symmetry

MDPI

Article

Homomorphic Comparison for Point Numbers with User-Controllable Precision and Its Applications

Heewon Chung [1], Myungsun Kim [2,*], Ahmad Al Badawi [3] and Khin Mi Mi Aung [3] and Bharadwaj Veeravalli [4]

[1] Department of Mathematics, Hanyang University, Seoul 04763, Korea; tyler.heewonchung@gmail.com
[2] Department of Information Security, College of ICT Convergence, The University of Suwon, Hwaseong 18323, Korea
[3] Institute for Infocomm Research (I2R), A*STAR, Singapore 138634, Singapore; ahmad_al_badawi@i2r.a-star.edu.sg (A.A.B.); mi_mi_aung}@i2r.a-star.edu.sg (K.M.M.A.)
[4] Department of Electrical and Computer Engineering, The National University of Singapore, Singapore 117576, Singapore; elebv@nus.edu.sg
* Correspondence: msunkim@suwon.ac.kr

Received: 10 April 2020; Accepted: 1 May 2020; Published: 8 May 2020

Abstract: This work is mainly interested in ensuring users' privacy in asymmetric computing, such as cloud computing. In particular, because lots of user data are expressed in non-integer data types, privacy-enhanced applications built on fully homomorphic encryption (FHE) must support real-valued comparisons due to the ubiquity of real numbers in real-world applications. However, as FHE schemes operate in specific domains, such as that of congruent integers, most FHE-based solutions focus only on homomorphic comparisons of integers. Attempts to overcome this barrier can be grouped into two classes. Given point numbers in the form of approximate real numbers, one class of solution uses a special-purpose encoding to represent the point numbers, whereas the other class constructs a dedicated FHE scheme to encrypt point numbers directly. The solutions in the former class may provide depth-efficient arithmetic (i.e., logarithmic depth in the size of the data), but not depth-efficient comparisons between FHE-encrypted point numbers. The second class may avoid this problem, but it requires the precision of point numbers to be determined before the FHE setup is run. Thus, the precision of the data cannot be controlled once the setup is complete. Furthermore, because the precision accuracy is closely related to the sizes of the encryption parameters, increasing the precision of point numbers results in increasing the sizes of the FHE parameters, which increases the sizes of the public keys and ciphertexts, incurring more expensive computation and storage. Unfortunately, this problem also occurs in many of the proposals that fall into the first class. In this work, we are interested in depth-efficient comparison over FHE-encrypted point numbers. In particular, we focus on enabling the precision of point numbers to be manipulated after the system parameters of the underlying FHE scheme are determined, and even after the point numbers are encrypted. To this end, we encode point numbers in continued fraction (CF) form. Therefore, our work lies in the first class of solutions, except that our CF-based approach allows depth-efficient homomorphic comparisons (more precisely, the complexity of the comparison is $O(\log \kappa + \log n)$ for a number of partial quotients n and their bit length κ, which is normally small) while allowing users to determine the precision of the encrypted point numbers when running their applications. We develop several useful applications (e.g., sorting) that leverage our CF-based homomorphic comparisons.

Keywords: (fully) homomorphic encryption; continued fraction; approximate arithmetic; sorting

1. Introduction

The success of cloud computing is primarily attributed to the fact that individuals with limited computing power cannot easily manage an exponentially increasing amount of data and knowledge in a cost-effective manner. Let us say that due to privacy concerns, a user Alice, who stores sensitive data at a remote server Bob, wishes to run some algorithms, such as sorting or classification, on the data without revealing any confidential information. In a cloud computing environment, a large variety of useful algorithms rely on secure comparisons. The secure comparison problem in this setting means that for two numbers that Alice is storing, r_0, r_1, Bob outputs r_b such that $r_b \geq r_{1-b}$ without disclosing the numbers, where $b \in \{0, 1\}$.

Following Yao's seminal work on garbled circuits [1], there have been several sensible solutions for cases in which the input numbers are integers. Putting such a restriction on integers seems unavoidable because their underlying cryptographic primitives are designed over integers. Since Fouque et al., in [2], developed an approach to encode rational numbers as integers and to perform computations on the encoded rational numbers, secure noninteger computations have received significant attention. Examples of prominent solutions, to the best of our knowledge, include Catrina and Saxena's solution for fixed-point numbers [3], Aliasgari et al.'s secure floating-point arithmetic [4], and Dimitrov et al.'s secure real-valued computations [5]. Secure comparison has many useful applications; for example, the evaluation of disease risk using clinical data [6] and the private analysis of satellite collisions [7].

One interesting observation is that while Fouque et al.'s solution using Paillier's cryptosystem is non-interactive, all other solutions have been studied under the multiparty computation (MPC) framework, which depends heavily on interactions between the user and the server. However, this property of the MPC can cause conflicts with the user's desire to minimize communications with the server due to limited bandwidth; consequently, existing MPC solutions may not be suitable for the cloud setting. The main advantage of the MPC-based approach can be explained by Archer et al.'s experimental study in [8], which states that the fastest alternative among solutions for secure computing is MPC based on secret sharing. This is one of the main reasons that this work considers fully homomorphic encryption (FHE), which allows anyone to evaluate arbitrary functions (more precisely, circuits representing the functions). Although FHE is notorious for being very computationally intensive, one of our main technical objectives is to devise a way to mitigate performance penalties and apply FHE for homomorphic comparisons over encrypted point numbers.

1.1. Contributions

In this work, our main contributions are twofold:

1. Controllable precision. A key observation by Chung and Kim [9] is that when real numbers are expressed as a continued fraction (CF), partial quotients consisting of their CF representation can be taken as small integers (e.g., $\mathbb{Z}_{1024}$). Thus, encoding point numbers as CFs enables depth-efficient, real-valued computation, even with a small plaintext space. Our new observation is that CF representation allows users and servers to adjust the precision of point-number computation by controlling the number of partial quotients rather than by configuring the FHE parameters. Moreover, it is well known that a CF is the best approximation of a real number among all rational numbers with the same or smaller denominators. In practice, separating precision from the parameter setup may be beneficial because it is more convenient for the user and the design is more modular. We make use of this observation in depth-efficient homomorphic comparisons, including equality and greater-than, to obtain such benefits.
2. Depth-efficient homomorphic comparisons of point numbers. We present three homomorphic comparison protocols—equality, less-than, and greater-than—that efficiently compare the encryption of two CF-encoded point numbers reporting an encrypted result. The key idea is that if two real numbers are the same, their CF representation will be the same; in essence, the equality test for two numbers in CF form is the same as that of the numbers in decimal form.

Our equality test thus requires a multiplicative depth of $O(\log \kappa + \log n)$, where κ is the bit length of the partial quotients and n is the number of partial quotients. Similarly, we devise less-than and greater-than algorithms by comparing two partial quotients at the same position from left to right. These algorithms also have multiplicative depths of $O(\log \kappa + \log n)$. Our results imply that the CF encoding of point numbers is much more suitable for homomorphic comparisons than for homomorphic computations, since comparison algorithms do not require heavy computation. Indeed, CF-based homomorphic computations need to run Gosper's algorithm, which involves modular arithmetic over FHE encryptions. See ([9] §3.1) for details.)

1.2. High-Level Sketch of Our Approach

The primary difference between our techniques and existing solutions lies in the way point numbers are represented. In this work, we write point numbers in CF form and then consider comparison algorithms over their FHE encryptions. Our choice leads to some technical benefits over other encoding techniques that are widely used for supporting homomorphic arithmetic over encrypted real numbers. The primary benefit is reducing the overhead of manipulating the computational precision of numeric values. Because precision-related parameters do not need to be examined when initializing the underlying FHE scheme, the proposed method may be attractive to FHE-based application developers, especially in cases wherein the precision level is not known or varies frequently. In addition to having FHE schemes take small system parameters—particularly a small plaintext space (see Theorem 3)—encryptions of CF-encoded point numbers can be used to execute homomorphic arithmetic without being switched to a different encoding.

We next describe the basic algorithms that enable two FHE-encrypted point numbers to be compared without decryption. As mentioned previously, it is straightforward to compare two point numbers written in CF form. For two $X, Y \in \mathbb{R}$, assume that X (resp., Y) is expressed as a sequence of partial quotients; i.e., $X = [x_0; x_1, \ldots, x_{n-1}]$ (resp., $Y = [y_0; y_1, \ldots, y_{n-1}]$), where for simplicity, all x_i, y_i are non-negative. Then, we only need to check the first pair of partial quotients that are different (i.e., x_i and y_i). For all even-numbered indices $i \in \{0, 2, 4, \ldots\}$, if $x_i < y_i$, then $X < Y$. However, if i is an odd-numbered index and $x_i > y_i$, then $X < Y$. See Theorem 4 for the details of performing comparisons between CF-represented real numbers.

Example 1. *We end this section with a toy example to show that our approach is plausible. Consider a value* $v = 1.2345678901$. *Writing* v *as a CF* $V = [1; 4, 3, 1, 3]$ *has an error of* $\frac{1}{9280} < |v - V| < \frac{1}{5184}$, *but writing it as* $V^* = [1; 4, 3, 1, 3, \mathbf{1}]$ *has only an error of* $\frac{1}{89812152} < |v - V^*| < \frac{1}{89805591}$.

