International Journal of Foundations of Computer Science

Encoding of Rational Numbers and Their Homomorphic Computations for FHE-based Applications

Heewon Chung
Department of Mathematical Science, Seoul National University, Gwanak-ro 1, Gwanak-gu, Seoul 08826, Republic of Korea
runghyun@snu.ac.kr

Myungsun Kim*
Department of Information Security, The University of Suwon Hwaseong, Gyeonggi-do 18523, Republic of Korea
msunkim@suwon.ac.kr

This work addresses a basic problem of security systems that operate on very sensitive information. Specifically, we are interested in the problem of privately handling numeric data represented by rational numbers (e.g., medical records). Fully homomorphic encryption (FHE) is one of the natural and powerful tools for ensuring privacy of sensitive data, while allowing complicated computations on the data. However, because the native plaintext domain of known FHE schemes is restricted to a set of quite small integers, it is not easy to obtain efficient algorithms for encrypted rational numbers in terms of space and computation costs. For example, the naïve decimal representation considerably restricts the choice of parameters in employing an FHE scheme, particularly the plaintext size.

Our basic strategy is to alleviate this inefficiency by using a different representation of rational numbers instead of naïve expressions. In this work we express rational numbers as *continued fractions*. Because continued fractions enable us to represent rational numbers as a sequence of integers, we can use a plaintext space with a small size while preserving the same quality of precision. However, this encoding technique requires performing very complex arithmetic operations, such as division and modular reduction. Theoretically, FHE allows the evaluation of any function, including modular reduction at encrypted data, but it requires a Boolean circuit of very high degree to be constructed. Hence, the primary contribution of this work is developing an approach to solve this efficiency problem using homomorphic operations with *small degrees*.

Keywords: Continued fractions; Gosper algorithm; Rational numbers; Homomorphic encryption.

1. Introduction

Consider the following scenario. There is a server that stores patients' medical data, and it has considerable computing power such that it can compute a predictive

*The corresponding author

model for each patient and inform patients on whether they are in the danger range.

For example, a Cox model is a statistical technique for exploring the relationship between the survival of a patient and several explanatory variables, and it provides an estimate of the effect of treatment on survival after adjusting for other explanatory variables. In addition, this technique allows us to estimate the hazard (or risk) of death for an individual given their prognostic variables. The prognostic variables are age, diabetes, smoking, systolic blood pressure, cholesterol, and HDL cholesterol. For instance, in reference [1], the predictive model for females is given by the function

$$\Pr_{\mathsf{Female}}(X_i) = 1 - 0.95012^{\exp(\sum_i \beta_i X_i - 26.1931)} \tag{1}$$

where X_i is the input for each risk factor, $\exp(\cdot)$ is the exponential function, and β_i is the regression coefficient. Specifically, the regression coefficients for the model for females are given by

$$\begin{aligned} \textstyle\sum_i \beta_i X_i = {} & 2.32888 \cdot \log A + 1.20904 \cdot \log C - \\ & 0.70833 \cdot \log H + 2.76157 \cdot \log B + \\ & 0.52873 \cdot S + 0.69154 \cdot D \end{aligned} \tag{2}$$

where A denotes the age, C denotes the cholesterol level, H denotes the HDL cholesterol level, B denotes the systolic blood pressure, and $D = 1$ if diabetes exists and $S = 1$ if an individual is a smoker; otherwise, $D = 0$ and $S = 0$.

To make this practical scenario work in the real world, we need to satisfy the following two basic requirements before other technical ones.

Privacy. For privacy reasons, one appealing approach is to have patients keep all medical data encrypted if possible. However, because decryption on the server side could cause a loss of privacy, we need to apply an encryption scheme that allows homomorphism to such sensitive data such that computations on its encryptions do not require decryption.

Inter-domain conflict. Again, let us examine Eq. (2). Due to the privacy requirement, we now need to compute the equation on the ciphertext domain of an underlying encryption scheme rather than its plaintext domain. The key point here is that the underlying encryption scheme should be able to take as input rational numbers such as HDL cholesterol values, but in general, encryption schemes are restricted to encrypt group or ring elements.

To resolve this conflict between the domain of a target function and the domain of an encryption scheme, message encoding before encryption appears to be the best choice. One widely accepted approach to encode a rational plaintext r is to use decimal expansion. Let $r \in \mathbb{Q}$ be a rational number that has ℓ-digit below a decimal point. Given a rational number $r = r_0.r_1r_2 \cdots r_\ell$, an encryption scheme with decimal expansion first converts each digit r_i into $x_i = r_i \cdot 10^{i-1} \in \mathbb{Z}, 1 \leq i \leq \ell$ and then encrypts each x_i. This implies that the plaintext space of the encryption scheme must be large to represent an integer $x_\ell = r_\ell \cdot 10^{\ell-1}$. For example, suppose

that a predictive algorithm uses a rational number $r = 0.1357908642$. Then, in this case, $x_{10} = 2 \cdot 10^9 \approx 2^{31}$, and thus, a proper plaintext space superficially seems to be $\mathbb{Z}_{2^{31}}$.

However, if we think a little bit more about our choice of the plaintext space, we see that $\mathbb{Z}_{2^{31}}$ is not always a correct one under decimal expansion. To explain why this is so, suppose that a fully homomorphic encryption (FHE) scheme and a target function f for evaluation are given. One may simply think that he only has to fix t (e.g., $t = 2^{31}$) to the FHE's plaintext space $\mathbb{Z}_t$. Apparently it is not the case. Because the encryption scheme should perform modular reduction when an evaluation result of f over FHE ciphertexts carries a plaintext larger than t, the plaintext space can be determined only after estimating the target evaluation function. For example, when we consider a proper plaintext space for encoding $r = 0.1357908642$, $\mathbb{Z}_{2^{31}}$ may not be a good choice because we may have to encode $8 \cdot r$. Specifically, $1357908642 \times 8 = 10863269136 \approx 2^{34}$ and thus $10863269136 \bmod 2^{31}$ should be performed. Hence, we need to set $t \geq 2^{34}$. The more are required the number of multiplications, the bigger plaintext space should be chosen. In conclusion, a proper plaintext space depends on a target evaluation function in order to avoid modular reduction.

Furthermore, since the size of the noise after homomorphic operations is proportional to t, it may lead to an efficiency problem. For example, consider the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [3]. If $\mathbb{Z}_2$ is taken as the plaintext space, one homomorphic multiplication consumes only one multiplicative depth roughly. However, in the case of $\mathbb{Z}_{2^{15}}$, we see that the same operation consumes two multiplicative depths, when implemented by the HElib library [15].

Group homomorphic encryption (e.g., El Gamal [10] and Paillier [19]) can also be considered. However, this type of encryption scheme is not suitable for bitwise operations because it severely wastes their plaintext space. For example, consider an equality check between two encrypted hematocrit values. In general, an equality comparison cannot be efficiently performed over Paillier ciphertexts because equality test works on bit representations of group elements.

Taking two requirements and the two together, we need to provide a better way of representing rational numbers to fit in homomorphic evaluations in the predictive model such that multiplications by rational numbers are needed. In this work, our technical goal is to represent rational numbers while using a small-sized plaintext space, but without losing its original precision.

Overview of our idea. Restricting our interest to the prediction model, such as Eq. (2) leads to the below observations that will be helpful to understand our design directions.

(1) Carefully looking into specific target functions used in the practical side, coefficients and input values are given in *rational numbers after truncating their lower digits*, but not in real numbers.
(2) Regression formulas of our interest perform multiplication between the encrypted medical data and *some known constant values* (e.g., coefficients

2.32888 and 1.20904 of Eq. (2)). This makes us free from performing multiplication of two encrypted medical data.

