdot)$ by some circuit of level $L_{dec} = O(\log n)$ which is independent of c. Hence the inequality can be satisfied by taking sufficiently large $q = O(n^{\log n})$ for any ciphertext c with $\delta = poly(n)$. 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 FX

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: $\tilde{x} = 2^{-m}x$. Let $t = 2^{m+l}$. Addition and multiplication of two fixed point numbers $\tilde{x} = 2^{-m}x$ and $\tilde{y} = 2^{-m}y$ are defined as

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

We see that 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, just a homomorphic addition in underlying FV scheme. However, product $\tilde{x} \cdot \tilde{y}$ is more complicated. It is encoded by integer $\big[\big\lfloor 2^{-m}\big[x \cdot y\big]_{2^m t}\big\rfloor\big]_t$, that results from m-bit right shift of integer product $\big[x \cdot y\big]_{2^m t}$. Now our problem is distinguished: How can we homomorphically compute m-bit right shift of given encrypted $(2m + l)$-bit integers?

Let $\mathsf{FV}(2^m t)$ 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, FX, that can homomorphically compute the m-bit right shift of encrypted encoding integers.

Parameters. The FX 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 FX scheme consists of the algorithms in Figure 2. Note that FX ciphertexts of parameter (q, m, l) are nothing but the $\mathsf{FV}(2^m t)$ ciphertexts, 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 $\mathsf{FV}(2^m t)$ scheme, since $\mathsf{FX}(q, m, l)$ only recovers $(l + m)$-bit integers rather than $(l + 2m)$-bit integers in $\mathsf{FV}(2^m t)$.) Especially, the FX scheme is also semantically secure and fully homomorphic with suitable choice of parameters that makes the underlying $\mathsf{FV}(2^m t)$ scheme so. The difference is in the way of homomorphic evaluation.

Let c be a FX ciphertext of parameter (q, m, l). By definition of Enc, 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 FX scheme has m bits more in plaintext space than finally required for decryption. FX scheme uses this room of m bits in plaintext space in order to homomorphically compute the m-bit right-shift of encrypted integer.

$\mathsf{Keygen}\ ()$: 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.

Fig. 2. The FX scheme

To homomorphically compute such m-bit right-shift, algorithms LowerClear and UpperClear are useful. Let c be an $\mathsf{FV}(2^m t)$ ciphertext 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 Lemma 5, the least m-bit x (lower garbage) of w will be cleared by LowerClearPlain algorithm in Figure 3.

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} (See the proof of Lemma 5)$\}$
    $g = g + w_i$
  return $w - g$. $\{= 2^m z\}$

Fig. 3. The LowerClearPlain algorithm

As directly verified, LowerClear (c) procedure (for c encrypting $w = x + 2^m z$) in Figure 2 is a homomorphic version of LowerClearPlain (w) procedure, that computes $2^m z$ given $w = x + 2^m z$. Here we add some 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 $\mathsf{FV}(2^m t)$ $(=\mathsf{FX}(q, m, l))$ scheme will satisfy

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

with some small noise $v \in R$ and $\alpha \in R$. 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 homomorphic version of the step:

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

in Figure 3 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 the LowerClear algorithm in Figure 2. The resulting ciphertext d_i will be treated as a $\mathsf{FV}(2^m t)$ ciphertext.

Now 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 get a ciphertext e of $u = z + ty$ with some integer y. (Recall that our plaintext space is $2m + l$ bits. So the term $ty = 2^{2m+l}/2^m \cdot y$ should appear with some y due to the division by 2^m.) Note that y is the significant m bits of $z + ty$ (upper garbage). As in the case of LowerClear, we use UpperClear

UpperClearPlain $(u = z + ty)$:

$r \leftarrow 0$

For $i \in [1..(m+l)]$: $u_i \leftarrow u - r$, $u_i = \left(\left(\frac{1}{2}\right)^{i-1} u_i\right)^{2^{2m+l-i}} \cdot 2^{i-1}$, $r = r + u_i$.

return r $\quad \{= z\}$

Fig. 4. The UpperClearPlain algorithm

algorithm in Figure 2 to clear the upper garbage y from $u = z + ty$, that is a homomorphic version of UpperClearPlain algorithm in Figure 4.

