Privacy-Preserving Computation Tookit on Floating-Point Numbers

Zekai Chen[1], Zhiwei Zheng[1], Ximeng Liu[1,2], and Wenzhong Guo[1,2](✉)

[1] College of of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China
guowenzhong@fzu.edu.cn

[2] Fujian Provincial Key Laboratory of Information Security of Network Systems, Fuzhou University, Fuzhou 350108, China

Abstract. Computation outsourcing using virtual environment is getting more and more prevalent in cloud computing, which several parties want to run a joint application and preserves the privacy of input data in secure computation protocols. However, it is still a challenging task to improve the efficiency and speed of secure floating point calculations in computation outsourcing, which has efficient secure integer calculations. Therefore, in this paper, we propose a framework built-up with a privacy-preserving computation toolkit with floating-point numbers (FPN), called PCTF. To achieve the above goal, we provide efficient toolkit to ensure their own data that FPN operations can be securely handled by homomorphic encryption algorithm. Moreover, we provide simulation results to experimentally evaluate the performance of the accuracy and the efficiency of PCTF, which will slowdown with 10x time consumption per in the secure floating-point addition and secure floating-point multiplication. Existing FPN division is constantly approaching result of division, or obtaining the quotient and remainder of division, in terms of precision, it is impossible to guarantee the precise range stably. However, our PCTF has higher precision in secure floating-point division, and the precision can be guaranteed at least 10^{-17}.

Keywords: Privacy computation · Homomorphic encryption · Multiple keys · Secure computation

1 Introduction

Nowadays, the cloud computing paradigm [1] revolutionizing the organizations' way of operating their data particularly in the way they store, access and process data. On one hand, according to IDC's data age 2025 [2] report, the annual data generated in the world will grow from 45 ZB to 175 ZB from 2019 to 2025. Local users can not afford such a large amount of data computing, more and more enterprise customers' applications such as Internet of Things (IoT) [3], self-driving car [4], and scientific research [5] need to be deployed in the data center or

J. Xiong et al. (Eds.): MobiMedia 2021, LNICST 394, pp. 462–476, 2021.
https://doi.org/10.1007/978-3-030-89814-4_33

cloud. On the other hand, cloud computing [6], with its cost-efficiency, flexibility, and offload of administrative overhead, is used by the majority of enterprise users. Organizations are prone to delegate their computational operations in addition to their data to the cloud.

Despite the tremendous advantages that the cloud offers, privacy and security issues in the cloud are preventing companies to utilize those advantages. When data are highly sensitive, e.g., the medical [6,7]. To protect the data privacy [8,9], data owners typically encrypt their data before outsourcing to the cloud. Homomorphic encryption [10] is a promising solution to avoid privacy leakage risks while protecting data confidentiality. Figure 1 plots a typical multiparty network with several different participants that own restricted storage ability and limited computing power. They jointly transmit data to the cloud server for data calculation. However, there exist challenges in the following two aspects. First, cloud service providers often store data belonging to multiple participants on the same server [11] to reduce operational costs. Therefore, different participants should be distributed with an individual key [12], to avoid multi-tenancy related attacks. How to achieve multiple keys computation without compromising the privacy of the outsourced data becomes a challenging issue. Second, cryptosystems are generally designed to protect only integer values. Computations on integers envivnately affect the accuracy of data and decision making results, which may even lead to wrong diagnosis in patients' ill [7] and wrong decision in self-driving [4].

In this paper, we present a **P**rivacy-Preserving **C**omputation **T**ookit on **F**loating-Point Numbers. The toolkit is constructed using secure computation based on homomorphic encryption and provide perfect or statistical privacy in the semi-honest model. The desirable features of PCTF include the following:

- **Secure FPN Computation:** We build a privacy-preserving outsourced computation of toolkit FPN operations with multiple keys. The toolkit offers multiplication, addition, subtraction, division with FPN. Secure multiplication addition, subtraction of FPN are trivial extensions of the integer operations. Meanwhile, we present new protocols for the division of FPN.
- **Multiple Users:** It has good expandability, including convenience and cost-effectiveness in supporting multiple secret keys in computing. Different data providers encrypt their data with their own public keys to the cloud server. The cloud server carries out secure computation in the case of multiple keys.
- **Efficiency and Accuracy:** Our calculation toolkit equips with secure and efficient encrypting FPN. Extensive experiments indicate that our scheme has high efficiency and lower accuracy loss, especially in the division.