Given a rational number r, there exist an unique integer q, called the integral part of r, and an unique rational $s \in [0, 1)$, called the fractional part of r, such that $r = q + s$. Roughly, continued fractions (CFs)[a] can represent the same rational numbers using only relatively quite small integers when their integral part is also a small integer. Thus we restrict our attention to how to efficiently handle rational numbers whose integral part is small, in a private manner. In this setting, we expect that it can provide significant efficiency gains, even in the case where elements of the set should be encrypted, with respect to the size of ciphertexts and later homomorphic arithmetic operations. For example, consider the same $r = 0.1357908642$ as above. Then, we can represent the r as a sequence of integers

$$[0; 7, 2, 1, 2, 1, 12, 2, 2, 7, 7, 2, 2, 1, 11, 2]$$

and indeed, it is a sequence of only 4-bit integers. That is, we can take $\mathbb{Z}_{2^4}$ rather than $\mathbb{Z}_{2^{31}}$ as the plaintext domain. As a result, if the integral part of a rational r is at most a length of ℓ bits, then it is expected that one can take $\mathbb{Z}_{2^\ell}$ rather than $\mathbb{Z}_{2^{\ell+31}}$ as the plaintext domain. We will discuss how to obtain each integer of the sequence in later sections. Accordingly, when we wish to homomorphically evaluate a function f on encrypted data, we first encode all inputs into a sequence of small integers and encrypt each small integer under an underlying FHE scheme. This allows the FHE scheme to be instantiated with a small-sized plaintext space. We notice that the second observation states that the coefficients of f need not be encrypted, unlike the input values.

Regarding a class of target functions with which we can efficiently deal, we will focus on a linear multivariate polynomial f. Even though all partial quotients are bounded after arithmetic operations (e.g., see [6, 20]), dealing with polynomials of higher degree requires to run with a large-sized plaintext space. Then we can no longer utilize benefits of using FHE while enjoying a proper level of efficiency. This is the primary reason for only considering rational polynomials of low degree at the cost of applicability's limitation. For these reasons, our solution may not cover a wide range of applications running on FHE.

When encoding a rational number into CF, we use the so-called Gosper algorithm [13]. However, some nontrivial problems arise when one has to perform computations on encrypted rational numbers that were encoded by the algorithm. The essential technical obstacle among these problems is that the algorithm requires modular arithmetic and division. Theoretically, FHE can compute arbitrary functions on encryptions, but in practice, both operations are fairly difficult to efficiently implement when their operands are ciphertexts. Throughout the remainder of the

[a]Precisely speaking, we only consider finite and simple continued fraction. That is, the number of partial quotients is finite and all partial quotients and the integral part are integers.

paper, we thus seek a solution to replace modular arithmetic and division with other operations with a *small degree*, i.e., efficiently realized. Indeed, we develop a variant of Gosper algorithm that works on ciphertexts of rational numbers encoded using the CF technique. Throughout the paper, by degree, we mean the multiplication depth required to perform a function.[b] Roughly, given a circuit associated with an FHE scheme, its multiplication depth means the total number of reduced levels in the circuit being homomorphically evaluated. We will formally define it later (see Definition 3 in Section 3.3).

Summary of our results. In summary, our contributions are as follows:

(1) We show that we can enable to divide an integer by an encrypted integer with a special case of Gosper algorithm. Further, we can compute a linear fractional transformation evaluation whose variables are rational numbers represented by continued fractions. In particular, the computational complexity for evaluating the linear fractional transformation is almost the same as the special case of Gosper algorithm, regardless of polynomial's coefficients.
(2) We address some efficiency challenges posed by using the Gosper algorithm during the encoding of rational numbers. Specifically, modular reduction over ciphertexts is very expensive in terms of multiplicative depth and thus we develop an approach to substitute a combination of other operations requiring a quite *less* multiplicative degree for modular reduction. We apply this technique to the original Gosper algorithm.

The outline of the paper. We firstly provide a survey of closely related works in Section 2. Section 3 provides a series of materials for better understanding our work. We describe our main result in Section 4 and Section 5, along with analysis of correctness and computational costs.

2. Study of the Existing

As related works, we first investigate Graepel et al.'s result [14] and Bos et al.'s work [1]. In both works, the authors first fix a desired precision, multiply through by a fixed denominator, and round to the nearest integer because any real number can be approximated by rational numbers to arbitrary numbers and subsequently encoded to ring elements. However, this approach to represent real numbers has some drawbacks. When two encoded rational numbers are multiplied, it should be performed without any modular reduction and thus the plaintext space of FHE must be sufficiently large.

The most closely related work to the present one is Jäschke et al.'s scheme and Costache et al.'s scheme [8]. In [17], Jäschke and Armknecht dealt with a

[b]Of course, the function should first be transformed into a binary Boolean circuit or an arithmetic circuit.

particular encoding for rational numbers in the FHE context. However, due to their specialty of encoding, Boolean comparison on encrypted data of n-bit length should be inefficiently implemented since $O(n)$ multiplication depth is needed rather than $O(\log n)$. More recently, the work of [8] investigated fixed-point arithmetic in ring-based somewhat homomorphic encryption (SHE). However, this approach does not allow to compute a reciprocal of encrypted data, nor does it support integer division.

3. Toolkits

In this section, we provide some basic materials to better understand the remainder of the paper. Background of the two different fields is required: one is mathematics, and the other is cryptography.

Notation. For readability, we introduce some notation and terminology for the rest of the paper.

- A bar over some integer means that the integer is encrypted by an FHE encryption algorithm E; $\bar{x} = \mathsf{E}(x)$ for $x \in \mathbb{Z}$.
- A maximum of a continued fraction signifies that the largest integer among its partial quotients, namely, for a continued fraction $x = [x_0; x_1, x_2, \ldots]$,
$$\max x := \max_i |x_i|$$
- A continued fraction is encrypted means that each partial quotient of the continued fraction is encrypted. For a continued fraction $x = [x_0; x_1, x_2, \ldots]$,
$$\mathsf{E}(x) := [\mathsf{E}(x_0); \mathsf{E}(x_1), \mathsf{E}(x_2), \ldots]$$
which can be abbreviated as $\bar{x} := [\bar{x}_0; \bar{x}_1, \bar{x}_2, \ldots]$.
- For a rational number x, we denote its fractional linear transformation $\frac{a+bx}{c+dx}$ for $a, b, c, d \in \mathbb{Z}$ by $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$.
- For two algorithms A and B, $\mathsf{A}(z)$ means that A takes as an input z and $\mathsf{B} \circ \mathsf{A}(z)$ means call A and B in order as an input z.

3.1. *Mathematical Tools*

A set of integers is closed under addition and multiplication, but not division. However, there are some methods representing rational numbers to integers, e.g., continued fractions and decimal representations. Continued fractions are more *mathematically natural* representations of rational numbers than decimal representations. First of all, the continued fraction representation for a rational number is finite, but decimal representation for a rational number may be infinite. Moreover, every rational number has an unique continued fraction representation with some restrictions. The successive approximations generated in finding the continued fraction representation of a number, i.e., by truncating the continued fraction representation, are in a certain sense (described below) the best possible. Therefore, it has occasionally

been considered to be a great tool in mathematics and it has been researched for a long time in a variety of topics. Here, we rephrase only those related to our works.