Now we turn to formal treatment.

Definition 2. *The* inherent noise term v *of FX ciphertext* $c = (c_0, c_1)$ *designed for* $x \in R_{2^m t}$ *is the term* v *of smallest norm* $\left\|v\right\|_\infty$ *satisfying* $c_0 + c_1 s = \Delta 2^{-m} x + v + q\alpha$ *for some* $\alpha \in R$.

Let $V_{max} = \frac{1}{2}\Delta 2^{-m} = \frac{1}{2}\left(\frac{q}{2^m t}\right)$. Immediately from Lemma 1,

Lemma 4. *Let* v *be the inherent noise term of FX ciphertext* c *designed for* $x \in R_t$. *If* $\left\|v\right\|_\infty < V_{max}$, *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 FX 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* v *be the inherent noise term of the ciphertext* $c \leftarrow \mathsf{LowerClear}(\tilde{c})$ *designed for constant polynomial* $2^m z \in 2^m R_{2^m t}$. *Then,* $\left\|v\right\|_\infty \le \left\|\tilde{v}\right\|_\infty + V_{\mathsf{LC}}$, *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 $w = x + 2^m z$. The key observation is the following simple fact [10, 19]: if w is equal to $b \in \{0, 1\}$ mod 2^i then w^2 is equal to the same 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 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 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 $2^m z = w - x$. Since $\mathsf{LowerClearPlain}$ has $2m + l - 1$ levels of nested multiplications, by Corollary 1 we get the claimed noise bound V_{LC} on the noise occurred by its homomorphic evaluation $\mathsf{LowerClear}$. □

Similarly, we have:

Lemma 6 (Upper Clear). *Let $\tilde{v}$ be the inherent noise term of FX 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 v be the inherent noise term of the ciphertext $c \leftarrow \mathsf{UpperClear}(\tilde{c})$ designed for constant polynomial $z \in R_t$. Then, $\|v\|_\infty \le \|\tilde{v}\|_\infty + V_{\mathsf{UC}}$, where $V_{\mathsf{UC}} = C_1^{2m+l-1} V + L C_1^{2m+l-2} C_2$.*

Proposition 1 (Additive Noise of FX scheme). *Let v and v' be inherent noise terms of FX ciphertexts 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 $[x + x']_t \in R_t$. Then, $\|v_{add}\|_\infty \le \|v\|_\infty + \|v'\|_\infty$.*

Proof. This is a restatement of Lemma 2.□

Proposition 2 (Multiplicative Noise of FX scheme). *Let v and v' be inherent noise terms of FX ciphertexts 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 $c_{mult} \leftarrow \mathsf{Mult}(c, c', \mathsf{evk})$ designed for $\big[\lfloor 2^{-m} [xx']_{2^m t} \rfloor\big]_t \in R_t$. Then, $\|v_{mult}\|_\infty \le \delta t(2+4\delta B_{key})V + \delta^2 B_{key}(B_{key} + 4 \cdot 2^m t^2) + l_w \delta w B_{err} + 2^{-m} V_{\mathsf{LC}} + V_{\mathsf{UC}}$.*

Proof. We use notation in Figure 2. For $\tilde{c} = \mathsf{FV}(2^m t).\mathsf{Mult}(c, c', \gamma)$ by Lemma 3 we have $\tilde{c}_0 + \tilde{c}_1 s = \Delta 2^{-m} [xx']_{2^m t} + v + q\alpha$ with $\|v\|_\infty \le \delta 2^m t(2 + 4\delta B_{key})V + \delta^2 B_{key}(2B_{key} + 4 \cdot 2^{2m} t^2) + 2^{-1}\delta l_w w B_{err}$. Then, by Lemma 5 for $d \leftarrow \mathsf{LowerClear}(\tilde{c})$ we have

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

with $\|w\|_\infty \le V_{\mathsf{LC}}$. Dividing by 2^m,

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

Hence, rounded $e = \Big(\big[\big\lfloor \frac{1}{2^m} d_0 \big\rceil \big]_q, \big[\big\lfloor \frac{1}{2^m} d_1 \big\rceil \big]_q \Big)$ must satisfy