The rest of this paper is organized as follows. In Sect. 2, we introduce the related work of secure multiparty computation protocols. In Sect. 3, we describe the preliminaries required in the understanding of our proposed FPN operations. In Sect. 4, we formalize the system model as well as the attacker model. Then we give a detailed introduction about privacy-preserving FPN computation protocols in Sect. 5. The security analysis and performance evaluation are presented in Sects. 6 and 7, respectively. Section 8 concludes this paper.

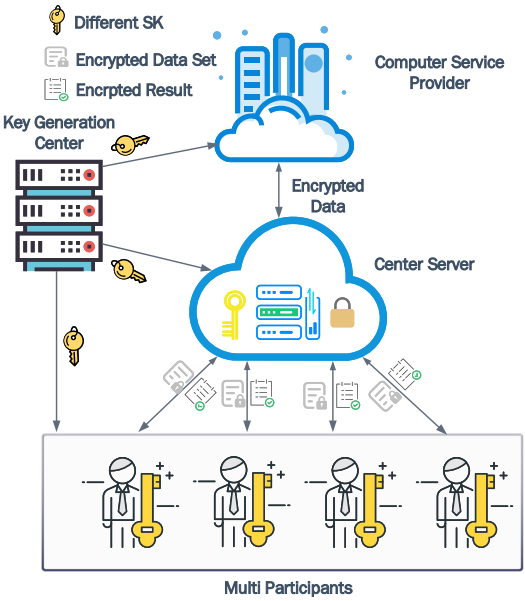

Fig. 1. System model

2 Related Work

Earlier work on privacy-preserving machine learning [13,14] has been proposed to provide privacy preservation during the training phase, but these schemes lacked implementation. Secure integer protocols have been studied in numerous researches. e.g., garbled circuit based protocol [15], homomorphic encryption based protocols [16–21] and secret sharing based protocols [21–24]. However, garbled circuit suffer from very high computation and communication complexities. Secret sharing needs all the servers to jointly compute some functions interactively. It requires multiple servers to store certain redundant data and requires pairwise secure channels between servers, so it is may difficult to implement in a number of real-world scenarios. Catrina et al. [20] presents a family of protocols for multiparty computation with rational numbers using floating-point representation. This approach offers more efficient solutions for secure computation than other usual representations, but it becomes quite complex when the divisor is secret. Veugen et al. [19] presents a secure integer division protocol that outputs an approximate quotient based on homomorphic encryption, such approximate protocols could be only useful for secure clustering and statistical analysis. Existing secure computation [17] with FPN base on homomorphic encryption can be realized at huge computation cost and communication overhead, and security division protocol is not implemented. Liu et al. [18] solves secure integer division by getting quotient and remainder, but it does not have precision, and it is impossible to obtain the complete value after division.

3 Preliminary

In this section, we introduce some terminologies and basic knowledge. First, some important notations used in this paper are provided in Table 1.

3.1 Distributed Two Trapdoors Public-Key Cryptosystem

DT-PKC splits a strong private key into different shares. In addition, the weak decryption algorithm should support distributed decryption to solve the authorization problem in the multi-key environment, follows the idea in [18], and works as follows:

- **raw_encryption**: Given a message $m \in \mathbb{Z}_N$, we compute $N = p \times q$ and $\lambda = \frac{(p-1)(q-1)}{2}$. We can choose a random number $r \in \mathbb{Z}_N$, and the ciphertext is generated as:

$$[\![m]\!]_{pk} = g^m r^N \bmod N^2 = (1 + m \times N) r^N \bmod N^2 \tag{1}$$

- **raw_decryption**: Given a message $[\![m]\!]_{pk}$, $N = p * q$ and $\lambda = \frac{(p-1)(q-1)}{2}$, using the decryption algorithm $D_{SK}(\cdot)$ and the corresponding private key $SK = \lambda$, the algorithm can compute to the plaintext. Since gcd $(\lambda, N) = 1$, the ciphertext is generated as:

$$[\![m]\!]^{\lambda} \bmod N^2 = r^{\lambda N}(1 + mN\lambda) \bmod N^2 = (1 + \mathrm{m}N\lambda) \tag{2}$$

$$m = L\left([\![m]\!]^{\lambda} \bmod N^2\right) \lambda^{-1} \bmod N \tag{3}$$

Table 1. Notations

Symbol	Definition
pk_u, sk_u	A pair of public-private keys
SK_1, SK_2	Partial decrypted private keys
$[\![m]\!]_{pk_u}$	A ciphertext encrypted with pk_u
$\|\cdot\|$	The data length of an arbitrary variable
κ	The security parameter variable
p, q	p and q are two big primes, $\|p\| = \|q\| = \kappa$
N	N is a big integer satisfying $N = p \times q$
$\mathbb{Z}_N$	An integer field of N
$\langle m \rangle$	Encrypted object of m with FPN

3.2 Privacy-Preserving Multiple Keys Integer Protocol

In particular, we assume there exits of semi-honest cloud server CP, CSP, [18] and their own public key pk_1, pk_2 from parties ($user_1$, $user_2$). Based on DT-PKC, the strong private key can be split into the different partial strong private key. Moreover, CP has partial strong private key SK_1, and CSP has partial strong private SK_2. All of the below protocols are considered under CP and CSP:

- **Secure Addition (SAD):** Consider CP with private input $(\llbracket a \rrbracket_{pk_1}, \llbracket b \rrbracket_{pk_2})$, partial strong private key SK_1, and CSP with the partial strong private SK_2. The goal of the secure addition is to return the encryption of $\llbracket a+b \rrbracket_{pk}$ to CP, and then the $\llbracket a+b \rrbracket_{pk}$ will be returned to users. The basic idea of the SAD protocol is based on the following property which holds for any given $a, b \in \mathbb{Z}_N$, with different public key.
  - CP: generate $r_a, r_b \in \mathbb{Z}_N$, obtain $\llbracket r_a + r_b \rrbracket_{pk}$ and compute $D_{SK_1}(\llbracket a + ra \rrbracket_{pk})$, $D_{SK_1}(\llbracket b + r_b \rrbracket_{pk})$ to CSP.
  - CSP: decrypt r_a, r_b by SK_2, compute $\llbracket a + r_a + b + r_b \rrbracket_{pk}$ to CP.
  - CP: obtain $\llbracket a+b \rrbracket_{pk} = \llbracket a + r_a + b + r_b \rrbracket_{pk} \cdot \llbracket -r_a - r_b \rrbracket_{pk}$.
- **Secure Multiplication (SM):** In this protocol, the goal of the secure multiplication is to return the encryption of $\llbracket a \times b \rrbracket_{pk}$ to CP, and then the $\llbracket a \times b \rrbracket_{pk}$ will be returned to users. The basic idea of the SM protocol is based on the following property which holds for any given $a, b \in \mathbb{Z}_N$, with the different public key.
  - CP: generate $r_a, r_b \in \mathbb{Z}_N$, obtain $\llbracket a + r_a \rrbracket_{pk}, \llbracket b + r_b \rrbracket_{pk}$ and compute $D_{SK_1}(\llbracket a + ra \rrbracket_{pk})$, $D_{SK_1}(\llbracket b + r_b \rrbracket_{pk})$ to CSP.
  - CSP: decrypt $(a + r_a)(b + r_b)$ by SK_2, compute $\llbracket h \rrbracket_{pk} = \llbracket a \times b + a \times r_b + b \times r_a + r_a \times r_b \rrbracket_{pk}$ to CP.
  - CP: obtain $\llbracket a \times b \rrbracket_{pk} = \llbracket h \rrbracket_{pk} \cdot \llbracket -a \times r_b \rrbracket_{pk} \cdot \llbracket -b \times r_a \rrbracket_{pk} \cdot \llbracket -r_a \times r_b \rrbracket_{pk}$.
- **Secure Exponent Calculation (SEXP):** Given a plaintext number a and an encrypted integer number $\llbracket b \rrbracket_{pk}$ $(a, b \in \mathbb{Z}_N)$, the SEXP protocol will provide the encrypted data $\llbracket U \rrbracket_{pk}$, s.t., $\llbracket U \rrbracket_{pk} = a^{\llbracket b \rrbracket_{pk}}$. The SEXP protocol is described as follows: $a^{\llbracket b \rrbracket_{pk}} = (a^{\llbracket b+r \rrbracket_{pk}})^{a^{-r}} = a^{\llbracket b+r-r \rrbracket_{pk}}$.
- **Secure Greater Than (SGT):** Based on DT-PKC, the goal of the secure greater than is to return the encryption of $\llbracket u^* \rrbracket$ to CP, and then the $\llbracket u^* \rrbracket_{pk}$ will be returned to users. The basic idea of the SGT protocol is that we flip a coin to computer $\llbracket a \rrbracket_{pk} \cdot \llbracket 2 \times b + 1 \rrbracket_{pk}^{N-1}$ or $\llbracket 2 \times b + 1 \rrbracket_{pk} \cdot \llbracket a \rrbracket_{pk}^{N-1}$ to get result of relationship between a and b. If $u^* = 0$, it shows $a \geq b$; and If $u^* = 1$, it shows $a < b$.
- **Secure Maximum (SMAX):** Consider CP with input$(\llbracket a \rrbracket_{pk_1}, \llbracket b \rrbracket_{pk_2}, \llbracket c \rrbracket_{pk_3})$. The goal of the secure maximum is to return the encryption of the maximum number between input to CP, and then the result will be returned to users. We can call SGT to computer $\llbracket u \rrbracket^* \leftarrow \mathsf{SGT}(\llbracket a \rrbracket_{pk_1}, \llbracket y \rrbracket_{pk_2})$ to get $\llbracket u^* \rrbracket_{pk}$, and then $\llbracket A \rrbracket_{pk}$ and $\llbracket L \rrbracket_{pk}$ can be calculated by $\llbracket A \rrbracket_{pk} = \llbracket (1 - u^*) \times a + u^* \times b \rrbracket_{pk}$ and $\llbracket L \rrbracket_{pk} = \llbracket (1 - u^*) \times b + u^* \times a \rrbracket_{pk}$, s.t., $A \geq I$. Through multiple use of SGT, we will find the encryption of the maximum number.

3.3 Floating-Point Number

In this paper, we present FPN as $u = (s_u, b, e_u)$. Thus, we focus on the IEEE 754 standard for oating-point arithmetic. Therefore, IEEE numbers are stored using a kind of scientic notation. Floats are approximated using a binary fraction with the numerator using the first 53 bits starting with the most signicant bit and with the denominator as a power of two.

Under these circumstances, mantissa can be expressed as FPN. We consider the FPN is the set of number, which can be expressed as fractional number like $\frac{a}{b}$ $(a, b \in \mathbb{Z}_N)$. Meanwhile, b can be set for approximated 2^n. Therefore, we can get the numerator of the integer by multiplying by the denominator:

Algorithm 1: Floating-Point Number Encoding Protocol (FPE)

Input: f of FPN
Output: (s_f, b, e_f)
$base = 16$
$s, e < -\text{Frexp}(f)$
$e_f = \frac{(e - FLOAT\_BIT)}{log_2(base)}$
$s_f = round(\text{Fraction}(f) \times \text{Fraction}(base)^{-e_f}$
return $(s_f, b = base, e_f)$

In this protocol, we use Frexp function in python, which is used to calculate mantissa and exponent of FPN. Meanwhile, we use Fraction function in python, which can be used to calculate the numerator and denominator of FPN. According to the maximum number of FPN that python can represent, we can convert the mantissa into an integer. Meanwhile, we can see the mantissa from FPN convert integer number in Table 2.