Continued fractions. A continued fraction can be obtained through an iterative process of representing a number as the sum of its integer part and the reciprocal of remaining part, and then writing the remaining part as the sum of its integer part and remaining part, and so on. In other words, given a real number x, we have

$$x = x_0 + \frac{1}{r_0} = x_0 + \frac{1}{x_1 + \frac{1}{r_1}} = x_0 + \frac{1}{x_1 + \frac{1}{x_2 + \frac{1}{r_2}}} = \cdots \tag{3}$$

where $r_i > 1$ for all i and use $x = [x_0; x_1, x_2, \ldots]$ to denote this. Note that x_0 can be any integer, but for $i \in \mathbb{N}, x_i$ must be positive and with this restriction, the continued fraction of x is unique. We can define further terminologies related to continued fractions.

Definition 1 ([16]) *For a continued fraction $x = [x_0; x_1, \ldots]$,*

- *x_i is called a partial quotient of x for all i.*
- *A continued fraction x is finite if the number of partial quotients of x is finite.*

Keeping this definition in mind, the following theorem states that the correspondence between rational numbers and a finite continued fraction $[a_0; a_1, \ldots, a_n]$ with an integer a_0 and positive integer a_i for $i > 0$ and $a_n > 1$ is one-to-one. We consider the situation that an integer is divided by another integer, so only rational numbers are considered in this paper.

Theorem 1 ([16]) *Any rational number can be represented as a finite continued fraction and the continued fraction representation is unique when the last partial quotient is larger than 1.*

Theorem 2 ([16]) *Let $\alpha = [a_0; a_1, a_2, \cdots]$ and $p_i/q_i = [a_0; a_1, \cdots, a_i]$. For any rational number $\frac{a}{b}$ with $a \in \mathbb{Z}$ and $b \in \mathbb{N}$, and $1 \leq b \leq q_i$, $\left|\frac{p_i}{q_i} - \alpha\right| \leq \left|\frac{a}{b} - \alpha\right|$, with equality if and only if $a/b = p_i/q_i$.*

Theorem 2 indicates that p_i/q_i is the best possible approximation to α among all rational numbers with the same or smaller denominator. In other words, a continued fraction is the best approximation tool of rational numbers.

Lastly, we give an assertion that the partial quotients have a small size in the bit length. With some restrictions in Theorem 1, continued fractions and rational numbers have one-to-one correspondence, however, one can find two different continued fractions for a rational number when these restrictions are weaken. For some $k \in \mathbb{N}$, we can easily derive that $[\ldots, a_k + a_{k+1}, \ldots] = [\ldots, a_k, 0, a_{k+1}, \ldots]$ because

$$\frac{1}{a_k + a_{k+1} + \frac{1}{a_{k+2}}} = \frac{1}{a_k + \frac{1}{0 + \frac{1}{a_{k+1} + \frac{1}{a_{k+2}}}}} \tag{4}$$

For this reason, we can insist that one can find a continued fraction with bounded partial quotient for an arbitrary rational number even though the number of partial quotients gets longer. Thus, a user working with rational numbers can determine the maximum of partial quotients, considering his environment. For example, if sufficient space is available, then he may reduce the maximum size of partial quotients at the cost of increasing the number of ciphertexts.

3.1.1. *Arithmetics over CF Representations*

Given two rational numbers x and y represented by decimal expansion, it is easy to compute $z = x + y$, as we have learned. However, we have not learned an arithmetic algorithm if one of the rational numbers is given by continued fraction form. In 1972, Bill Gosper [13] proposed the general arithmetic algorithm on continued fractions. This algorithm enables arithmetics between two continued fractions as well as a continued fraction and a rational number. We recall a brief description of Gosper algorithm.

Gosper algorithm. The goal of Gosper algorithm is, for a real number x, computing $\frac{a+bx}{c+dx}$ for $a, b, c, d \in \mathbb{Z}$ because every elementary single operation of two rational numbers, such as addition, subtraction, multiplication and division, can be replaced by a linear fractional transformation. Of course, it can also represent various equations in addition to the elementary single operations, and thus, we can calculate more complex arithmetic than elementary operations.

Suppose that x is a rational number with a continued fraction form and that $z := \frac{a+bx}{c+dx}$ for $a, b, c, d \in \mathbb{Z}$. The main idea is that z has a value between $\frac{a}{c}$ and $\frac{b}{d}$ for almost everywhere, and thus, if $\frac{a}{c}$ and $\frac{b}{d}$ have the same integer part, the integer part of z is $\lfloor \frac{a}{c} \rfloor = \lfloor \frac{b}{d} \rfloor$; otherwise, z requires more information about x. Because z behaves differently for each case, it requires two sub-algorithms, say InTake and OutTake. For convenience, we denote z by $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$.

The following algorithm (see Algorithm 1) is the Gosper algorithm for an arithmetic algorithm on continued fractions using the above idea. For each iteration, call either OutTake or InTake according to the condition of whether two truncations are the same.

The details of OutTake algorithm. It occurs when z exactly knows its own integer part, as we stated above. The integer part of z should be $q := \lfloor \frac{z_0}{z_2} \rfloor$ if the value of the flooring two ratios is the same. Thus, z can determine its own integer part by comparing the two values, and when they have the same value, z must emit q and becomes the reciprocal of $z - q$.

Algorithm 1 Gosper Algorithm

Input: $x = [x_0; x_1, x_2, \ldots], f(X) = \frac{a+bX}{c+dX}$
Output: $f(x) = [y_0; y_1, y_2, \ldots]$

$\begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix} \leftarrow \begin{pmatrix} a & b \\ c & d \end{pmatrix}$
$i \leftarrow 0, j \leftarrow 0$
while z_2 and z_3 are not all zero **do**
  $q_1 \leftarrow \lfloor z_0/z_2 \rfloor, q_2 \leftarrow \lfloor z_1/z_3 \rfloor$
  if $q_1 = q_2$ **then**
    $y_i \leftarrow q_1$
    $\begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix} \leftarrow \begin{pmatrix} z_2 & z_3 \\ z_0 - z_2 y_i & z_1 - z_3 y_i \end{pmatrix}$ /* InTake */
    $i \leftarrow i + 1$
  else
    $\begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix} \leftarrow \begin{pmatrix} z_1 & z_0 + z_1 x_j \\ z_3 & z_2 + z_3 x_j \end{pmatrix}$ /* OutTake */
    $j \leftarrow j + 1$
  end if
end while
return $[y_0; y_1, y_2, \ldots]$

Let $z' = \frac{1}{z-q}$. Then,

$$z' = \frac{1}{z-q} = \left(\frac{a+bx}{c+dx} - q\right)^{-1} = \left(\frac{a-cq+bx-dqx}{c+dx}\right)^{-1}$$
$$= \frac{c+dx}{a-cq+(b-dq)x} = \begin{pmatrix} c & d \\ a-cq & b-dq \end{pmatrix} \tag{5}$$

Because $a = cq + a \bmod c$, $a \bmod c = a - cq$; thus, OutTake actually performs the modular arithmetic. In summary, if $z = \begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix}$ knows its own integer part, then it becomes

$$\begin{pmatrix} z_2 & z_3 \\ z_0 \bmod z_2 & z_1 \bmod z_3 \end{pmatrix} \tag{6}$$

The details of InTake algorithm. It occurs when z does not exactly know its own integer part. If the value of the two ratios is not the same, then z cannot determine which integer is correct, and thus, z requires more information of x and thus requests a term from x. In this case, additional information of x is its partial quotient.