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

with $\big\|w'\big\|_\infty \leq \frac{1}{2}(1 + \delta B_{key})$. Lemma 6 shows for $c_{mult} = \mathsf{UpperClear}(e)$, $c_{mult,0} + c_{mult,1} s = \Delta 2^{-m} \cdot \big[\big\lfloor \frac{\big[xx'\big]_{2^m t}}{2^m} \big\rfloor \big]_t + \frac{v}{2^m} + \frac{w}{2^m} + w' + w'' + q\beta$ with $\big\|w''\big\|_\infty \leq V_{\mathsf{UC}}$.

Accumulated noise $z = \frac{v}{2^m} + \frac{w}{2^m} + w' + w''$ satisfies

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

By Proposition 1, 2 we have

Theorem 1. *The* FX *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* $\tilde{x} \cdot \tilde{y} = 2^{-m} \big(\big[\big\lfloor 2^{-m} \big[x \cdot y\big]_{2^{2m+l}} \big\rfloor \big]_{2^{m+l}} \big)$.

Efficiency. We estimate efficiency of homomorphic operations of FX scheme of parameter q, l, m. Addition $\mathsf{Add}(c, c')$ is done by one addition of $\mathsf{FV}(q, 2^m t)$ scheme. We estimate complexity of multiplication $\mathsf{Mult}(c, c', \gamma)$ in terms of "multiplicative depth" and "multiplicative number". The multiplicative depth means the required depth of nested multiplications of the underlying FV scheme to perform the target operation. It determines noise growth due to the target operation and larger noise 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, complexity of $\mathsf{Mult}(c, c', \gamma)$ operation is dominated by $\mathsf{UpperClear}(e)$. The multiplicative depth of $\mathsf{UpperClear}$ is $2m + l - 1$ and the multiplicative number of $\mathsf{UpperClear}$ is $(2m + l - 1) + \cdots + (m + l) = \frac{1}{2}(3m + 2l - 1)(m + l)$. Thus, roughly

estimated, 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

As an application of our FX 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 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 [14, 13] such problem is tackled by Greater-Than protocol based on (such as) the one given by Golle [11]. 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 requires $O(\left|D\right|)$ encryptions and $O(\left|D\right|)$ homomorphic additions among them and needs an interaction with secret key holder. We show that, by using the FX scheme, we 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.

First, we describe procedure $\mathsf{MSb}(c)$, that computes an encryption of the most significant bit of $x \in R_t$, given a $\mathsf{FX}(q, l, m)$ ciphertext c of some fixed point number $\tilde{x} = 2^{-m}x$. The point is that one can multiply any fixed point numbers by 0.5 in FX 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 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 fixed point number $b.0$, as desired. Figure 5 gives a description of MSb.

Using $\mathsf{MSb}(\cdot)$, it is straightforward to compute an encryption of greater-than bit b indicating $x_1 > x_2$, given $c_1 = \mathsf{Enc}(x_1)$ and $c_2 = \mathsf{Enc}(x_2)$. Basically it simply computes $\mathsf{MSb}(c_2 - c_1)$. 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, $b = \mathsf{MSb}(x_2)$. Figure 5 gives also a description of GTb algorithm, in which $\mathsf{FX.Add}(c, c')$ (or $\mathsf{FX.Mult}(c, c')$) is simply written as $c + c'$ (or $c \cdot c'$, respectively).

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

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

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}
  $one \leftarrow \mathsf{FX.Enc}(2^m)$ {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}
  return $d \leftarrow same\_sign \cdot d_1 + (one - same\_sign) \cdot d_2$.

Fig. 5. The MSb and GTb algorithms

5 The Proposed Scheme FL

In this section, we construct a fully homomorphic encryption scheme for floating point numbers, FL, using the GTb algorithm by FX. 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', i.e. $s'' = (1-s)s' + (1-s')s$. Significands are multiplied and exponents are added, $f'' = ff'$ and $e'' = e + e'$. If $f'' > 2$, we need to 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.