1.3. Applications

Our choice of encoding and basic comparison algorithms have readily usable applications in a variety of areas. We provide a list of some interesting applications for which they are appropriate in scenarios involving sensitive numerical data, particularly real numbers, which are encrypted and stored on a remote untrusted server and require a large number of comparisons without user intervention.

Sorting. Sorting is one of the most frequently used basic data manipulations. Well-known applications of sorting include conducting private auctions [10] and comparing genetic sequences [11], and more generic primitives, such as top-k queries [12] and (weighted) set intersection [13,14]. In [15], Chatterjee et al. studied a way to accelerate an FHE-based sorting algorithm. Unfortunately, their proposed approach requires 235 s to perform a bubble sort on five FHE ciphertexts whose underlying messages are 32-bit integers. In [16], Cetin et al. introduced a new sorting algorithm to optimize a multiplicative depth for efficient homomorphic sorting. The merge sort, which is known as the most efficient sorting algorithm, requires a depth of $O(N \log^2 N)$ for data of size N, but the algorithm of Cetin et al. requires a multiplicative depth of $O(\log N + \log \ell)$, where ℓ is the size of the data.

Machine learning. Millions of people are generating tremendous amounts of data through the increased use of various electronic devices that are normally resource-constrained. Considering the continuously increasing power of modern computers, it is not probable that individuals will be able to keep up with this pace. Thus, one cost-effective approach to handle this asymmetry in computational resources is to rely on cloud computing. A natural solution to prevent privacy breaches of personal data stored in the cloud is to encrypt the data and delegate computations on encrypted data to the cloud.

In this context, homomorphic comparisons over encrypted real numbers are required for basic machine learning (ML) functions, such as classification [17] and the evaluation of decision trees [18]. Then, one can apply these basic ML operations to design a variety of applications, including clinical decision-making aids [19], disease risk estimators [6], and recommendation systems [20].

Private database queries. Other useful applications of homomorphic comparisons include range queries over encrypted databases [21], algorithms for optimization problems in supply chain management [22,23], and privacy-preserving signal processing [24,25]. Note that these examples are a nonexhaustive list of promising applications of secure comparisons using FHE.

1.4. Closely Related Work

This subsection reviews existing work that is closely related to making secure comparisons using homomorphic encryption. The majority of existing solutions focus on securely performing homomorphic computations of point numbers. Despite the wide range of applications, few studies have considered how to securely compare two real numbers.

Given a rational number $q = t/s$, where $\gcd(s,t) = 1$, Fouque et al., in [2], wrote q as $q' = ts^{-1} \pmod N$ for an RSA modulus N where $\gcd(s,N) = 1$, and encrypted q' with Paillier's cryptosystem [26]. After decryption, (t,s) can be recovered from q' by using the Gauss algorithm, which enables a basis to be found in a 2-dimensional lattice. Since Paillier encryption is additively homomorphic, one can perform addition multiple times over encrypted rational numbers; however, a comparison of two encrypted rational numbers cannot be performed.

To our knowledge, there are a few works that attempt to handle point numbers in the FHE framework. In [27], Jäschke and Armknecht proposed a special-purpose encoding algorithm for rational numbers in the FHE context. However, due to the specificity of their proposed encoding, Boolean comparisons on encrypted data of bit length ℓ are not performed efficiently, since $O(\ell)$ multiplicative depth is required rather than a shallower depth of only $O(\log \ell)$. Recently, Costache et al.'s scheme in [28] investigated fixed-point arithmetic in somewhat homomorphic encryption. However, they focused only on basic arithmetic operations. Chung and Kim proposed a CF representation [9], but their protocol does not support homomorphic computations between FHE encryptions (i.e., one input must be given in the clear). In [29,30], an encoding method for fixed-point numbers was presented that was tailored for homomorphic function evaluation and had the advantage of allowing a smaller plaintext coefficient modulus to be chosen. However, even when the coefficients of a polynomial are small, the degree of the polynomial doubles when multiplication is performed in order to preserve correctness. For this reason, when two real numbers are compared with high precision, several multiplications are required and the polynomial degree becomes large, which may not lead to efficient implementation.

In a new line of research, Cheon et al.'s approximate FHE scheme, called homomorphic encryption for arithmetics of approximate numbers (HEAAN) [31], allowed point numbers to be encrypted directly. More recently, a method for depth-efficient comparison of two point numbers encrypted under HEAAN has been suggested by Cheon et al. in [32]. However, for high precision, HEAAN requires a large scale factor Δ (for example, 40–50 bits), since multiplying by a larger scale factor enables higher precision. Another drawback of high-precision settings is the consumption of a higher modulus number Q during the rescaling process. Consequently, HEAAN needs to configure precision-related parameters during its setup algorithm and may suffer from performance degradation by customizing the parameters for high precision. Moreover, existing efficient implementations of HEAAN use residue-number-system

variants of the scheme that set Q as a product of coprime moduli [33]. In these implementations, precision is limited to the size of these primes for maximum accuracy and efficiency. This can limit the choices of the user in setting the computational precision. Table 1 gives a brief summary of what we have reviewed above and the gap we are trying to fill.

Table 1. A comparison of prior works on secure comparison of encrypted operands using FHE.

Approach	Encoding Method	Operations	Logic Function Representation	Error in Comparison	Multiplicative Depth	Precision Control
[27]	Fixed-point	$>, <, =$	Boolean circuit	0	$\mathcal{O}(n)$	×
[32]	Fixed-point	max, min, $=$	Polynomial approximation	$2^{-\alpha}$	$\{\min, \max\} : \Theta(\alpha)$, $\{=\} : \alpha \log \alpha$	×
[34]	Fixed-point	max, min, $=$	Boolean circuit	0	$\mathcal{O}(\log \ell)$	×
Our work	CF	$>, <, =$	Boolean circuit	0	$\mathcal{O}(\log n + \log k)$	◯

Structure of the paper. In Section 2, we introduce definitions, cryptographic tools, and key building blocks for our construction along with a precise description of the system model and security goal. Section 3 provides a brief review of CFs. We then describe our proposed secure comparison algorithms in Section 4. Useful applications and results of our solutions are discussed in Section 5. Finally, Section 6 concludes the work.

2. Definitions and Cryptographic Tools

We begin by introducing some notation and terminology that will be used in the remainder of the paper. We denote that x is encrypted by writing $\bar{x}$. To distinguish integers and real numbers, a small letter indicates an integer (e.g., x), whereas a capital letter indicates a real number (e.g., X). Since real numbers are replaced with approximate point numbers in computing, the term "point numbers" is used throughout the paper. We refer to $\alpha \leftarrow \mathtt{A}(a, b)$ as an algorithm that takes two inputs a and b and outputs α. The base-2 logarithm is denoted as log.

2.1. System Model

We precisely describe the system model on which our possible applications are designed. There are two entities in the system model, i.e., the user and server, and each entity is a protocol participant. A user holding data wishes to store her data on a cloud that is controlled by the server. We focus on the case in which the data are numerical, specifically real numbers, and are meant to be protected from the server.

Our primary goal is to privately compare real numbers with full delegation of the computation to the server. Therefore, when a user submits a request for comparison as a service, she needs to provide a specification of the comparison to the server. The server is then expected to run a set of algorithms to perform the designated computation without interacting with the user. After completing the computation, the server returns an encrypted result to the user. Because the user has a pair of encryption and decryption keys, she can decrypt the results. We assume that the user is honest; however, the server does not need to be honest. A semi-honest server can be assumed. For high privacy, one may require security in the presence of a malicious server. However, due to our performance requirement for a reasonable response time, we only consider a semi-honest server.

2.2. Fully Homomorphic Encryption

An FHE scheme, denoted by $\mathsf{FHE} = (\mathtt{Kg}, \mathtt{En}, \mathtt{De}, \mathtt{Ev})$, is a quadruple of probabilistic polynomial-time (PPT) algorithms, as follows:

Key generation. This 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 $(pk, ek, sk) \leftarrow \mathtt{Kg}(1^\lambda)$ and assume that the public key specifies the plaintext space P and the ciphertext space C.

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

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

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

Since Gentry's first secure FHE scheme was introduced in [35], lots of studies (e.g., [36–38]) have focused on constructing efficient FHE schemes. To tackle their poor performance, Brakerski and Vaikuntanathan [39] newly introduced the notion of leveled FHE, which allows one to evaluate functions of at most a prefixed multiplicative depth rather than arbitrary functions. For a while, Brakerski, Gentry, and Vaikuntanathan [40] developed a leveled FHE scheme over polynomial rings so that their scheme significantly improved performance over the previous schemes. More recently, Cheon et al. suggested a new FHE scheme [31] that supports an approximate computation over ciphertexts. Therefore, there are good solutions for instantiating leveled FHE (e.g., [40,41]). Unless otherwise stated explicitly, hereafter, FHE refers to leveled FHE.

Formally, an FHE scheme is semantically secure if it achieves indistinguishability against chosen-plaintext attackers. We follow a standard definition of semantic security given in [42].

Definition 1 (Semantic Security)**.** *An FHE scheme is semantically secure if for any polynomial-time adversary $\mathcal{A}$,*

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

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

2.3. Security Model

For two given point numbers X_0 and X_1, the functionality that our protocol realizes, outputs the larger number X_b such that $X_b \geq X_{1-b}$ for $b \in \{0,1\}$. To show that our protocol is a secure instantiation of this functionality, we consider a simple form of the standard definition of security in the static, semi-honest model proposed by Goldreich [43]. Generally, all players in the semi-honest model are assumed to follow the instructions prescribed in the protocol. Thus, security in our setting is more straightforward, since we only need to consider the case where the server acts semi-honestly. In other words, no PPT server obtains information about the user's private inputs other than what can be deduced from the result of the protocol. We argue for the protocol's security in the semi-honest model by comparing an ideal-world model, in which a trusted third party (TTP) receives the inputs of the user and outputs the result of the protocol, to a real-world model of the protocol without a TTP.