Let $x = p + \frac{1}{x'}$ for some x'. Then,

$$z = \frac{a+bx}{c+dx} = \frac{a+b(p+1/x')}{c+d(p+1/x')} = \frac{b+(a+bp)x'}{d+(c+dp)x'}$$
$$= \begin{pmatrix} b & a+bp \\ d & c+dp \end{pmatrix} \tag{7}$$

In summary, if $z = \left(\begin{smallmatrix} z_0 & z_1 \\ z_2 & z_3 \end{smallmatrix}\right)$ requests more information from x and obtains p, then it becomes $\left(\begin{smallmatrix} z_1 & z_0+z_1p \\ z_3 & z_2+z_3p \end{smallmatrix}\right)$. Additionally, InTake takes a partial quotient of x one at a time. Because

$$\cdots + \frac{1}{x_n} = \cdots + \frac{1}{x_n + \frac{1}{\infty}}, \tag{8}$$

it is a fact that $[\,, \ldots, x_n] = [\,, \ldots, x_n, \infty]$. Note that ∞ is just a symbolic of the last partial quotient of any rational number and so we do not need to store and encrypt ∞. When InTake takes an input ∞, $z = \left(\begin{smallmatrix} z_0 & z_1 \\ z_2 & z_3 \end{smallmatrix}\right)$ becomes $\left(\begin{smallmatrix} z_1 & z_1 \\ z_3 & z_3 \end{smallmatrix}\right)$ because $b \cdot \infty$ and $d \cdot \infty$ dominate any a and c, respectively; thus, two columns of z always behave in the same way after taking ∞. This implies that z always has the same integer part, and thus, $x = \infty$ is the last input of InTake.

A toy example. We will provide a brief example of how this algorithm actually works. Let $x = \frac{13}{11} = [1; 5, 2]$ and $f(X) = X + \frac{1}{2} = \frac{1+2X}{2}$.

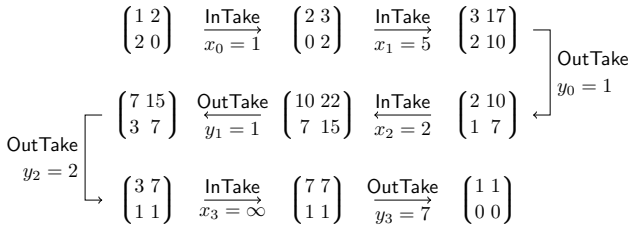

Fig. 1. A toy example of Gosper algorithm

Because $\mathsf{OutTake}$ yields $y_0 = 1, y_1 = 1, y_2 = 2$ and $y_3 = 7$, $y = f(x) = [y_0; y_1, y_2, y_3] = [1; 1, 2, 7] = 37/22$. It is trivial that $13/11 + 1/2 = 37/22$ and thus we can verify this algorithm works correctly in this case.

3.2. *Fully Homomorphic Encryption*

An FHE scheme, denoted by $\mathsf{FHE} = (\mathsf{Kg}, \mathsf{E}, \mathsf{D}, \mathsf{Ev})$, is a quadruple of probabilistic polynomial-time algorithms, as follows.

Key generation. The algorithm takes the security parameter λ and outputs a public encryption key pk, a public evaluation key ek and a secret decryption key sk. We write the algorithm as $(pk, ek, sk) \leftarrow \mathsf{Kg}(1^\lambda)$ and assume that the public key specifies the plaintext space P and the ciphertext space C.

Encryption. The algorithm $\bar{x} \leftarrow \mathsf{E}_{pk}(x)$ takes the public key pk and a message $x \in \mathsf{P}$ and outputs a ciphertext $\bar{x} \in \mathsf{C}$.

Decryption. The algorithm $x^* \leftarrow \mathsf{D}_{sk}(\bar{x})$ takes the secret key sk and a ciphertext $\bar{x}$ and outputs a message $x^* \in \mathsf{P}$.

Homomorphic evaluation. The algorithm takes the evaluation key ek, a function $f : (\{0,1\}^*)^n \to \{0,1\}^*$, and a set of n ciphertexts $\bar{x}_1, \ldots, \bar{x}_n$ and outputs a ciphertext $\bar{x}_f$, denoted by $\bar{x}_f \leftarrow \mathsf{Ev}_{ek}(f, \bar{x}_1, \ldots, \bar{x}_n)$.

Concrete instantiation and security. In 2009, since Gentry's first secure FHE scheme from ideal lattices [11], various studies [7, 9, 21] have been presented on constructing efficient FHE schemes. However they have fairly poor performance. As a solution of efficient FHE, Brakerski and Vaikuntanathan [4] introduced the concept of leveled FHE schemes which allows the evaluation of functions of at most a pre-determined multiplicative depth, instead of arbitrary functions. Shortly after, Brakerski, Gentry, and Vaikuntanathan [3] proposed a leveled FHE scheme over polynomial rings, which has significantly improved performance over the previous schemes. Therefore, there are several good candidates for instantiating FHE; examples include the BGV scheme [3] and Bos et al.'s scheme [2]. We notice that our technique can work on any FHE scheme.

To our knowledge, Bos et al.'s scheme is efficient when the plaintext space is small because it is scale-invariant and its ciphertext consists of only one single ring element. However, since an implementation of the BGV scheme is available in an open source library, named HElib [15], it has been widely deployed by a variety of applications. For the purpose of better performance, we recommend the use of an FHE scheme supporting SIMD-type (single-instruction multiple-data) operations and the Frobenius map in a depth-free manner.

An FHE scheme is said to be semantically secure if it achieves indistinguishability against chosen plaintext adversaries. We use a widely known formulation of semantic security [12], defined as follows.

Definition 2 (Semantic Security) *An FHE scheme is semantically secure if for any polynomial-time adversary $\mathcal{A}$, it holds that*

$$|\Pr[\mathcal{A}(pk, \mathsf{E}(pk, m_0)) = 1] - \Pr[\mathcal{A}(pk, \mathsf{E}(pk, m_1)) = 1]|$$

is negligible in security parameter λ where $\mathsf{Kg}(1^\lambda) \to (pk, ek, sk)$ and $m_0, m_1 \in \mathsf{P}$ are chosen by the adversary $\mathcal{A}$.

Privacy of CF-based arithmetics against a semi-honest server follows from the semantic security of an underlying FHE scheme. Thus, the security proof is straightforward and we can omit it.

3.3. *Circuit Construction of two Main Submodules*

We provide a brief description of underlying circuits for our construction. The below two circuits are used in replacing modular reduction with cheaper operations in terms of multiplicative depth. Multiplicative depth has a significant effect on the performance, and thus it plays an important role in terms of complexity.

Definition 3. Multiplicative depth *of the circuit under homomorphic encryption is the total number of reduced levels in a circuit that is being evaluated homomorphically.*

Greater-than comparison. For two n-bit integers, a greater-than circuit $\mathsf{GT}(\bar{x}, \bar{y})$ outputs $\bar{1}$ if $x \geq y$ and $\bar{0}$ otherwise. This operation can be recursively defined as follows:

$$\mathsf{GT}(\bar{x}, \bar{y}) = 1 - \bar{c}_{n-1},$$

where $\bar{c}_i = (1 + \bar{x}_i) \cdot \bar{y}_i + (1 + \bar{x}_i + \bar{y}_i) \cdot \bar{c}_{i-1}$ for $i \geq 1$ with an initial value $\bar{c}_0 = (1 + \bar{x}_0) \cdot \bar{y}_0$.

It is easy to see that since c_{n-1} has degree $n+1$, it has $\lceil \log(n+1) \rceil$ multiplication depth. Moreover, a naïve construction of the circuit requires $\frac{n^2+n}{2}$ homomorphic multiplications. However, by carefully applying SIMD operations, the circuit can be improved to require only $2n-2$ homomorphic multiplications (see reference [5]). In this case, the circuit has a multiplicative depth of $\lceil \log n \rceil + 1$.