Table 2. Frexp number table

Number	Mantissa	Base	Exponent
232342.3353423	0.88631567131919865	2	18
232342.3353423	15966443708343096	16	−9

Based on Algorithm 1, we easily get decoding of encoding-floating. Mantissa is defined as s. Base is defined as b. Exponent is defined as e. Thus, such a format is a number for which there exists at least one representation triplet (s, b, e):

$$f = s \times b^e \tag{4}$$

4 System Model and Privacy Requirement

In this section, we system model and e privacy requirement of PCTF.

4.1 System Model

In this session, we first present PCTF model that is comprised of four kinds of entities, namely a center server (CP), computer service provider (CSP), multi-participant (MP), and key generation center (KGC).

- **CP**: A CP stores and manages data outsourced from MP. CP also stores all the intermediate and final results in encrypted form. Furthermore, a CP is able to perform certain calculations over encrypted data.

- **CSP**: A CSP is responsible for assisting CP to complete the complex computations, which is able to partial decrypt ciphertexts sent by the CP, perform certain calculations over the partial decrypted data, and then reencrypt the calculated results.
- **MP**: Multi participants provide the encrypted data with the their own public keys. The goal of MP is to request a CP to perform some calculations over the encrypted data under multiple keys.
- **KGC**: The trusted KGC is tasked with the distribution and management of both public and private keys in the system. The KGC can split the strong private key into SK_1, SK_2 to CP and CSP, respectively.

4.2 Threat Model

In our threat model, MP, CP, and CSP are semi-honest parties, which strictly follow the protocol, but are also interested to learn data belonging to other parties. Therefore, we introduce an adversary $\mathcal{A}$ in our model. The goal of $\mathcal{A}$ is to decrypt the challenge MP's ciphertext with the following capabilities:

- $\mathcal{A}$ may eavesdrop on all communication links to obtain encrypted data.
- $\mathcal{A}$ may compromise the CSP to guess plaintext values of all ciphertexts sent from the CP by executing an interactive protocol.
- $\mathcal{A}$ may compromise the CP to guess plaintext values of all ciphertexts outsourced from the challenge MP, and all the ciphertexts sent from the CSP by executing an interactive protocol.
- $\mathcal{A}$ may compromise MP, with the exception of the challenge MP, to get access to their decryption capabilities, and try to guess all plaintexts belonging to the challenge MP.

The adversary $\mathcal{A}$, however, is restricted from compromising (1) both the CSP and the CP concurrently, and (2) the challenge MP. We remark that such restrictions are typical in adversary models used in cryptographic protocols.

5 Privacy-Preserving FPN Computation Protocols

In this section, we propose a set of generic protocols, which is used in cloud computation. All of the below protocols are under semi-honest parties. In particular, the FPN can be convert to FPE with Algorithm 1.

In order to realize PCTF, we can define the relative functions as follow:

- $\mathsf{FPE}(m)$: Function is defined by converting FPN to FPE, which outputs a result that $m = (s_m, b, t_m)$.
- $\mathsf{Encrypt}(s_m, e_m)$: Function encrypts the s_m and e_m by raw_encryption function, which outputs Encryption Object $\langle m \rangle = (\llbracket s_m \rrbracket_{pk}, b, \llbracket t_m \rrbracket_{pk})$.
- $\mathsf{Decrypt}((\llbracket s_m \rrbracket_{pk}, b, \llbracket t_m \rrbracket_{pk}))$: Given encrypted object, we choose raw_decryption to decrypt $\llbracket s_m \rrbracket_{pk}$ and $\llbracket t_m \rrbracket_{pk}$ separately. We also use FPE to assemble s_m and t_m into FPN as output.
- $\mathsf{SforCP}(\llbracket m \rrbracket_{pk})$: Given encrypted integer number, we can choose random $r \in \mathbb{Z}_N$, and get $\llbracket r \rrbracket_{pk}$ to send $\llbracket m + r \rrbracket_{pk}$ from CP to CSP.
  - CP: generate $r \in \mathbb{Z}_N$, compute $D_{SK_1}(\llbracket m + r \rrbracket_{pk})$ to CSP.
  - CSP: obtain $D_{SK_1}(\llbracket m + r \rrbracket_{pk})$ compute $D_{SK_2}(D_{SK_1}(\llbracket m + r \rrbracket_{pk}))$ to CP.
  - CP: obtain m.