We use $\mathcal{F}_{\mathtt{op}}$ to denote the real-number comparison functionality and use $\pi_{\mathtt{op}}$ to denote a comparison protocol over real numbers, where the comparison operator $\mathtt{op} \in \{>, <, =\}$. Let x and y be the inputs from the user and server, respectively. We define the user's view of the protocol as $(x, r^c, m_1^c, \ldots, m_k^c)$ and the server's view as $(y, r^s, m_1^s, \ldots, m_{k'}^s)$, where r^c and r^s are the user's and server's respective internal coin tosses and m_i^c and m_i^s are the user's and server's respective i-th messages during the execution of the protocol. For compatibility with existing notation, we will write the server's view as $\mathsf{view}(x, y)$.

Definition 2. *Protocol π_{op} is a* secure *comparison protocol for a comparison operator* $\mathsf{op} \in \{>, <, =\}$ *in the presence of a semi-honest server if a PPT simulator $\mathcal{S}$ exists such that*

$$\{\mathcal{S}(y, \mathcal{F}_{\mathsf{op}}(x, y))\}_{x,y\in\{0,1\}^*} \stackrel{c}{\equiv} \{\mathsf{view}(x, y)\}_{x,y\in\{0,1\}^*},$$

where $\stackrel{c}{\equiv}$ denotes computational indistinguishability by any polynomial-time algorithm.

Note that in our setting, it is not necessary for a server to provide a private input, and thus, y does not have a value.

2.4. Homomorphic Comparison of Integers

As mentioned previously, our solution can be built on top of the homomorphic comparison of two FHE-encrypted integers. Hence, the goal of this subsection is to review homomorphic comparison algorithms of encrypted integers. Recall that $\bar{x}$ denotes an encryption of an integer x.

For simplicity, we follow the convention used in [44] for constructing an integer-valued comparison whose native plaintext space is $\mathbb{Z}_2$. However, it is straightforward to extend this convention to a larger plaintext space such as $\mathbb{Z}_{2^t}$ for a moderately-sized integer t (e.g., 64) while preserving the same multiplicative depths as those of [44] (see [45,46] for the detailed theories).

2.4.1. Equality

The homomorphic equality algorithm for two integers x and y outputs $\bar{1}$ if $x = y$; otherwise, it outputs $\bar{0}$. To compare two integers, bits at the same position are considered. Adding two bits to 1 allows the coincidence of two bits to be checked, since for $a, b \in \mathbb{Z}_2$, $a + b + 1$ outputs 1 in $\mathbb{Z}_2$ if $a = b$; otherwise, it outputs 0 in $\mathbb{Z}_2$. Using this equation, we define an equality test over the integers, denoted by $\mathtt{EQ}_{\mathbb{Z}}$, as follows: for $\bar{x} = \bar{x}_{n-1} \cdots \bar{x}_0$ and $\bar{y} = \bar{y}_{n-1} \cdots \bar{y}_0$, where $x_i, y_i \in \mathbb{Z}_2$,

$$\mathtt{EQ}_{\mathbb{Z}}(\bar{x}, \bar{y}) := \prod_{i=0}^{n-1}(1 + \bar{x}_i + \bar{y}_i). \tag{1}$$

It is clear that the circuit requires $\log n$ multiplicative depths.

2.4.2. Greater-Than and Less-Than

We have two types of inequality test circuits: less-than and greater-than algorithms. The less-than algorithm compares two integers and outputs $\bar{1}$ if $x < y$ and $\bar{0}$ if $x > y$. Similarly, the greater-than algorithm compares two integers and outputs $\bar{1}$ if $x > y$ and $\bar{0}$ if $x < y$. Indeed, the principle of these two algorithms is the same; hence, the greater-than algorithm can easily be obtained from the less-than circuit by adding 1 to the output, and vice versa. If an integer $x = x_{n-1} \cdots x_0$ is less than (resp., greater than) an integer $y = y_{n-1} \cdots y_0$, then there exists $i \in \{0, 1, \ldots, n-1\}$ such that $x_i < y_i$ (resp., $x_i > y_i$) and $x_j = y_j$ for all $i < j$. Since $y_i(x_i + 1)$ (resp., $x_i(y_i + 1)$) outputs $1 \in \mathbb{Z}_2$ if $x_i = 0 < y_i = 1$ (resp., $y_i = 0 < x_i = 1$) and outputs 0 otherwise, we define the less-than circuit, denoted by $\mathtt{LT}_{\mathbb{Z}}$ (resp., greater-than circuit, denoted by $\mathtt{GT}_{\mathbb{Z}}$) over the integers:

$$\mathtt{LT}_{\mathbb{Z}}(\bar{x}, \bar{y}) = \bar{y}_{n-1}(\bar{x}_{n-1} + 1) + \sum_{i=0}^{n-2}\left(\bar{y}_i(\bar{x}_i + 1) \cdot \prod_{j>i}^{n-1}(\bar{x}_i + \bar{y}_i + 1)\right)$$

and

$$\mathtt{GT}_{\mathbb{Z}}(\bar{x}, \bar{y}) = \bar{x}_{n-1}(\bar{y}_{n-1} + 1) + \sum_{i=0}^{n-2}\left(\bar{x}_i(\bar{y}_i + 1) \cdot \prod_{j>i}^{n-1}(\bar{x}_i + \bar{y}_i + 1)\right).$$

We note that a naive implementation of these circuits incurs $O(n^2)$ homomorphic multiplications; however, by rewriting each expression in closed form and applying the single-instruction, multiple-data (SIMD) technique [47,48] to the rewritten expressions, we can have these circuits incur $2(n-1)$ multiplications so that they have a multiplicative depth of only $(1+\log n)$ (see [44]).

3. Rationale of CF Encoding

The goal of this section is to explain why writing numbers in CF form enables us to design depth-efficient homomorphic comparisons such that users can control the precision accuracy independently. For this purpose, we briefly review the basic facts about CFs and their properties. Thus, readers who are familiar with this topic may move on to the next section.

3.1. Definitions

There are some methods that represent point numbers as integers (e.g., decimal expansion and CFs). However, there are reasons why CFs are a more mathematically natural representation than other representations of point numbers. First, the CF representation for a rational number is finite, but the decimal representation for the same number may be infinite. Moreover, every rational number has a unique CF representation, under some restrictions. The successive approximations generated in finding the CF representation of a number by truncating the CF representation are in a certain sense (described below) the best possible option for representing the number. Therefore, the CF representation has been considered to be a great tool in mathematics; however, here we only discuss topics related to our work.

A CF can be obtained by running an iterative process of expressing a number as the sum of its integer part and the reciprocal of the remaining part, and then writing the remaining part as the sum of its integer part and remaining part, and so on. Namely, given a number $X \in \mathbb{R}$ and $r_i > 1$, for all i, we have

$$X = x_0 + \frac{1}{r_0} = x_0 + \frac{1}{x_1 + \frac{1}{r_1}} = x_0 + \frac{1}{x_1 + \frac{1}{x_2 + \frac{1}{r_2}}} = \cdots, \tag{2}$$

and we use $X = [x_0; x_1, x_2, \ldots]$ to denote this. It is clear that x_0 can be any integer, but some definitions and theorems [49] are needed to show that for $i \in \mathbb{N}$, x_i is positive, and that with this restriction, the CF of X is unique.

Definition 3. *For a CF $X = [x_0; x_1, x_2, \ldots]$,*

- *x_i is called a* partial quotient *of X for each i.*
- *A CF X is* finite *if the number of partial quotients of some X is finite.*

All the partial quotients are closely related to each other in the sense that each partial quotient can be written in the previous quotient. Moreover, when a rational number is represented as a CF, its denominator and numerator can be seen as a multivariate polynomial in terms of the partial quotients.

Definition 4 (Convergent). *Let $X = [a_0; a_1, \ldots, a_{n-1}]$, where $X \in \mathbb{Q}$ and $a_i \in \mathbb{Z}$. The* convergents *of X are defined as*

$$C_0 := [a_0], C_1 := [a_0; a_1], \ldots, C_i := [a_0; a_1, \ldots, a_i],$$

where C_i represents a rational number; it is denoted by $C_i = p_i/q_i$, where p_i/q_i is in reduced form.

3.2. Some Properties

We begin by confirming that every rational number has a unique, finite, simple CF representation. Theorem 1 implies that the correspondence between rational numbers and finite CFs $[x_0; x_1, \ldots, x_n]$ with an integer x_0 and positive integer x_i for $i > 0$ and $x_n > 1$ is one-to-one.

Theorem 1 ([49]). *Any rational number can be represented as a finite CF, and the CF representation is unique when the last partial quotient is larger than 1.*

Next, we argue that a CF is the best tool for approximating real numbers as point numbers. Indeed, Theorem 2 indicates that p_i/q_i is the best possible approximation of X among all rational numbers while having the same or smaller denominator.