Full adder. Let x and y be n-bit integers. We then obtain two $\tilde{n}$-bit integers by padding zeros on the left for an integer $\tilde{n} > n$. We define a full-adder of size $\tilde{n}$, denoted by FA, in a recursive manner as follows:

$$\mathsf{FA}(\bar{x}, \bar{y}) = (\bar{z}_0, \bar{z}_1, \ldots, \bar{z}_{\tilde{n}-1})$$

where a sum $\bar{z}_i = \bar{x}_i + \bar{y}_i + \bar{c}_{i-1}$ and a carry-out $\bar{c}_i = (\bar{x}_i \cdot \bar{y}_i) + ((\bar{x}_i + \bar{y}_i) \cdot \bar{c}_{i-1})$ for $1 \leq i \leq \tilde{n} - 1$ with initial values $\bar{z}_0 = \bar{x}_0 + \bar{y}_0$ and $\bar{c}_0 = \bar{x}_0 \cdot \bar{y}_0$. The more important thing with respect to efficiency is that we have an integer addition circuit of depth $\lceil \log(\tilde{n}-2) \rceil + 1$ if an underlying FHE scheme supports SIMD operations and depth-free Frobenius map evaluation [5]. Indeed, while its naïve construction of the circuits incurs $(\tilde{n}^3 - 6\tilde{n}^2 + 8\tilde{n})/6$ homomorphic multiplications, its optimized constructions requires only $3\tilde{n} - 5$ homomorphic multiplications.

Even though these primitive circuits are designed to merely process encrypted integers, because our solution encodes rational numbers as a sequence of small integers (i.e., partial quotients) and works on their encryptions, we can assemble them into our solution without extra computational cost.

4. Basics

In a nutshell, our main goal is to make the Gosper algorithm, which is originally designed to run in the clear, work over FHE ciphertexts, and it will be described in Section 5. But before, we deal with *a simple case* in which the Gosper algorithm takes as input an encrypted integer but not rational number. By "simple" we deal with only a CF whose partial quotients are a singleton. Then we will return the main topic after verifying our idea via the case study.

Running Gosper algorithm on FHE settings. When running Gosper algorithm on FHE ciphertexts, there are two main technical challenges. As shown in

Algorithm 1, the first challenge is that Gosper algorithm needs to perform modular arithmetic over encrypted partial quotients, while computing an output in a CF representation. Recall that Eq. (6) in OutTake requires to perform the modular reduction operation.

It is obvious that modular arithmetic of integers in the clear is quite easy to compute. However, because performing modular arithmetic requires to repeatedly invoke division by integer, it is non-trivial to run the algorithm on FHE ciphertexts. Of course, FHE theoretically enables to evaluate arbitrary functions at FHE ciphertexts. However, there is no efficient way of performing divisions between two FHE ciphertexts in the practical sense. To address this problem, we need to develop an efficient way for dividing an integer by an encrypted integer. Consequently, we will focus on developing an efficient way to perform modular arithmetic on FHE ciphertexts later in the section.

The second challenge is that Gosper algorithm needs to evaluate a conditional branch on encryptions. Specifically, the result of a conditional expression (`if-then-else`) is used to determine which sub-algorithm (i.e., either InTake or OutTake) should be invoked. However, we can solve this problem by modifying the Gosper algorithm so that the invocation of sub-algorithm can be pre-determined.

4.1. *Efficient Modular Arithmetic over FHE Ciphertexts*

To our knowledge, there are no efficient division algorithm and thus, modular arithmetic algorithm over FHE ciphertexts. As mentioned above, however, the OutTake sub-module in the Gosper algorithm requires modular arithmetic. For efficiency reasons, modular arithmetic should be *replaced* by different operations of low degrees.

Our key idea for an alternative to modular arithmetic is subtracting a modulus while repeatedly applying greater-than comparison circuit. Let a be an integer and b be a modulus. The modular reduction of a with respect to modulus b can be done by subtracting b from a until the difference is smaller than b. We have a GT circuit that allows to compare two encryptions in a bit-by-bit manner. Furthermore, the circuit has a multiplicative depth of $\lceil \log n \rceil + 1$ where n is the bit length of input plaintexts. As a result, the remaining issue is to determine the number of homomorphic subtractions required for this procedure.

The following theorem that we present here is needed for ensuring the efficiency of our approach. More formally, Theorem 3 indicates that every partial quotient is bounded by denominator and numerator and then, with the boundedness, modular arithmetic can be replaced by bounded invocations of the GT circuit, which requires a quite smaller multiplicative depth than that of a division circuit.

Theorem 3. *Let $x = [x_0; x_1, \ldots, x_{n-1}]$ be a continued fraction of p/q, where $\alpha \in \mathbb{Q}$ and $x_i, p, q \in \mathbb{Z}$. Then,*

$$\log x_0 + \log x_1 + \cdots + \log x_{n-1} < \log p$$
$$\log x_0 + \log x_1 + \cdots + \log x_{n-1} < \log q$$

Proof. Let $x = [x_0; x_1, \ldots, x_{n-1}]$, where $x \in \mathbb{Q}$, $x_i \in \mathbb{Z}$ and its convergents $C_i = [x_0; x_1, \ldots, x_i] = p_i/q_i$. By mathematical induction on i, we can easily derive that the numbers p_i and q_i can be obtained by the recurrence

$$p_i = x_i p_{i-1} + p_{i-2} \tag{9}$$

$$q_i = x_i q_{i-1} + q_{i-2} \tag{10}$$

with initial conditions $p_0 = x_0, p_{-1} = 1, q_0 = 1$, and $q_{-1} = 0$. We can rewrite Eq. (9) and Eq. (10) with a matrix as follows.

$$\begin{aligned}
\begin{pmatrix} p_i & p_{i-1} \\ q_i & q_{i-1} \end{pmatrix} &= \begin{pmatrix} p_{i-1} & p_{i-2} \\ q_{i-1} & q_{i-2} \end{pmatrix} \begin{pmatrix} a_i & 1 \\ 1 & 0 \end{pmatrix} \\
&= \begin{pmatrix} p_{i-2} & p_{i-3} \\ q_{i-2} & q_{i-3} \end{pmatrix} \begin{pmatrix} x_{i-1} & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_i & 1 \\ 1 & 0 \end{pmatrix} \\
&= \cdots \\
&= \begin{pmatrix} x_0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} x_i & 1 \\ 1 & 0 \end{pmatrix}.
\end{aligned}$$

Therefore, for all i, p_i and q_i contain $x_0 x_1 \cdots x_i$ and there is no negative term, which implies that $p_i > x_0 x_1 \cdots x_i$ and $q_i > x_0 x_1 \cdots x_i$. Taking a logarithm with base 2 for both inequalities, one can obtain which was to be demonstrated. □

As an example, consider $x = \frac{13}{11} = [1; 5, 2]$. By Theorem 3, we can easily see that $\log 13 > \log 1 + \log 5 + \log 2$ and $\log 11 > \log 1 + \log 5 + \log 2$.

4.2. *Our Suggestion*

An integer in decimal form is exactly the same as the value in CF expansion because it has only one single partial quotient. Hence, the InTake sub-module is invoked only once, but the OutTake sub-module is invoked multiple times. First, we need to determine the number of invocations of OutTake. Then because OutTake outputs one partial quotient per invocation, the number of invocations is equal to the number of partial quotients. Second, we need to know the number of times that the GT circuit is executed to compute each partial quotient. Indeed, in our suggested algorithm, the outer loop is dependent on the number of invocations and the inner loop is dependent on the number of times that we have to execute the GT circuit.