Add_{00}: If $s = s'$ and $e > e'$, let $s'' = s$, $f'' = f + 2^{e'-e} f'$, and $e'' = e$. If $f'' > 2$, let $f'' = f''/2$ and $e'' = e'' + 1$.

Add_{01}: If $s = s'$ and $e \le e'$, let $s'' = s$, $f'' = 2^{e-e'} f + f'$, and $e'' = e'$. If $f'' > 2$, let $f'' = f''/2$ and $e'' = e'' + 1$.

Add_{11}: If $s \neq s'$ and $e > e'$, let $s'' = s$, $f'' = f - 2^{e'-e}f'$, and $e'' = e$. While $f'' < 1$, do $f'' = 2f''$, $e'' = e'' - 1$.
Add_{12}: If $s \neq s'$ and $e < e'$, let $s'' = s'$, $f'' = f' - 2^{e-e'}f$, and $e'' = e'$. While $f'' < 1$, do $f'' = 2f''$, $e'' = e'' - 1$.
Add_{101}: If $s \neq s'$, $e = e'$ and $f > f'$, let $s'' = s$, $f'' = f - f'$, $e'' = e$. While $f'' < 1$, do $f'' = 2f''$, $e'' = e'' - 1$.
Add_{102}: If $s \neq s'$, $e = e'$ and $f < f'$, let $s'' = s'$, $f'' = f' - f$, and $e'' = e'$. While $f'' < 1$, do $f'' = 2f''$, $e'' = e'' - 1$.
Add_{100}: If $s \neq s'$, $e = e'$ and $f = f'$, let $s'' = 0$, $f'' = 1.0$, $e'' = 0$.

5.1 The scheme FL

Parameters. Consider a floating point number $N = (-1)^s f 2^e$. The FL 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 FL 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 FX schemes FXs, FXf and FXe that share a same ciphertext modulus q and a same key pair $(\mathsf{pk}, \mathsf{sk})$.

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

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

Scheme Description. Figure 6 shows the first part, i.e. Keygen, Enc and Dec algorithms, of our FL scheme.

Conversion. Note that we can publicly and efficiently convert ciphertexts, keeping its underlying plaintext unchanged, between FX schemes that share a same ciphertext modulus q and a same key pair $(\mathsf{pk}, \mathsf{sk})$. In fact, suppose two schemes $\mathsf{FX}(q, l, m)$ and $\mathsf{FX}(q, l', m')$ 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 $\mathsf{FX}(q, l, m)$ 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

$\mathsf{Keygen}\ () : \mathsf{return}\ (\mathsf{sk} = s, \mathsf{pk} = (a, b), \mathsf{evk} = (\boldsymbol{a}, \boldsymbol{b})) \leftarrow \mathsf{FXs.Keygen}().$ { The key $(\mathsf{sk}, \mathsf{pk}, \mathsf{evk})$ will be shared among the three schemes FXs, FXf and FXe. }
$\mathsf{Enc}\ (\mathsf{pk},\ N = (s, f, e) \in R_2 \times R_{2^{2+m}} \times R_{2^{l+2}}) :$ $[\lvert s\rvert]_{\mathsf{s}} \leftarrow \mathsf{FXs.Enc}(\mathsf{pk}, s), \quad [\lvert f\rvert]_{\mathsf{f}} \leftarrow \mathsf{FXf.Enc}(\mathsf{pk}, f), \quad [\lvert e\rvert]_{\mathsf{e}} \leftarrow \mathsf{FXe.Enc}(\mathsf{pk}, e)$ $\mathsf{return}\ c = ([\lvert s\rvert]_{\mathsf{s}}, [\lvert f\rvert]_{\mathsf{f}}, [\lvert e\rvert]_{\mathsf{e}}).$
$\mathsf{Dec}\ (\mathsf{sk},\ c = ([\lvert s\rvert]_{\mathsf{s}}, [\lvert f\rvert]_{\mathsf{f}}, [\lvert e\rvert]_{\mathsf{e}})) :$ $s \leftarrow \mathsf{FXs.Dec}(\mathsf{sk}, [\lvert s\rvert]_{\mathsf{s}}), \quad f \leftarrow \mathsf{FXf.Dec}(\mathsf{sk}, [\lvert f\rvert]_{\mathsf{f}}), \quad e \leftarrow \mathsf{FXe.Dec}(\mathsf{sk}, [\lvert e\rvert]_{\mathsf{e}})$ $\mathsf{return}\ N = (s, f, e).$

Fig. 6. The first part of FL algorithms

$x \in R_{l'}$ as a ciphertext in $\mathsf{FX}(q, l', m')$ 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 FX scheme that can treat an encryption of "0.5".