Theorem 2 ([49]). *Let $X = [x_0; x_1, x_2, \ldots]$ and $\frac{p_i}{q_i} = [x_0; x_1, \ldots, x_i]$. For any rational number $\frac{a}{b}$ with $a \in \mathbb{Z}$ and $b \in \mathbb{N}$, where $1 \leq b \leq q_i$, $\left|\frac{p_i}{q_i} - X\right| \leq \left|\frac{a}{b} - X\right|$, and equality holds if and only if $\frac{a}{b} = \frac{p_i}{q_i}$.*

We now show that each partial quotient is much smaller than the denominator and numerator. Because any real number can be approximated as a rational number, and by Theorem 1, any rational number can be written as a finite CF, we only consider finite CFs henceforth. Nevertheless, we can consider infinite CFs simply as $X = \lim_{i\to\infty} C_i$, where $X \in \mathbb{R}$.

Using the property of convergents, we claim that the size of a partial quotient is much smaller in bit length than those of the denominator and numerator.

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

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

and

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

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 show that the numbers p_i and q_i can be obtained by the recurrence

$$p_i = x_i p_{i-1} + p_{i-2} \tag{3}$$
$$q_i = x_i q_{i-1} + q_{i-2}, \tag{4}$$

with initial conditions $p_0 = x_0, p_{-1} = 1, q_0 = 1$, and $q_{-1} = 0$. We can rewrite Equations (3) and (4) with a matrix as follows:

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

Therefore, all i, p_i, and q_i contain $x_0 x_1 \cdots x_i$, and there is no negative term, which implies that

$$p_i > x_0 x_1 \cdots x_i \text{ and } q_i > x_0 x_1 \cdots x_i.$$

Taking the logarithm with base 2 for both inequalities completes the proof. □

4. Homomorphic Comparison between FHE-Encrypted Point Numbers

For a better understanding, we first present comparison algorithms of two CFs in the clear, and then we describe our homomorphic comparison algorithms over their FHE encryptions. Our constructions rely on the integer-valued comparisons discussed in Section 2.4.

*4.1. Comparisons of Two CF-Encoded Point Numbers **in the Clear***

Consider representation in terms of decimal expansion. One of its greatest advantages is that it makes it easy to compare two point numbers. However, even for a low precision accuracy, a relatively large plaintext space should be configured, so large parameters will be bound to the underlying FHE scheme. This might, in turn, cause the inherent noise to grow very rapidly in homomorphic multiplications. For example, consider a point number r with precision 6; i.e., $r = n_0.n_1 \cdots n_6$. In decimal form, an appropriate plaintext space should be larger than 20 bits simply because r is represented as $(n_0.n_1 \cdots n_6) \cdot 10^6$ and $10^6 \approx 2^{20}$. Furthermore, users cannot control the precision of their data after completing the underlying FHE scheme. As observed in Section 3, introducing CF encoding helps us address these problems.

For simplicity, we write two CF representations as $X = [x_0; x_1, \ldots, x_{n-1}]$ and $Y = [y_0; y_1, \ldots, y_{m-1}]$, where n, m refer to the numbers of partial quotients in each CF representation. To demonstrate our idea effectively, we show a sequence of examples and then prove Theorem 4, on which we base the comparison algorithm between CF-encoded point numbers. Depending on n and m, we have two cases; we discuss each case in turn.

4.1.1. When $n = m$

We divide this case into three subcases in which the partial quotients at specific positions are different from each other.

Case 1. *If $x_0 \geq y_0$, then $X \geq Y$.*

From the definition of a CF (see Equation (2)), we find that the partial quotients x_0 and y_0 dominate all other partial quotients x_i and y_i for $i > 0$. Therefore, X is greater than (resp., less than) Y if x_0 is greater than (resp., less than) y_0.

Case 2. *If $x_0 = y_0$ and $x_1 \geq y_1$, then $X \leq Y$.*

Suppose that $x_0 = y_0$. Then, X and Y from Equation (2) can be written as follows:

$$\frac{1}{X - x_0} = x_1 + \frac{1}{x_2 + \frac{1}{x_3 + \cdots}}, \quad \text{and} \quad \frac{1}{Y - y_0} = y_1 + \frac{1}{y_2 + \frac{1}{y_3 + \cdots}}.$$

We then rewrite these as $\frac{1}{X-x_0} = [x_1; x_2, \ldots, x_{n-1}]$ and $\frac{1}{Y-y_0} = [y_1; y_2, \ldots, y_{n-1}]$. We can see that the result of the comparison between $\frac{1}{X-x_0}$ and $\frac{1}{Y-y_0}$ fully depends on x_1 and y_1. The only difference from Case 1 is that if x_1 is greater than (resp., less than) y_1, then X is less than (resp., greater than) Y, because $\frac{1}{X-x_0} > \frac{1}{Y-y_0}$ (resp., $\frac{1}{X-x_0} < \frac{1}{Y-y_0}$), where $x_0 = y_0$.

Case 3. *If $x_0 = y_0$, $x_1 = y_1$, and $x_2 \geq y_2$, then $X \geq Y$.*

Suppose that $x_0 = y_0$ and $x_1 = y_1$. In the same way as above, we have that $\frac{1}{X-x_0} - x_1 = [0; x_2, \ldots, x_{n-1}]$ and $\frac{1}{Y-y_0} - y_1 = [0; y_2, \ldots, y_{n-1}]$. This time, x_2 and y_2 determine the result of comparing $\frac{1}{X-x_0} - x_1$ and $\frac{1}{Y-y_0} - y_1$. If x_2 is greater than (resp., less than) y_2, then $\frac{1}{X-x_0} - x_1$ is less than (resp., greater than) $\frac{1}{Y-y_0} - y_1$, which implies that X is greater than (resp. less than) Y.

We can continue this process to an arbitrary $n(=m)$, but it is not difficult to infer a relationship between the sizes of the point numbers and the indices of their partial quotients.

4.1.2. When $n \neq m$

This case is much simpler than the case above, because the comparison result is totally determined by n and m. Without loss of generality, we assume that $n < m$. For $i \in \{0, 1, \ldots, n-1\}$, it is trivial to compare two CFs by applying the same rule as above. However, it is unclear how to compare two CFs, because $x_i = y_i$ for all $i \in \{0, 1, \ldots, n-1\}$, but there is no partial quotient of X corresponding to y_i for some $i \geq n$. To tackle this problem, we make two point numbers in CF form have the same number of partial quotients using the simple trick below.

Our key idea is that because $0 = \frac{1}{\infty}$, adding ∞ to the end of partial quotients on the X side has no side effects. Specifically, since

$$\frac{1}{x_{n-1}} = \frac{1}{x_{n-1} + \frac{1}{\infty}},$$

we have that $X = [x_0; x_1, \ldots, x_{n-1}] = [x_0; x_1, \ldots, x_{n-1}, \infty] = [x_0; x_1, \ldots, x_{n-1}, \overbrace{\infty, \infty, \ldots, \infty}^{m-n}]$. Consequently, we can always write two point numbers as two CFs with the same numbers of partial quotients. Thus, it is possible to compare two real numbers in terms of CFs when they have different numbers of partial quotients. Our argument can be formally stated by Theorem 4. Because the proof of the theorem is clear by mathematical induction on k, we omit the details of the proof.

Theorem 4. *Let $X = [x_0; x_1, \ldots, x_{n-1}]$, $Y = [y_0; y_1, \ldots, y_{m-1}]$ and $n \leq m$. Let k be the smallest index for which $x_k \neq y_k$. Then,*

$$X < Y \text{ if } (-1)^k (x_k - y_k) < 0,$$

and $X > Y$ otherwise. If there is no k that implies that $x_i = y_i$ for all $i < n$ and $n < m$, then $X < Y$ if n is odd and $X > Y$ if n is even. If there is no such k and $n = m$, then clearly $X = Y$.

This section ends with Algorithm 1, which provides a concrete algorithm for comparing two CFs in the clear.

Algorithm 1 Comparing two continued fractions in the clear

`Input.` $X = [x_0; x_1, \ldots, x_{n-1}]$ and $Y = [y_0; y_1, \ldots, y_{m-1}]$
`Output.` The result of the comparison of X and Y

Find k such that k is the smallest index for which $x_k \neq y_k$
if $\exists k$ **then**
    Set $c \leftarrow (-1)^k (x_k - y_k)$
    if $c < 0$ **then**
        return $X < Y$
    else
        return $X > Y$
else
    if $n = m$ **then**
        return $X = Y$
    else if $n < m$ **then**
        if n is odd **then**
            return $X < Y$
        else ▷ n is even
            return $X > Y$
    else
        if m is odd **then**
            return $X < Y$
        else ▷ m is even
            return $X > Y$

4.2. Our Homomorphic Comparisons over FHE Encryptions

We are ready to provide our depth-efficient homomorphic comparison algorithms over FHE encryptions. To this end, we revise Algorithm 1 so that it operates correctly on FHE ciphertexts, whose original numbers have been written in CF form. We note that an FHE scheme with a larger plaintext space than $\mathbb{Z}_2$ can be set up without additional costs in terms of multiplicative depth.

4.2.1. Equality Tests

We define an equality circuit, denoted by $\mathtt{EQ}_{\mathbb{R}}$, which takes as input two encrypted CFs, as follows:

$$\mathtt{EQ}_{\mathbb{R}}(\bar{X}, \bar{Y}) := \prod_{i=0}^{n-1} \mathtt{EQ}_{\mathbb{Z}}(\bar{x}_i, \bar{y}_i). \tag{5}$$

The correctness of the algorithm is formally proven by Lemma 1.