To develop a better understanding of determining the number of invocations of OutTake, we look at the CF algorithm from a different angle. To this end, we compare this with Euclidean algorithm. Let a and b be integers. For computing $\gcd(a, b)$, Euclidean algorithm starts with $a = q_1 b + r_1$. This is the same as $\lfloor \frac{a}{b} \rfloor = q_1$ and $\frac{a}{b} = q_1 + \frac{r_1}{b}$, which is the first iteration of the CF algorithm. The next step in Euclidean algorithm is $b = q_2 r_1 + r_2$. Similarly, it is also equal to $\lfloor \frac{b}{r_1} \rfloor = q_2$ and $\frac{b}{r_1} = q_2 + \frac{r_2}{r_1}$, which is the second iteration of the CF algorithm. Continuing in this

way, we finally get to the last line of the Euclidean algorithm: $r_{n-2} = q_n r_{n-1} + 0$. In the continued fraction, we have $\frac{a}{b} = [q_1, q_2, \ldots, q_n]$. This implies that the Euclidean algorithm and the CF algorithm for rational numbers are essentially the same. Thus, we can set the number of invocations of $\mathsf{OutTake}$ as the iterations n in the Euclidean algorithm.

By Theorem 3, every partial quotient y_i is bounded by denominator x and numerator a. Since x is encrypted, we assume every partial quotient is bounded by a in this paper. This assumption is again justified by Theorem 3.

Algorithm 2 Our Variant of Gosper Algorithm

Input: $\bar{x}, f(X) = \frac{a}{X}$
Output: $f(\bar{x}) = [\bar{y}_0; \bar{y}_1, \ldots, \bar{y}_{n-1}]$

/* InTake */
$\begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix} \leftarrow \begin{pmatrix} a & a \\ \bar{x} & \bar{x} \end{pmatrix}$
$n \leftarrow$ the number of ciphertexts of $\bar{x}$
/* OutTake */
for $i = 0$ to $n - 1$ **do**
  $\bar{y}_i \leftarrow 0$
  $tmp \leftarrow z_1$
  for $j = 0$ to a **do**
    $\bar{t} \leftarrow \mathsf{GT}(tmp, z_3)$
    $\bar{y}_i \leftarrow \bar{y}_i + \bar{t}$
    $tmp \leftarrow tmp - \bar{t} \cdot z_3$
  end for
  $\begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix} \leftarrow \begin{pmatrix} z_3 & z_3 \\ tmp & tmp \end{pmatrix}$
end for
return $[\bar{y}_0; \bar{y}_1, \ldots, \bar{y}_{n-1}]$

4.3. *Analysis*

In this section, we analyze the performance of Algorithm 2 in terms of the number of multiplications. Note that multiplication over FHE ciphertext is the most expensive operation.

Correctness. We first need to argue the correctness of our suggestion, which is a special case ($n = 1$) of the Gosper algorithm. Algorithm 2 is fundamentally based on the Gosper algorithm and thus we compare the Gosper algorithm and our suggestion.

Without loss of generality, we may assume that two truncate ratios are different and so InTake always incurs at first. This is the same procedure that the Gosper

algorithm, but only differs whether or not encryption. After executing InTake, the remaining step is to execute OutTake until the algorithm is over. OutTake requires modular reductions, but we do not perform modular reductions because we know the upper bound of partial quotients by Theorem 3. Modular reductions between a and b ($a > b$) in the clear can be viewed as continually subtracting b from a until b is larger and the number of subtract is quotient of a and b. In this sense, this should be the same value compared with the output of OutTake, which corresponds to line 5 to line 10.

To sum up, every line of Algorithm 2 corresponds to the original algorithm and every step does not influence the result. Therefore, the resulting value of our suggestion is the same as the output of the Gosper algorithm which implies that Algorithm 2 works correctly. This completes the proof of correctness.

Complexity. We provides an analysis of the computation complexity and space complexity without relying on any asymptotic notation. We first examine computational complexity for Algorithm 2. For readability, the remainder of this section does not use the bar notation, so instead of $\bar{x}, \bar{y}$, etc. we will use x, y, etc.

Computational Complexity. In our algorithm, $f(X) = \frac{a}{X}$ means $z = \begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix}$. Thus, for any encrypted rational number $\bar{x}$, InTake does not require any multiplications because it always becomes $\begin{pmatrix} \frac{a}{\bar{x}} & \frac{a}{\bar{x}} \end{pmatrix}$ after taking ∞ as an input. During running one instance of OutTake for $z = \begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix}$, $z_1 \bmod z_3$ requires to perform a homomorphic multiplications and to execute the GT circuit a times, because we replace modular arithmetic by repeatedly executing the GT circuit. However, z_0 and z_1 in x do not require these two homomorphic operations. Thus, the total number of operations for running OutTake n times is $a \cdot n$ homomorphic multiplications and $a \cdot n$ invocations of the GT circuit.

For each component of z, it requires to perform $\frac{an}{2}$ multiplications and GT executions, respectively. Since the multiplicative depth for GT circuit is $\lceil \log n \rceil + 1$, the necessary multiplicative depth is $\log \frac{an}{2} + \frac{n}{2}(\lceil \log n \rceil + 1)$.

Space Complexity. Let x be an integer and thus it has the only one partial quotient. In our suggested algorithm, the storage requires that a ciphertext corresponding to x and additional four ciphertexts corresponding to $\begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix}$. Let γ denote a bit-size of a ciphertext. Since the total number of ciphertexts are 5, the memory requires that 5γ-bit to perform our proposed algorithm.

5. Gosper Algorithm on Encryptions

Section 4 indicates that a method for dividing an integer by an encrypted integer by restricting an input to an integer. In this section, we treat more common situation by allowing a rational number as an input and we can accomplish our main goal, performing Gosper algorithm over FHE ciphertexts. By the reason, the FHE ciphertexts are rational numbers encoded and encrypted into CF representations. We only examine a special case (the only one partial quotient) of the Gosper algorithm

in Section 4, but we can also apply the same idea for general case (several number of partial quotients) of the Gosper algorithm.

5.1. *Tuning the Gosper algorithm*

A rational number has a couple of partial quotients and thus InTake should be occurred as many as the number of partial quotients. Then, according to Algorithm 1 described in Section 3.1.1, the order of OutTake and InTake is determined by the condition of whether two truncations are the same. However, it is a big burden to check the condition for each iteration, so prior to presenting the specific description of our construction, we slightly modify the Gosper algorithm to reduce the complexity.

In this section, we will provide two theorems to reduce the complexity of the Gosper algorithm by removing the condition of whether two rational numbers have the same integer part, and these might be very helpful for our construction.

Theorem 4. *If* $\lfloor \frac{a}{c} \rfloor = \lfloor \frac{b}{d} \rfloor$, *then* $\lfloor \frac{a+bq}{c+dq} \rfloor = \lfloor \frac{a}{c} \rfloor = \lfloor \frac{b}{d} \rfloor$ *for any* $q \in \mathbb{Z}$.