Figure 7 shows the second part, i.e. Mult and Add algorithms, of FL scheme, in which conversions between, say $[\lvert b\rvert]_{\mathsf{e}}$ and $[\lvert b\rvert]_{\mathsf{f}}$, are implicit.

A ciphertext of the FL scheme of parameter (q, m, l) is just a triple of ciphertexts of three FX schemes that share a same ciphertext modulus q and a same key pair $(\mathsf{pk}, \mathsf{sk})$. Recall among those three FX 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 FL scheme is also semantically secure and fully homomorphic with suitable choice of parameters that makes the underlying $\mathsf{FX}(q, m, l)$ scheme so.

Theorem 2. *The* FL *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* FX *scheme (of parameter q, l, m) fully homomorphic.*

Efficiency. Multiplicative depth, that is the depth of nested multiplications of the underlying FX scheme, of $\mathsf{Mult}(c, c')$ is $O(m)$, dominated by the complexity of MSb in it. Multiplicative depth of $\mathsf{Add}(c, c')$ 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.

$\mathsf{Mult}\ (c = ([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e})),\ c' = ([|s'|]_\mathsf{s}, [|f'|]_\mathsf{f}, [|e'|]_\mathsf{e})) :$
$[|s''|]_\mathsf{s} \leftarrow [|1.0 - s|]_\mathsf{s} \cdot [|s'|]_\mathsf{s} + [|s|]_\mathsf{s} \cdot [|1.0 - s'|]_\mathsf{s},\ [|f''|]_\mathsf{f} \leftarrow [|f|]_\mathsf{f} \cdot [|f'|]_\mathsf{f},\ [|e''|]_\mathsf{e} \leftarrow [|e|]_\mathsf{e} + [|e'|]_\mathsf{e}$
$[|b|]_\mathsf{f} \leftarrow \mathsf{MSb}([|f''|]_\mathsf{f})$ {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 $c'' = ([|s''|]_\mathsf{s}, [|f''|]_\mathsf{f}, [|e''|]_\mathsf{e})$.

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

$\mathsf{Add}_0\ (c = ([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e})),\ c' = ([|s'|]_\mathsf{s}, [|f'|]_\mathsf{f}, [|e'|]_\mathsf{e})) :$ $\{s = s'\}$
$[|b|]_\mathsf{e} \leftarrow \mathsf{GTb}([|e|]_\mathsf{e}, [|e'|]_\mathsf{e}),\ ([|s''|]_\mathsf{s}, [|f''|]_\mathsf{f}, [|e''|]_\mathsf{e}) \leftarrow \mathsf{IfTenElse}([|b|], \mathsf{Add}_{00}(c, c'), \mathsf{Add}_{01}(c, c'))$
$[|d|]_\mathsf{f} \leftarrow \mathsf{MSb}([|f''|]_\mathsf{f})$ {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 $c'' = ([|s''|]_\mathsf{s}, [|f''|]_\mathsf{f}, [|e''|]_\mathsf{e})$.

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

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

$\mathsf{Add}_1\ (c = ([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e})),\ c' = ([|s'|]_\mathsf{s}, [|f'|]_\mathsf{f}, [|e'|]_\mathsf{e})) :$ $\{s \neq s'\}$
$[|b_e|]_\mathsf{e} \leftarrow \mathsf{GTb}([|e|]_\mathsf{e}, [|e'|]_\mathsf{e}),\ c'' \leftarrow \mathsf{IfThenElse}([|b_e|], \mathsf{Add}_{11}(c, c'), \mathsf{Add}_{1a}(c, c'))$
return $\mathsf{Normalize}(c'')$.