Lemma 1. *Let $\bar{X}$ and $\bar{Y}$ be encrypted CFs for real numbers X and Y, respectively. The equality circuit $\mathtt{EQ}_{\mathbb{R}}$ in Equation (5) correctly computes an encryption of the equality test result between the two encrypted real numbers as CFs.*

Proof. Let $\bar{X} = [\bar{x}_0; \bar{x}_1, \ldots, \bar{x}_{n-1}]$ and $\bar{Y} = [\bar{y}_0; \bar{y}_1, \ldots, \bar{y}_{m-1}]$ be two encrypted CFs. Only if $n = m$ and $x_i = y_i$ for all $0 \leq i \leq n-1$ are the two CFs equal. We can easily check the first condition by the number of their ciphertexts, and the second condition by comparing two encrypted partial quotients at the same position; i.e., by invoking $\mathtt{EQ}_{\mathbb{Z}}(\bar{x}_i, \bar{y}_i)$. If $x_i = y_i$, then $\mathtt{EQ}_{\mathbb{Z}}(\bar{x}_i, \bar{y}_i)$ outputs $\bar{1}$; otherwise, it outputs $\bar{0}$. Hence, if $x_i = y_i$ for all $0 \leq i \leq n-1$, then all outputs of $\mathtt{EQ}_{\mathbb{Z}}$ are $\bar{1}$ so that the product of all of these outputs is $\bar{1}$. However, if there is at least one pair of different partial quotients, then because the output of the corresponding $\mathtt{EQ}_{\mathbb{Z}}$ is $\bar{0}$, the product of the outputs is $\bar{0}$. □

We then construct a comparison protocol, denoted by $\Pi_{=}$, by extending the equality algorithm. The protocol $\Pi_{=}$ consists of the following two rounds:

- (The first round.) A user sends two FHE encryptions $(\bar{X}, \bar{Y})$ to a server.
- (The second round.) The server responds to the user by sending the result of homomorphically evaluating $\mathtt{EQ}_{\mathbb{R}}$ at the two FHE encryptions $\bar{X}$ and $\bar{Y}$.

4.2.2. Greater-Than and Less-Than Tests

Using integer-based circuits, we define two comparison circuits, denoted by $\mathtt{GT}_{\mathbb{R}}$ and $\mathtt{LT}_{\mathbb{R}}$, taking as input two encrypted CFs, as follows:

$$\begin{aligned}\mathtt{GT}_{\mathbb{R}}(\bar{X}, \bar{Y}) := \mathtt{GT}_{\mathbb{Z}}(\bar{x}_{n-1}, \bar{y}_{n-1}) + \sum_{i=0}^{\frac{n-2}{2}} \mathtt{GT}_{\mathbb{Z}}(\bar{x}_{2i}, \bar{y}_{2i}) \cdot \prod_{j \geq 2i}^{n-1} \mathtt{EQ}_{\mathbb{Z}}(\bar{x}_j, \bar{y}_j) + \\ \sum_{i=0}^{\frac{n-2}{2}} \mathtt{LT}_{\mathbb{Z}}(\bar{x}_{2i+1}, \bar{y}_{2i+1}) \cdot \prod_{j \geq 2i+1}^{n-1} \mathtt{EQ}_{\mathbb{Z}}(\bar{x}_j, \bar{y}_j),\end{aligned}$$

and

$$\mathtt{LT}_{\mathbb{R}}(\bar{X},\bar{Y}) := \mathtt{LT}_{\mathbb{Z}}(\bar{x}_{n-1},\bar{y}_{n-1}) + \sum_{i=0}^{\frac{n-2}{2}} \mathtt{LT}_{\mathbb{Z}}(\bar{x}_{2i},\bar{y}_{2i}) \cdot \prod_{j\geq 2i}^{n-1} \mathtt{EQ}_{\mathbb{Z}}(\bar{x}_j,\bar{y}_j) + \sum_{i=0}^{\frac{n-2}{2}} \mathtt{GT}_{\mathbb{Z}}(\bar{x}_{2i+1},\bar{y}_{2i+1}) \cdot \prod_{j\geq 2i+1}^{n-1} \mathtt{EQ}_{\mathbb{Z}}(\bar{x}_j,\bar{y}_j).$$

Lemma 2. *Let $\bar{X}$ and $\bar{Y}$ be encrypted CFs for real numbers X and Y, respectively. The comparison circuit $\mathtt{GT}_{\mathbb{R}}$ (resp., $\mathtt{LT}_{\mathbb{R}}$) correctly computes an encryption of the greater-than (resp., less-than) comparison result between two encrypted real numbers in CF form.*

Proof. Let $x, Y \in \mathbb{R}$, and let $\bar{X}$ and $\bar{Y}$ be their respective encrypted CFs. The comparison test between CFs proceeds in a left-to-right direction, like comparing two integers. For two integers given in binary, when the first difference occurs at an index i in their binary expansions, the integer of the i-th bit of being 1 is greater than the other integer. Additionally, as proven in Theorem 4, when the first difference occurs at the index i in the two CF expansions, the comparison between the two CFs needs to consider the index i itself along with the i-th partial quotients. Specifically, if the index i is odd, then the direction of the inequality of the two i-th partial quotients is the same as the direction of the inequality of the two real numbers; but if the index i is even, then the direction of the inequality of the two i-th partial quotients is the opposite of that of the two real numbers.

On the other hand, we need to carefully deal with the case that $n \neq m$, when comparing the two encrypted CFs. As many dummy partial quotients ∞ as the difference between n and m are added, such that both CFs have the same number of partial quotients. For efficiency reasons, we do not encrypt the dummy partial quotients. Instead we just have to check if the index at which the first partial quotient appears is even or odd, as ∞ is larger than any integer. □

Similarly, we can design a comparison protocol for the greater-than (resp., less-than) test, denoted by $\Pi_{>}$ (resp., $\Pi_{<}$), as follows:

- A user sends two FHE encryptions $(\bar{X},\bar{Y})$ to a server.
- The server responds to the user by sending the result of homomorphically evaluating $\mathtt{GT}_{\mathbb{R}}$ (reps. $\mathtt{LT}_{\mathbb{R}}$) for the two FHE encryptions $\bar{X}$ and $\bar{Y}$.

4.3. Efficiency

We evaluate the performance of the algorithms by finding the multiplicative depth. Note that for two ℓ-bit integers, the best-known equality test algorithm has a multiplicative depth of $\lceil \log \ell \rceil$ and the best-known comparison test algorithm has a multiplicative depth of $\lceil \log \ell \rceil + 1$.

Given a real number X, we write it as $X = [x_0; x_1, \ldots, x_{n-1}]$ in CF form but as $X = x \cdot 10^k$ in decimal form with precision k. Graepel et al. take a systematic approach to encoding real numbers as elements of polynomial rings [50]. However, in essence, their approach is the same as decimal representation, because the polynomial of a real number X with precision k has a degree of $\lceil \log x \cdot 10^k \rceil$. As studied in [9], our algorithms use a plaintext space of $\kappa = \lceil \ell/n \rceil$ bits on average, where we set ℓ to $\ell = \lceil \log X \rceil = \lceil \log x \cdot 10^k \rceil$. However, as a trade-off, it seems that the number of ciphertexts must be equal to the number of partial quotients; i.e., there are n ciphertexts for κ-bit messages. Fortunately, we can fix this problem by using FHE schemes (e.g., [31,40]) that support ciphertext packing [47,48]. The ciphertext packing technique allows us to evaluate a function homomorphically, in parallel, on n blocks of encrypted data. Essentially, it works by packing multiple plaintexts into one ciphertext. More specifically, when we use a variety of parameters with a fixed plaintext space $p = 2$, key-switching digit parameter $c = 3$ and cyclotomic polynomial $\Phi_{19811}(z)$, yielding a plaintext space of $\mathbb{Z}_2[z]/\langle \Phi_{19811}(z) \rangle \cong \prod_{i=1}^{360} \mathbb{F}_{2^{50}}$, we are able to pack 360 48-bit partial quotients into one FHE ciphertext.

Note that packing allows SIMD-like computation on the vectors of encrypted data simultaneously without additional cost.

Multiplying each result of the comparison between encrypted partial quotients requires that the equality test should have a multiplicative depth of $\lceil \log n \rceil$. Similarly, the comparison test icurs an additional $2\lceil \log n \rceil$ homomorphic multiplications and $\lceil \log n \rceil$ homomorphic additions. Table 2 reports on measurements with respect to the plaintext space, the number of ciphertexts, and the multiplicative depth.

Table 2. Summary of efficiency.