5.1 Secure FPN Addition

Given two encrypted FPNs $\langle u \rangle = (\llbracket s_u \rrbracket_{pk}, b, \llbracket e_u \rrbracket_{pk})$ and $\langle v \rangle = (\llbracket s_v \rrbracket_{pk}, b, \llbracket e_v \rrbracket_{pk})$, the goal of the SFAD protocol is to securely compute two encrypted FPNs addition. The overall steps of SFAD protocol are shown as follows:

Algorithm 2: Secure Floating Addition Protocol (SFAD)

Input: Two encrypted FPNs $\langle u \rangle$ and $\langle v \rangle$
Output: Encrypted FPN $\langle T \rangle$

Both CP and CSP jointly calculate
$\llbracket G \rrbracket_{pk} \leftarrow \mathsf{SGT}(\llbracket e_u \rrbracket_{pk_1}, \llbracket e_v \rrbracket_{pk_2})$
if $\mathit{SforCP}(\llbracket G \rrbracket_{pk})$ **then**
  $\llbracket e_T \rrbracket_{pk} \leftarrow \mathsf{SAD}(\llbracket e_v \rrbracket_{pk_2}, \llbracket O \rrbracket_{pk})$
  $\llbracket t \rrbracket_{pk} \leftarrow \mathsf{SAD}(\llbracket e_u \rrbracket_{pk_1}, \llbracket e_v \rrbracket_{pk_2}^{N-1})$
  $\llbracket p \rrbracket_{pk} \leftarrow \mathsf{SM}(\llbracket s_u \rrbracket_{pk_1}, \mathsf{SEXP}(b, \llbracket t \rrbracket_{pk}))$
  $\llbracket s_T \rrbracket_{pk} \leftarrow \mathsf{SAD}(\llbracket s_v \rrbracket_{pk_2}, \llbracket p \rrbracket_{pk})$
else
  $\llbracket e_T \rrbracket_{pk} \leftarrow \mathsf{SAD}(\llbracket e_u \rrbracket_{pk_1}, \llbracket O \rrbracket_{pk})$
  $\llbracket t \rrbracket_{pk} \leftarrow \mathsf{SAD}(\llbracket e_v \rrbracket_{pk_2}, \llbracket e_u \rrbracket_{pk_1}^{N-1})$
  $\llbracket p \rrbracket_{pk} \leftarrow \mathsf{SM}(\llbracket s_v \rrbracket_{pk_2}, \mathsf{SEXP}(b, \llbracket t \rrbracket_{pk}))$
  $\llbracket s_T \rrbracket_{pk} \leftarrow \mathsf{SAD}(\llbracket s_u \rrbracket_{pk_1}, \llbracket p \rrbracket_{pk})$
$\langle T \rangle = (\llbracket s_T \rrbracket_{pk}, b, \llbracket e_T \rrbracket_{pk})$
return $\langle T \rangle$

During this protocol, using the SAD, SM, SEXP, SforCP protocols, it easily aligns two encrypted number of exponent part. Meanwhile, SAD might be used to add two encrypted numbers of mantissa to get the result.