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

$\mathsf{Add}_{1a}\ (c = ([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e})),\ c' = ([|s'|]_\mathsf{s}, [|f'|]_\mathsf{f}, [|e'|]_\mathsf{e})) :$ $\{s \neq s', e \leq e'\}$
$[|b'_e|]_\mathsf{e} \leftarrow \mathsf{GTb}([|e'|]_\mathsf{e}, [|e|]_\mathsf{e})$, return $\mathsf{IfThenElse}([|b'_e|], \mathsf{Add}_{12}(c, c'), \mathsf{Add}_{10}(c, c'))$

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

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

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

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

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

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

$\mathsf{Normalize}\ ([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e}) :$
Repeat m times:
$[|b|]_\mathsf{f} \leftarrow \mathsf{MSb}([|f|]_\mathsf{f}, 1)$ {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 $([|s|]_\mathsf{s}, [|f|]_\mathsf{f}, [|e|]_\mathsf{e})$.

$\mathsf{RightShift}\ ([|a|]_\mathsf{f}, [|e|]_\mathsf{e}) :$ $\{e > 0\}$
$r \leftarrow [|1.0|]_\mathsf{f}$
For i in $[1..(l-1)]$: $\{l$ 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 $([|s''|]_\mathsf{s}, [|e''|]_\mathsf{e}, [|f''|]_\mathsf{f})$.

Fig. 7. The second part of FL algorithms.

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. 45-64, 2013.
2. Zvika Brakerski, Fully homomorphic encryption without modulus switching from classical GapSVP. Crypto 2012, LNCS 7417, pages 868-886. Springer, 2012.
3. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, ITCS, pages 309-325. ACM, 2012.
4. 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 315-335. Springer, 2013.
5. H.W. Chung, M. Kim, Encoding Rational Numbers for FHE-based Applications. IACR Cryptology ePrint Archive (2016/344).
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. A. Costache, N. P. Smart, S. Vivek, A. Waller, Fixed-Point Arithmetic in SHE Schemes. IACR Cryptology ePrint Archive (2016/250).
8. Junfeng Fan and Frederik Vercauteren, Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, (2012/144).
9. Craig Gentry, Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, STOC, pages 169-178. ACM, 2009.
10. C. Gentry, S. Halevi, and N. P. Smart, Better bootstrapping in fully homomorphic encryption. In Public Key Cryptography, pages 1-16. 2012.
11. Philippe Golle, A private stable matching algorithm. In Financial Cryptography and Data Security, pages 65-80. Springer, 2006.
12. Thore Graepel, Kristin Lauter, Michael Naehrig, ML Confidential: Machine Learning on Encrypted Data. ICISC 2012, LNCS 7839, Springer-Verlag (2013), pp 1-21.
13. Yu Ishimaki, Kana Shimizu, Koji Nuida, Hayato Yamana, Faster privacy-preserving search for genome sequences using fully homomorphic encryption, SCIS'16, 2016, Japan.
14. Wen-jie Lu, Shohei Kawasaki, Jun Sakuma, Cryptographically-secure Outsourcing of statistical Data Analysis I: Descriptive Statistics. CSS'15, 2015, Japan.
15. K. Lauter, A. Lpez-Alt, M. Naehrig, Private Computation on Encrypted Genomic Data. LATINCRYPT 2014, LNCS 8895, Springer-Verlag, pp 3-27.
16. 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. 70-81, 2015.
17. Tancrède Lepoint, Michael Naehrig, A Comparison of the Homomorphic Encryption Schemes FV and YASHE. AFRICACRYPT 2014, LNCS 8469, Springer-Verlag, pp 318-335, 2014.
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. J. Alperin-Sheriff and C. Peikert, Practical Bootstrapping in Quasilinear Time. CRYPTO 2013, Part I, LNCS 8042, pp. 1-20, 2013.
20. N.P. Smart and F. Vercauteren, Fully Homomorphic SIMD Operations. Designs, Codes and Cryptography, April 2014, Volume 71, Issue 1, pp 5781.