Algorithms	Measures	Values
$\mathtt{EQ}_{\mathbb{R}}$	Plaintext space	$\mathbb{Z}_2^{\kappa}$
	# of Ciphertexts	n
	Multiplicative depth	$\lceil \log \kappa \rceil + \lceil \log n \rceil$
$\mathtt{GT}_{\mathbb{R}}/\mathtt{LT}_{\mathbb{R}}$	Plaintext space	$\mathbb{Z}_2^{\kappa}$
	# of Ciphertexts	n
	Multiplicative depth	$\lceil \log \kappa \rceil + \lceil \log n \rceil + 2$

Remark 1. *In this work, we do not consider arithmetic on encrypted point numbers. However, a beneficial aspect of homomorphic encryption compared to other cryptosystems is that anyone can add and multiply the underlying messages that are being encrypted. Because homomorphic evaluation results can also be represented as CFs of small terms, our argument holds for arithmetic operations on encrypted real numbers. Additionally, for decimal or polynomial representation cases, their evaluation results should not overflow beyond the underlying message space. Therefore, an FHE scheme is not easy to set up, even for a small number of homomorphic multiplications.*

Experimental setup. Our experiments use a variant of the BGV FHE scheme [40] to implement the equality and comparison circuits on real numbers. A complete C++ implementation of the underlying FHE scheme is provided by Halevi and Shoup based on the number theoretic library NTL [51] (version 10.3.0), named HElib [52]. We also utilize Open Multi-Processing (OpenMP) [53] to take advantage of available processing cores to run code concurrently and efficiently. While HElib supports bootstrapping, we use a leveled variant of the BGV-type scheme that supports homomorphic evaluations up to a predefined level.

Microbechmarks. In the following, we demonstrates simple experiments for comparing the run times in the same setting. For this purpose, we select several random CFs by varying the number of partial quotients and express these numbers in decimal form with various degrees of precision. Then, we encrypt each encoding to generate two ciphertexts for these numbers. During setting up an instance of the FHE scheme, we chose the minimum FHE parameters that may guarantee correct evaluations. In Table 3, we compare the average run times with each encoding with various degrees of precision.

Table 3. Comparisons of run times between encodings.

n	ℓ	k	Equality Test		Comparison Test	
			Ours	**Decimal**	**Ours**	**Decimal**
3	3	5	0.361	0.686	2.108	0.884
		10		0.777		2.050
		20		1.311		5.119
	5	5	0.679	0.652	2.223	0.888
		10		0.707		2.095
		20		1.354		5.222
	7	5	0.717	0.577	2.227	0.884
		10		0.719		2.116
		20		1.251		5.289
5	3	5	0.440	0.671	2.142	0.920
		10		0.720		2.163
		20		1.310		5.173
	5	5	0.759	0.657	4.155	0.942
		10		0.748		2.075
		20		1.330		5.268
	7	5	0.768	0.634	4.248	0.946
		10		0.748		2.076
		20		1.298		5.268
7	3	5	0.488	0.631	2.389	0.867
		10		0.704		2.109
		20		1.330		5.266
	5	5	0.854	0.639	4.366	0.934
		10		0.759		2.038
		20		1.323		5.254
	7	5	0.927	0.688	4.554	0.836
		10		0.811		2.163
		20		1.334		5.430

4.4. Security

The underlying FHE scheme we use is semantically secure; that is, no polynomial-time servers can distinguish between $\bar{m}_0$ and $\bar{m}_1$ for arbitrary messages $m_0, m_1 \in \mathsf{P}$. Because all comparisons are performed by the server without decryption, it is obvious that no semi-honest server can learn any information about the encrypted numbers. For completeness, we begin by arguing for the correctness of our protocols. The completeness property is trivial from Lemmas 1 and 2.

Theorem 5 (Correctness)**.** *Let $\bar{X}$ and $\bar{Y}$ be encrypted CFs for real numbers X and Y, respectively. The comparison protocols Π_{op} correctly compute an encryption of the comparison result between the two encrypted point numbers in CF form, where $\mathsf{op} \in \{>, <, =\}$.*

Next, we will check whether our protocol securely performs real-number comparison.

Theorem 6. *Assume that the FHE scheme used in the comparison protocols is semantically secure. Then, in our comparison protocols Π_{op} of point numbers, a polynomial-time semi-honest server can learn nothing about the information other than what could be revealed in the ideal model using the same private inputs with a TTP.*

Proof. Because of the semantic security of the underlying FHE scheme, no PPT semi-honest servers can distinguish $\bar{m}_0$ and $\bar{m}_1$ for two same-length plaintexts m_0, m_1. Thus, nothing is revealed to the server before decryption by the user, since all user inputs are encrypted by the FHE scheme but the

server has no input. Note that a description of the comparison circuit is known to the server before running the comparison protocol.

Specifically, for two inputs $\bar{X}$ and $\bar{Y}$, all partial quotients $\bar{x}_i$ and $\bar{y}_i$ are encrypted by the semantically secure FHE scheme and provided to the server. Thus, the server cannot learn information about $\bar{X}$ and $\bar{Y}$ except for the numbers of partial quotients (i.e., n and m). However, these numbers should also be known to an ideal adversary when the decimal precision for representing point numbers is fixed. Furthermore, while computing the intermediate values, the server evaluates the comparison circuit using only encrypted summands and learns nothing about the final result of the circuit. Consequently, no information of the honest user concerning X and Y is revealed to the semi-honest server other than the decimal precision, as implied by the encrypted comparison result of the protocol at the end of execution. □

There are a variety of privacy-enhanced applications that can employ our FHE-based point number comparison solution. We present two such useful applications: sorting a list of FHE-encrypted point numbers and processing retrieval queries on FHE-encrypted databases.

5. Applications

5.1. Sorting on Encrypted Databases

Sorting is one of the most fundamental operations undertaken in computers, and it has been studied for a long time. Furthermore, with the advent of cloud services and cloud computing, sorting on encrypted databases has also been getting a lot of attention. Order-preserving encryption [54] and order-revealing encryption [55–57] are two appealing solutions for sorting encrypted data for efficiency on the server side. However, these encryption schemes can leak information on the relative distance between the underlying messages. In particular, they can leak more information in specific settings; for example, a set of non-uniform data [58].

On the other hand, because FHE schemes leak no information about the underlying messages, FHE-based sorting does not leak any information to a semi-honest server beyond that revealed by its output (e.g., the size of the list). Several solutions have been suggested for sorting FHE-encrypted lists (see, e.g., [15,16]). However, to the best of our knowledge, there have been known no solutions working for a list of point numbers.

The construction. Two fundamental operations of classical sorting algorithms are comparisons and swaps. Thus, the main goal of our construction is to use our comparison algorithms $\mathtt{GT}_{\mathbb{R}}$ and $\mathtt{LT}_{\mathbb{R}}$ to construct a swapping algorithm for two encrypted values. A swap operation takes a sequence of point numbers and outputs an ordered sequence. Without loss of generality, we consider only ascending order, but descending order can be easily obtained with small changes. For two encrypted point numbers $\bar{X}$ and $\bar{Y}$, swapping in ascending order is defined as

$$\mathtt{Swap}(\bar{X},\bar{Y}) := [\mathtt{LT}_{\mathbb{R}}(\bar{X},\bar{Y}) \cdot \bar{X} + \mathtt{LT}_{\mathbb{R}}(\bar{Y},\bar{X}) \cdot \bar{Y}, \mathtt{GT}_{\mathbb{R}}(\bar{X},\bar{Y}) \cdot \bar{X} + \mathtt{GT}_{\mathbb{R}}(\bar{Y},\bar{X}) \cdot \bar{Y}],$$

where by $[\bar{X},\bar{Y}]$ we mean that $\bar{X}$ and $\bar{Y}$ are stored in order.

Efficiency. We focus on Gizem et al.'s two sorting algorithms, called direct sort and greedy sort [16]. The reason for considering the scheme is that different from existing algorithms with $O(N \log^2 N)$ multiplicative depth, their scheme incurs merely $O(\log N + \log \ell)$ multiplicative depth for the size of the input list N and the bit size of the elements ℓ. More specifically, to sort N values, $a_0, \ldots, a_{N-1}$, the protocol first builds an $N \times N$ matrix whose components (i, j) are $\mathtt{GT}_{\mathbb{Z}}(a_i, a_j)$ for $i, j \in \{0, 1, N-1\}$. Because the sum of the i-th row of the matrix means the number of elements greater than a_i, it can sort all values according to this sum. Moreover, combining their algorithms with merge sort can provide better performance.

We can apply their technique to our scenario using the comparison algorithms for point numbers. Specifically, we construct a comparison matrix whose components (i, j) are $\mathtt{GT}_{\mathbb{R}}(a_i, a_j)$

for $0 \leq i, j < N$ rather than $\mathtt{GT}_{\mathbb{Z}}(a_i, a_j)$, and we can sort an encrypted list of point numbers with the same multiplicative depth.

5.2. Database Queries on Encrypted Databases

Informally, by a retrieval database query or database query for short we mean the process of retrieving the data that satisfy a set of predicates from a database. Database queries are one of the the most important and basic operations that make databases useful. However, if users' data are sensitive (e.g., medical or financial information), part (occasionally all) of the database involving these sensitive attributes should be protected from unauthorized access. For this reason, lots of studies have paid great attention to providing privacy-preserving query services for outsourced data, called private database queries (PDQs).

PDQ solutions enable users to learn desired information from encrypted databases without revealing information to the server. Searchable encryption [59] can be used to retrieve information as a result of queries; however, it can leak information about plaintext messages. In particular, it does not allow the evaluation of encrypted functions.

In this work, we take into account two simple types of PDQs, called retrieval queries and aggregate queries. For example, consider two queries over a medical database:

- What are the names of all patients whose preprandial plasma glucose levels were above 11.3450098875 mmol/L?
- What is the average age of female patients whose postprandial plasma glucose is below 13.10134111097 mmol/L?