Proof. Denote $p = \lfloor \frac{a}{c} \rfloor = \lfloor \frac{b}{d} \rfloor$. Then,

$$a = cp + r_1 \text{ for } 0 \leq r_1 < c \tag{11}$$

$$b = dp + r_2 \text{ for } 0 \leq r_2 < d \tag{12}$$

For any $q \in \mathbb{Z}$,

$$a + bq = p(c + dq) + r_1 + r_2 q, \tag{13}$$

and it implies that

$$\frac{a + bq}{c + dq} = p + \frac{r_1 + r_2 q}{c + dq} \tag{14}$$

Because $0 \leq r_1 + r_2 q < c + dq$, $\left\lfloor \frac{a+bq}{c+dq} \right\rfloor = p$. This completes the proof. □

Theorem 5. *For* $\begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix}$ *with* $\lfloor \frac{z_0}{z_2} \rfloor = \lfloor \frac{z_1}{z_3} \rfloor$,

$$\mathit{InTake} \circ \mathit{OutTake}(z) = \mathit{OutTake} \circ \mathit{InTake}(z).$$

Proof. Suppose that $z = \begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix}$, $q := \lfloor \frac{z_0}{z_2} \rfloor = \lfloor \frac{z_1}{z_3} \rfloor$ and the next input partial quotient is $p \in \mathbb{Z}$. First, we look at the left-hand side of the equality.

$$\begin{aligned} \mathsf{InTake} \circ \mathsf{OutTake}(z) &= \mathsf{InTake}\left(\begin{pmatrix} z_2 & z_3 \\ z_0 - z_2 q & z_1 - z_3 q \end{pmatrix} \right) \\ &= \begin{pmatrix} z_3 & z_2 + z_3 p \\ z_1 - z_3 q & (z_0 - z_3 q) + p(z_1 - z_3 q) \end{pmatrix} \end{aligned} \tag{15}$$

By Theorem 4, $q = \lfloor z_1/z_3 \rfloor = \lfloor \frac{z_0+z_1p}{z_2+z_3p} \rfloor$. Thus, the right-hand side of the equality becomes

$$\begin{aligned} \mathsf{OutTake} \circ \mathsf{InTake}(z) &= \mathsf{OutTake}\left(\begin{pmatrix} z_1 & z_0 + z_1p \\ z_3 & z_2 + z_3p \end{pmatrix} \right) \\ &= \begin{pmatrix} z_3 & z_2 + z_3p \\ z_1 - z_3q & (z_0 + z_1p) - q(z_2 + z_3p) \end{pmatrix} \end{aligned} \tag{16}$$

Therefore, $\mathsf{InTake} \circ \mathsf{OutTake}(z) = \mathsf{OutTake} \circ \mathsf{InTake}(z)$, and thus, we may conclude the theorem. □

By Theorem 4 and Theorem 5, Gosper algorithm can be slightly modified to performing InTake for all partial quotients of input including ∞ and then performing OutTake, not alternative. Thus, the condition of whether two rational numbers have the same integer part is no longer required.

5.2. *Our Proposal*

Although Gosper algorithm supports any continued fractions despite an infinite continued fraction, we limit the input as a finite continued fraction and rational numbers. That is, for some $n \in \mathbb{N}$, any rational number is approximated by the first n partial quotients.

Before describing our suggestion, we need to determine the number of invocations of the GT circuit for each partial quotient. For this pupose, we exploit the fact that for a continued fraction $x = [x_0; x_1, x_2, \ldots, x_{n-1}]$, $\max \frac{a+bx}{c+dx} \leq |ad-bc| \cdot \max x$, where $\max x = \max\{x_0, x_1, \ldots, x_{n-1}\}$ [18]. Roughly, this indicates that one can find the upper bound of partial quotients of resulting CF in terms of maximum partial quotient and a coefficient of a fractional linear transform. Thus, the modular reduction can be replaced by iteratively running only GT circuits as many as the number of upper bound partial quotient. In Algorithm 3, we now provide a modified algorithm that enables to compute between rational numbers and encrypted continued fractions.

5.3. *Analysis*

Similar to Section 4.3, we also analyze the correctness and performance of Algorithm 3.

Correctness. Since Algorithm 3 is an extended version of Algorithm 2, the correctness and security are almost the same as Section 4.3; however, we modified the algorithm by exploiting Theorem 4 and Theorem 5. Thus, we explain the reason why our suggestion outputs the same result as Algorithm 1.

For $z = \begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix}$, Algorithm 1 invokes InTake when $\lfloor \frac{z_0}{z_2} \rfloor = \lfloor \frac{z_1}{z_3} \rfloor$ and invokes OutTake when $\lfloor \frac{z_0}{z_2} \rfloor \neq \lfloor \frac{z_1}{z_3} \rfloor$. Eventually, since $z = \begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix}$ becomes $\begin{pmatrix} z_1 & z_1 \\ z_3 & z_3 \end{pmatrix}$ when

InTake takes an input ∞, this condition should be satisfied at some point. Suppose $\lfloor \frac{z_0}{z_2} \rfloor$ is equal to $\lfloor \frac{z_1}{z_3} \rfloor$ after executing InTake k_1 times. Until this point, Algorithm 1 is exactly the same as Algorithm 3 except for their working domain. After executing InTake k_1 times, Algorithm 1 invokes a couple of OutTake and InTake depending on the branch condition. Suppose OutTake and InTake are executed k_2 and k_3 times, respectively. Then, by Theorem 5, we have

$$\begin{aligned}
&\underbrace{\mathsf{OutTake} \circ \cdots \mathsf{OutTake}}_{k_2} \circ \underbrace{\mathsf{InTake} \circ \cdots \mathsf{InTake}}_{k_3} \\
&\qquad = \underbrace{\mathsf{OutTake} \circ \cdots \mathsf{OutTake}}_{k_2-1} \circ \mathsf{InTake} \circ \mathsf{OutTake} \circ \underbrace{\mathsf{InTake} \circ \cdots \mathsf{InTake}}_{k_3-1} \\
&\qquad = \cdots \\
&\qquad = \mathsf{InTake} \circ \underbrace{\mathsf{OutTake} \circ \cdots \mathsf{OutTake}}_{k_2} \circ \underbrace{\mathsf{InTake} \circ \cdots \mathsf{InTake}}_{k_3-1}
\end{aligned}$$

Algorithm 3 Gosper Algorithm on Ciphertext Domains

Input: $\bar{x} = [\bar{x}_0; \bar{x}_1, \ldots, \bar{x}_{n-1}], f(X) = \frac{a+bX}{c+dX}$
Output: $f(\bar{x}) = [\bar{y}_0; \bar{y}_1, \ldots, \bar{y}_{m-1}]$

$\ell \leftarrow$ maximum number of ciphertexts among $\bar{x}_i$
$\begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix} \leftarrow \begin{pmatrix} a & b \\ c & d \end{pmatrix}$
for $i = 0$ to $n-1$ **do**
    $\begin{pmatrix} \bar{z}_0 & \bar{z}_1 \\ \bar{z}_2 & \bar{z}_3 \end{pmatrix} \leftarrow \begin{pmatrix} z_1 & z_0 + z_1\bar{x}_i \\ z_3 & z_2 + z_3\bar{x}_i \end{pmatrix}$
end for
$\begin{pmatrix} \bar{z}_0 & \bar{z}_1 \\ \bar{z}_2 & \bar{z}_3 \end{pmatrix} \leftarrow \begin{pmatrix} \bar{z}_1 & \bar{z}_1 \\ \bar{z}_3 & \bar{z}_3 \end{pmatrix}$
$m \leftarrow$ the number of ciphertext z_1
$k \leftarrow |ad - bc| \cdot 2^{\ell}$
/* OutTake */
for $i = 0$ to $m-1$ **do**
    $y_i \leftarrow 0$
    $rem \leftarrow z_0$
    for $j = 0$ to $k-1$ **do**
        $\bar{t} \leftarrow \mathsf{GT}(tmp, z_2)$
        $\bar{y}_i \leftarrow \bar{y}_i + \bar{t}$
        $rem \leftarrow rem - \bar{t} \cdot z_2$
    end for
    $\begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix} \leftarrow \begin{pmatrix} z_3 & z_3 \\ rem & rem \end{pmatrix}$
end for
return $[\bar{y}_0; \bar{y}_1, \ldots, \bar{y}_{n-1}]$

Continuing in this procedure for every InTake, we can obtain

$$\underbrace{\mathsf{OutTake} \circ \cdots \mathsf{OutTake}}_{k_2} \circ \underbrace{\mathsf{InTake} \circ \cdots \mathsf{InTake}}_{k_3} = \underbrace{\mathsf{InTake} \circ \cdots \mathsf{InTake}}_{k_3} \circ \underbrace{\mathsf{OutTake} \circ \cdots \mathsf{OutTake}}_{k_2}$$

which is exactly the same procedure in Algorithm 3. Therefore, Algorithm 3 outputs the same result as Algorithm 1.