To respond to these queries, database systems have to identify tuples satisfying a given search condition and sometimes perform computations on the set of results. Because of this need, FHE-based PDQ protocols (e.g., [60,61]) have been developed to support various queries.

Search queries. Search queries (a.k.a., retrieval queries) consist of a list of attributes, their owner table names, and a search condition. One way to offer reasonable performance is to encrypt only the private constant values in the search condition statement. Then, on receiving such a such query, the cloud database server can identify which attributes in a given table are compared to the encryptions of constants. In principle, search queries can be grouped into conjunctive queries and disjunctive queries. Conjunctive queries require that all predicates in a search condition should be satisfied, whereas disjunctive queries require that at least one predicate in a search condition only have to be satisfied.

Suppose that a database R has a schema $R(A_1, A_2, \ldots, A_\tau)$, where R is a relation name and each A_i is an attribute name, and that it has N encrypted tuples. Note that all tuples in the relation do not need to be encrypted; for performance reasons, some of sensitive attribute values have to be encrypted. For simplicity, we use $\mathtt{Cmp}(A_i[j], \bar{X}_i)$ to denote the i-th predicate to compare an encrypted constant $\bar{X}_i$ to an encrypted attribute value $\bar{Y}_{i,j}$ of the attribute A_i in the j-th tuple where $j \in \{1, 2, \ldots, N\}$. Recall that the comparison predicate $\mathtt{Cmp}$ performs one of comparison operations in $\{<, =, >\}$. Then, we can describe search queries over the encrypted database R as

$$\mathtt{Conj}(\bar{X}_{i_1}, \ldots, \bar{X}_{i_\eta})[j] := \mathtt{Cmp}(A_{i_1}[j], \bar{X}_{i_1}) \wedge \cdots \wedge \mathtt{Cmp}(A_{i_\eta}[j], \bar{X}_{i_\eta})$$

and

$$\mathtt{Disj}(\bar{X}_{i_1}, \ldots, \bar{X}_{i_\eta})[j] := \mathtt{Cmp}(A_{i_1}[j], \bar{X}_{i_1}) \vee \cdots \vee \mathtt{Cmp}(A_{i_\eta}[j], \bar{X}_{i_\eta}).$$

If A_k is a target attribute, at the end of the protocol the server produces as a resulting set $\{\bar{Y}_{k,j} \cdot \mathtt{Conj}(\bar{X}_{i_1}, \ldots, \bar{X}_{i_\eta})[j]\}_{1 \leq j \leq N}$. In the same way, we can figure out the output of disjunctive queries.

Aggregate queries. Aggregate functions return a single resulting value based on a group of data that is formed by applying search queries; Examples include `sum`, `avg`, and `count`. For the purpose of

arithmetic with CFs, we can use Gosper's arithmetic algorithm in [62] between CFs. The algorithm allows arithmetic operations to be performed not only between two CFs but also between a CF and a rational number. More concretely, the algorithm receives two numbers $X, Y \in \mathbb{R}$ in CF form and outputs $F(X,Y) := (\alpha + \beta X + cY + \delta XY)/(\epsilon + fX + \zeta Y + \eta XY)$, where $\alpha, \beta, c, \delta, \epsilon, f, \zeta, \eta \in \mathbb{Z}$. For example, setting $\beta = 1, c = -1, \epsilon = 1$ indicates $F(X,Y) = X - Y$, setting $\delta = -1, \epsilon = 1$ indicates $F(X,Y) = -X \cdot Y$, and setting $\alpha = 2, \beta = 1, \zeta = 3$ indicates $F(X,Y) = (2 + X)/3Y$. This implies that $F(X,Y)$ can express every primitive arithmetic operation, Therefore, we can evaluate every primitive arithmetic operation along with a two-variable linear fractional transformation.

In [9], the authors first investigate the possibility of employing FHE with the Gosper algorithm to allow anyone to evaluate two encrypted CFs. In principle, with the Gosper algorithm, we can evaluate arbitrary polynomials utilizing the same approach as discussed above. However, because this approach becomes fairly impractical when the polynomial has the degree of greater than 2, the algorithm is not suitable for aggregate queries for performance reasons. More specifically, in the Gosper algorithm, the process of expressing rational numbers $Z = F(X,Y)$ using the partial quotients of X and Y requires quite high computational costs. Indeed, each reconstruct of the numerator and denominator for each partial quotient requires on homomorphic multiplication. For this reason, separately stroing both numerators and denominators with their CFs enables to reduce the computational overhead. Then, executing arithmetic operations with numerators and denominators becomes much more efficient than directly applying the Gosper algorithm, as this can significantly relieve the cost of reconstructing rational numbers.

6. Summary of Results

In this work, we addressed the goal of allowing users to manipulate the computational precision of encrypted numerical data, namely, point numbers, that are to be stored at a remote server. In most existing schemes, the precision must be configured when setting up the underlying FHE scheme, so users have no control over the precision accuracy of the FHE-encrypted point numbers. To solve this problem, we encoded point numbers as continued fractions and suggested a set of efficient homomorphic comparison algorithms over their FHE encryptions. We then showed that depth-efficient homomorphic comparison algorithms can be obtained by only using efficient homomorphic comparisons over FHE-encrypted integers. In particular, we argued that our comparison algorithms require only logarithmic multiplicative depth. Our technique may incur the costs of the overhead to store every partial quotient of the point numbers; however, using an FHE scheme that allows for the SIMD technique, we can reduce this overhead. Thus, our solutions have a wide range of applications in cloud-based tasks.

Author Contributions: Conceptualization, H.C. and M.K.; writing—original draft preparation, H.C. and M.K.; writing—review and editing, A.A.B.; project administration, K.M.M.A. and B.V. All authors have read and agreed to the published version of the manuscript.

Funding: Ahmad Al Badawi and Khin Mi Mi Aung are supported by A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Programmtic Programme (Award A19E3b0099). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the A*STAR. Myungsun Kim was supported by the Institute for Information and Communication Technology Promotion (IITP) grant funded by the Korean government (MSIT) (2018-0-00251, Privacy-Preserving and Vulnerability Analysis for Smart Contract).

Acknowledgments: This work was partially conducted while Heewon Chung was doing internships at A*STAR. We thank the anonymous reviewers for providing us valuable feedback for improving our paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yao, A.C. How to generate and exchange secrets (extended abstract). In Proceedings of the 27th Annual Symposium on Foundations of Computer Science (SFCS 1986), Toronto, ON, Canada, 27–29 October 1986; pp. 162–167.
2. Fouque, P.; Stern, J.; Wackers, J. CryptoComputing with rationals. In *Financial Cryptography, Proceedings of the 6th International Conference, Bermuda, March, 11–14 March 2002*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 136–146.
3. Catrina, O.; Saxena, A. Secure computation with fixed-point numbers. In *Financial Cryptography and Data Security, Proceedings of the 14th International Conference, Tenerife, Canary Islands, 25–28 January 2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 35–50.
4. Aliasgari, M.; Blanton, M.; Zhang, Y.; Steele, A. Secure computation on floating point numbers. In Proceedings of the NDSS Symposium 2013, San Diego, CA, USA, 24–27 February 2013.
5. Dimitrov, V.; Kerik, L.; Krips, T.; Randmets, J.; Willemson, J. Alternative implementations of secure real numbers. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 553–564.
6. Ayday, E.; Raisaro, J.L.; McLaren, P.; Fellay, J.; Hubaux, J. *Privacy-Preserving Computation of Disease Risk by Using Genomic, Clinical, and Environmental Data*; Usenix HealthTech: Washington, DC, USA, 2013.
7. Kamm, L.; Willemson, J. Secure floating point arithmetic and private satellite collision analysis. *Int. J. Inf. Sec.* **2015**, *14*, 531–548. [CrossRef]
8. Archer, D.W.; Bogdanov, D.; Pinkas, B.; Pullonen, P. Maturity and performance of programmable secure computation. *IEEE Secur. Priv.* **2016**, *14*, 48–56. [CrossRef]
9. Chung, H.; Kim, M. Encoding of rational numbers and their homomorphic computations for FHE-based applications. *Int. J. Found. Comput. Sci.* **2018**, *29*, 1023–1044. [CrossRef]
10. Bogetoft, P.; Damgård, I.; Jakobsen, T.P.; Nielsen, K.; Pagter, J.; Toft, T. A practical implementation of secure auctions based on multiparty integer computation. In Proceedings of the International Conference on Financial Cryptography and Data Security, Anguilla, UK, 27 February–2 March 2006; pp. 142–147.
11. Jha, S.; Kruger, L.; Shmatikov, V. Towards practical privacy for genomic computation. In Proceedings of the 2008 IEEE Symposium on Security and Privacy (sp 2008) 2008, Oakland, CA, USA, 18–22 May 2008; pp. 216–230.
12. Burkhart, M.; Dimitropoulos, X. Fast privacy-preserving top-k queries using secret sharing. In Proceedings of the International Conference on Computer Communications and Networks, Zurich, Switzerland, 2–5 August 2010; pp. 1–7.
13. Many, D. Privacy-Preserving Collaboration in Network Security. Master's Thesis, ETH Zürich, Zürich, Switzerland, 2009.
14. Huang, Y.; Evans, D.; Katz, J. Private set intersection: Are garbled circuits better than custom protocols? In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, San Diego, CA, USA, 5–8 February 2012.
15. Chatterjee, A.; Kaushal, M.; Sengupta, I. Accelerating sorting of fully homomorphic encrypted data. In *Progress in Cryptology—INDOCRYPT, Proceedings of the 14th International Conference on Cryptology in India, Mumbai, India, 7–10 December 2013*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 262–273.
16. Çetin, G.S.; Doröz, Y.; Sunar, B.; Savas, E. Depth optimized efficient homomorphic sorting. In *Progress in Cryptology—LATINCRYPT*; Springer: Cham, Switzerland, 2015; pp. 61–80.
17. Bost, R.; Ada Popa, R.; Tu, S.; Goldwasser, S. Machine learning classification over encrypted data. In Proceedings of the NDSS Symposium 2015, San Diego, CA, USA, 8–11 February 2015.
18. Wu, D.; Feng, T.; Naehrig, M.; Lauter, K. Privately evaluating decision trees and random forests. *PoPETs* **2016**, *2016*, 335–355. [CrossRef]
19. Rahulamathavan, Y.; Phan, R.C.; Veluru, S.; Cumanan, K.; Rajarajan, M. Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud. *IEEE Trans. Dependable Sec. Comput.* **2014**, *11*, 467–479. [CrossRef]