Complexity. We provides an analysis of the computation complexity and space complexity without relying on any asymptotic notation. We first examine computational complexity for Algorithm 2. For readability, the remainder of this section does not use the bar notation, so instead of $\bar{x}, \bar{y}$, etc. we will use x, y, etc.

Computational Complexity. For any z, let us define $z' = \mathsf{InTake}(z) = \begin{pmatrix} z'_0 & z'_1 \\ z'_2 & z'_3 \end{pmatrix}$. Because z'_1 and z'_3 are the output of multiplications and $z_0 + z_1 x$ and $z_2 + z_3 x$ are stored during the next InTake in z'_1 and z'_3 positions, respectively, no multiplications are required when z'_0 and z'_2 are stored. We would like to emphasize that no multiplications are required when an input of InTake is ∞. Hence, if z is the production after performing InTake $n+1$ times including input as ∞, the left side of z, that is, ciphertexts z'_0 and z'_2, should be supported by at least $n-1$ multiplications, and the right side of z, that is, ciphertexts z'_1 and z'_3, should be supported by n multiplications. Since the computational complexity of $\mathsf{OutTake}$ is the same as before, it requires mk homomorphic multiplications and mk invocations of the GT circuit.

Since InTake requires $n+1$ homomorphic multiplications, it requires a multiplicative depth of $\log(n+1)$. $\mathsf{OutTake}$ requires exactly the same cost as that in Algorithm 2; thus it requires a multiplicative depth of $\log \frac{mk}{2} + \frac{mk}{2}(\lceil \log m \rceil + 1)$. Therefore, in total, Algorithm 3 incurs a multiplicative depth of $\log(n+1) + \log mk + \frac{mk}{2}(\lceil \log m \rceil + 1) = m - 1 + \log m(n+1) + mk(\lceil \log m \rceil + 1)$.

Space Complexity. For a rational number x, suppose x has n partial quotients. Through Gosper algorithm, the storage requires that n partial quotients and additional four integers corresponding to $\begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix}$. In the same way, in our suggested algorithm, it requires n ciphertexts corresponding to n partial quotients and an additional four ciphertexts corresponding to $\begin{pmatrix} z_0 & z_1 \\ z_2 & z_3 \end{pmatrix}$. Let γ denote a bit-size of one ciphertext. Since $4+n$ ciphertexts are used, the memory requires that $(4+n)\gamma$-bit to perform Algorithm 3.

6. Conclusion

In this paper, we proposed an arithmetic algorithm that enables a constant divided by an encrypted integer using a special case of the Gosper algorithm. Our algorithm outputs a rational number in form of continued fractions form. Continued fractions are a great tool for representing rational numbers to a sequence of small integers and they are a best approximation of rational numbers. Further, we can extend our suggestion to an arithmetic with a rational number. With our extending algorithm,

addition, subtraction, and multiplication for encrypted rational numbers are possible, and linear fractional transformation also can be evaluated and the complexity is almost the same because every step is the same for any arithmetic.

We leave it as a future work to provide a new version of the current implementation to which we apply SIMD techniques to construct underlying circuits and further optimization techniques.

Acknowledgments

We thank two anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions. Heewon Chung was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.B0717-16-0098). Myungsun Kim was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017-R1D1A1B04035209). All our code, including the programs used for checking the correctness of our suggestion, is publicly available in our project site (http://github.com/heewon-chung/cfhe); but note that any optimization technique is not applied to.

References

[1] J. Bos, K. Lauter and M. Naehrig, Private predictive analysis on encrypted medical data, *Journal of Biomedical Informatics* **50** (2014) 234–243. 2, 5

[2] J. W. Bos, K. E. Lauter, J. Loftus and M. Naehrig, Improved security for a ring-based fully homomorphic encryption scheme, *IMA International Conference on Cryptography and Coding (IMACC) 2013*, ed. M. Stam *LNCS* **8308**, (Springer, 2013), pp. 45–64. 11

[3] Z. Brakerski, C. Gentry and V. Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping, *ITCS*, (2012), pp. 309–325. 3, 11

[4] Z. Brakerski and V. Vaikuntanathan, Fully homomorphic encryption from ring-LWE and security for key dependent messages, *Advances in Cryptology–Crypto*, (2011), pp. 505–524. 11

[5] J. H. Cheon, M. Kim and M. Kim, Search-and-compute on encrypted data, *WAHC*, *LNCS 8976* (2015), pp. 1–18. 12

[6] J. Cooper, Continued fractions with partial quotients bounded in average, *Fibonacci Quarterly* **44**(4) (2006) 297–301. 4

[7] J. Coron, A. Mandal, D. Naccache and M. Tibouchi, Fully homomorphic encryption over the integers with shorter public keys, *Advances in Cryptology - CRYPTO 2011*, ed. P. Rogaway *LNCS* **6841**, (Springer, 2011), pp. 487–504. 11

[8] A. Costache, N. Smart, V. Vivek and A. Waller, Fixed point arithmetic in SHE scheme, *IACR Cryptology ePrint Archive*, **2016** (2016), p. 250. 5, 6

[9] L. Ducas and D. Micciancio, FHEW: Bootstrapping homomorphic encryption in less than a second, *Advances in Cryptology - EUROCRYPT 2015*, eds. E. Oswald and M. Fischlin *LNCS* **9056**, (Springer, 2015), pp. 617–640. 11

[10] T. El Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *Advances in Cryptology-Crypto*, eds. G. R. Blakley and D. Chaum *LNCS 196* (1984), pp. 10–18. 3

[11] C. Gentry, Fully homomorphic encryption using ideal lattices, *STOC*, (2009), pp. 169–178. 11
[12] S. Goldwasser and S. Micali, Probabilistic encryption, *J. Comput. Syst. Sci.* **28**(2) (1984) 270–299. 11
[13] R. Gosper, Continued fraction arithmetic, *HAKMEM Item 101B, MIT Artificial Intelligence Memo 239*, (1972). 4, 8
[14] T. Graepel, K. Lauter and M. Naehrig, ML confidential: Machine learning on encrypted data, *ICISC*, **7839** (2013), pp. 1–21. 5
[15] S. Halevi and V. Shoup, `HElib`-An implementation of homomorphic encryption. 3, 11
[16] G. Hardy and E. Wright, *An Introduction to the Theory of Numbers* (Clarendon Press, 1979). 7
[17] A. Jäschke and F. Armknecht, Accelerating homomorphic computations on rational numbers, *ACNS*, (2016), pp. 405–423. 5
[18] J. Lagarias and J. Shallit, Linear fractional transformations of continued fractions with bounded partial quotients, *Journal de théorie des nombres de Bordeaux* **9**(2) (1997) 267–279. 18
[19] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, *Advances in Cryptology - EUROCRYPT 1999*, ed. J. Stern *LNCS 1592* (1999), pp. 223–238. 3
[20] P. Stambul, Continued fractions with bounded partial quotients, *Proceedings of the American Mathematical Society* **128**(4) (2000) 981–985. 4
[21] M. van Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan, Fully homomorphic encryption over the integers, *Advances in Cryptology–Eurocrypt*, (2010), pp. 24–43. 11