20. Erkin, Z.; Veugen, T.; Toft, T.; Lagendijk, R. Generating private recommendations efficiently using homomorphic encryption and data packing. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 1053–1066. [CrossRef]
21. Kim, H.I.; Choi, M.; Kim, H.J.; Chang, J.W. A secure range query processing algorithm for the encrypted database on the cloud. In *Advanced Multimedia and Ubiquitous Engineering*; Springer: Singapore, 2016; pp. 101–110.
22. Catrina, O.; de Hoogh, S. Secure multiparty linear programming using fixed-point arithmetic. In Proceedings of the ESORICS, Athens, Greece, 20–22 September 2010; pp. 134–150.
23. Kerschbaum, F.; Schröpfer, A.; Zilli, A.; Pibernik, R.; Catrina, O.; de Hoogh, S.; Schoenmakers, B.; Cimato, S.; Damiani, E. Secure collaborative supply-chain management. *IEEE Comput.* **2011**, *44*, 38–43. [CrossRef]
24. Piva, A.; Katzenbeisser, S. Signal processing in the encrypted domain. *EURASIP J. Inf. Secur.* **2007**, *2007*, 082790. [CrossRef]
25. Franz, M.; Katzenbeisser, S. processing encrypted floating point signal. In Proceedings of the thirteenth ACM multimedia workshop on Multimedia and Security, Buffalo, NY, USA, 29–30 September 2011; pp. 103–108.
26. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology—EUROCRYPT*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 223–238.
27. Jäschke, A.; Armknecht, F. Accelerating homomorphic computations on rational numbers. In Proceedings of the ACNS, London, UK, 19–22 June 2016; Springer: Cham, Switzerland, 2016; pp. 405–423.
28. Costache, A.; Smart, N.; Vivek, V.; Waller, A. Fixed point arithmetic in SHE scheme. In Proceedings of the 23rd International Conference, St. John's, NL, Canada, 10–12 August 2016; Volume 250.
29. Bonte, C.; Bootland, C.; Bos, J.W.; Castryck, W.; Iliashenko, I.; Vercauteren, F. Faster homomorphic function evaluation using non-integral base encoding. In *Cryptographic Hardware and Embedded Systems*; Springer: Cham, Switzerland, 2017; pp. 579–600.
30. Chen, H.; Laine, K.; Player, R.; Xia, Y. High-precision arithmetic in homomorphic encryption. In Proceedings of the Cryptographers' Track at the RSA Conference, San Francisco, CA, USA, 16–20 April 2018; Springer: Cham, Switzerland, 2018; pp. 116–136.
31. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y.S. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology—ASIACRYPT*; Springer: Cham, Switzerland, 2017; pp. 409–437.
32. Cheon, J.H.; Kim, D.; Kim, D.; Lee, H.; Lee, K. Numerical method for comparison on homomorphically encrypted numbers. In *Advances in Cryptology—ASIACRYPT*; Springer: Cham, Switzerland, 2019; pp. 415–445.
33. Microsoft. SEAL: Simple Encrypted Arithmetic Library. 2014. Available online: https://www.microsoft.com/en-us/research/project/simpleencrypted-arithmetic-library/ (accessed on 2 February 2019).
34. Togan, M.; Plesca, C. Comparison-based computations over fully homomorphic encrypted data In Proceedings of the International Conference on Communications, Bangkok, Thailand, 10–12 October 2014; pp. 1–6.
35. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
36. Coron, J.; Mandal, A.; Naccache, D.; Tibouchi, M. Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology—CRYPTO, Proceedings of the 31st Annual Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2011*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 487–504.
37. Ducas, L.; Micciancio, D. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology—EUROCRYPT, Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, 26–30 April 2015*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 617–640.
38. van Dijk, M.; Gentry, C.; Halevi, S.; Vaikuntanathan, V. Fully homomorphic encryption over the integers. In *Advances in Cryptology—EUROCRYPT, Proceedings of the Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco and Nice, France, 30 May–3 June 2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 24–43.
39. Brakerski, Z.; Vaikuntanathan, V. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In *Advances in Cryptology—CRYPTO, Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 15-19 May 2011*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 505–524.

40. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, New York, NY, USA, 8–10 January 2012; pp. 309–325.
41. Bos, J.W.; Lauter, K.E.; Loftus, J.; Naehrig, M. Improved security for a ring-based fully homomorphic encryption scheme. In *IMA International Conference on Cryptography and Coding*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 45–64.
42. Goldwasser, S.; Micali, S. Probabilistic encryption. *J. Comput. Syst. Sci.* **1984**, *28*, 270–299. [CrossRef]
43. Goldreich, O. *Foundations of Cryptography-Volume II Basic Applications*; Cambridge University Press: Cambridge, UK, 2004.
44. Cheon, J.H.; Kim, M.; Kim, M. Optimized search-and-compute circuits and their application to query evaluation on encrypted data. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 188–199. [CrossRef]
45. Kim, M.; Lee, H.T.; Ling, S.; Wang, H. On the efficiency of FHE-based private queries. *IEEE Trans. Dependable Sec. Comput.* **2018**, *15*, 357–363. [CrossRef]
46. Kim, M.; Lee, H.T.; Ling, S.; Ren, S.Q.; Tan, B.H.M.; Wang, H. Search condition-hiding query evaluation on encrypted databases. *IEEE Access* **2019**, *7*, 161283–161295. [CrossRef]
47. Gentry, C.; Halevi, S.; Smart, N. Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology—EUROCRYPT*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 465–482.
48. Smart, N.P.; Vercauteren, F. Fully homomorphic SIMD operations. *Des. Codes Cryptogr.* **2014**, *71*, 57–81. [CrossRef]
49. Hardy, G.; Wright, E. *An Introduction to the Theory of Numbers*; Clarendon Press: Oxford, UK, 1979.
50. Graepel, T.; Lauter, K.; Naehrig, M. ML confidential: Machine learning on encrypted data. In Proceedings of the International Conference on Information Security and Cryptology, Seoul, Korea, 27–29 November 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 1–21.
51. Shoup, V. NTL: A Library for Doing Number Theory Version 11.3.2. 2018. Available online: http://www.shoup.net/ntl/ (accessed on 2 February 2019).
52. Halevi, S.; Shoup, V. HElib: Software Library for Homomorphic Encryption. 2018. Available online: http://github.com/shaih/HElib.git (accessed on 2 February 2019).
53. OpenMP Architecture Review Board. OpenMP Application Program Interface Version 3.0, 2018. Available online: http://www.openmp.org/mp-documents/spec30.pdf (accessed on 2 February 2019).
54. Agrawal, R.; Asonov, D.; Srikant, R. Enabling sovereign information sharing using web services. In Proceedings of the 2004 ACM SIGMOD international conference on Management of data, Paris, France, 13–18 June 2004; pp. 873–877.
55. Boneh, D.; Lewi, K.; Raykova, M.; Sahai, A.; Zhandry, M.; Zimmerman, J. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In Advances in Cryptology—EUROCRYPT, Proceedings of the 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, 26–30 April 2015.
56. Chenette, N.; Lewi, K.; Weis, S.A.; Wu, D.J. Practical order-revealing encryption with limited leakage. In *Fast Software Encryption*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 474–493.
57. Lewi, K.; Wu, D.J. Order-revealing encryption: New constructions, applications, and lower bounds. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 1167–1178.
58. Durak, F.B.; DuBuisson, T.M.; Cash, D. What else is revealed by order-revealing encryption? In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 1155–1166.
59. Song, D.X.; Wagner, D.; Perrig, A. Practical techniques for searches on encrypted data. In Proceeding of the 2000 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 14–17 May 2000; pp. 44–55.
60. Boneh, D.; Gentry, C.; Halevi, S.; Wang, F.; Wu, D.J. Private database queries using somewhat homomorphic encryption. In *Applied Cryptography and Network Security*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 102–118.

61. Cheon, J.H.; Kim, M.; Kim, M. Search-and-compute on encrypted data. In Proceeding of the 2000 International Conference on Financial Cryptography and Data Security, San Juan, Puerto Rico, 26–30 January 2015; pp. 1–18.
62. Gosper, R. Continued fraction arithmetic. In *HAKMEM Item 101B, MIT Artificial Intelligence Memo 239*; The MIT Press: Cambridge, MA, USA, 1977.