5.2 Secure FPN Multiplication

Given two encrypted FPNs $\langle u \rangle = (\llbracket s_u \rrbracket_{pk}, b, \llbracket e_u \rrbracket_{pk})$ and $\langle v \rangle = (\llbracket s_v \rrbracket_{pk}, b, \llbracket e_v \rrbracket_{pk})$, the goal of the SFM protocol is to securely compute two encrypted FPNs multiplication. The overall steps of SFM protocol are shown as follows:

Algorithm 3: Secure Floating Multiplication Protocol (SFM)

Input: Two encrypted FPNs $\langle u \rangle$ and $\langle v \rangle$
Output: Encrypted FPN $\langle T \rangle$

Both CP and CSP jointly calculate
$\llbracket s_T \rrbracket_{pk} \leftarrow \mathsf{SM}(\llbracket s_u \rrbracket_{pk1}, \llbracket s_v \rrbracket_{pk2})$
$\llbracket e_T \rrbracket_{pk} \leftarrow \mathsf{SAD}(\llbracket e_u \rrbracket_{pk1}, |\llbracket e_v \rrbracket_{pk2})$
$\langle T \rangle = (\llbracket s_T \rrbracket_{pk}, b, \llbracket e_T \rrbracket_{pk})$
return $\langle T \rangle$

Base on SM and SAD protocol, we multiply the encrypted mantissa part by SM protocol and add the encrypted exponent part by SAD protocol.

5.3 Secure Integer Digits

Here, given encrypted integer $[\![n]\!]_{pk}$, we propose the secure digits protocol to obtain result $[\![d]\!]_{pk}$ to get digit of n.

Algorithm 4: Secure Digits Protocol (SDG)

Input: Encrypted integer $[\![n]\!]_{pk}$
Output: Encrypted digit $[\![d]\!]_{pk}$

In CP:
  Random select $r_a \in \mathbb{Z}_N$, then Compute $[\![r_a]\!]_{pk}$
  $[\![n + r_a]\!]_{pk} \leftarrow \mathsf{SM}([\![n]\!]_{pk_1}, [\![r_a]\!]_{pk})$
  $[\![t]\!]_{pk} = D_{sk_1}([\![n \times r_a]\!]_{pk})$
  Send $[\![t]\!]_{pk}, [\![r_a]\!]_{pk}$ to CSP
In CSP:
  Receive $[\![t]\!]_{pk}$ from CP
  $t = D_{sk_2}([\![t]\!]_{pk})$
  $t_n = Math.Log_{10} t$ in plaintext
  $t_{d_n} = int(round(tnum - int(tnum)) \times 100))$
  Send $[\![t_{d_n}]\!]_{pk}, [\![int(t_n)]\!]_{pk}$ to CP
In CP:
  Receive $[\![t_{d_n}]\!]_{pk}, [\![int(t_n)]\!]_{pk}$ from CSP
  $t_r = Math.Log_{10} r_a$ in plaintext
  $t_{d_r} = int(round(tnum - int(tnum)) \times 100))$
  $[\![t_s]\!]_{pk} = [\![t_{dn}]\!]_{pk} \cdot [\![t_{dr}]\!]_{pk}^{N-1}$
  $[\![lt]\!]_{pk} \leftarrow \mathsf{SGT}([\![t_s]\!]_{pk}, [\![50]\!]_{pk})$
  $[\![d]\!]_{pk} = [\![int(t_n)]\!]_{pk} \cdot [\![int(t_r)]\!]_{pk}^{N-1} \cdot [\![lt]\!]_{pk} \cdot [\![1]\!]_{pk}$
  return $[\![d]\!]_{pk}$

In this protocol, the goal of the SDG protocol is to securely compute the decimal digits of the encrypted number. During this protocol, we assume that CP can coordinate with CSP. The basic idea of the SDG protocol is based on the following property which holds for any given number:

$$\mathrm{d} = \lceil \log_{10} n \rceil + 1 \tag{5}$$

where all the arithmetic operations are performed under $\mathbb{Z}_N$. The overall steps in SDG are shown in Algorithm 4, we can suppose the d_s to get $\log_{10} n$ whether to add one after rounding up. The following equation is d_s:

$$\mathrm{d}_s = \lceil \log_{10}(\mathrm{n} \times r_a) - \lfloor \log_{10}(\mathrm{n} \times r_a) \rfloor - \log_{10} r_a + \lfloor \log_{10} r_a \rfloor \times 100 \rceil \tag{6}$$

$$[\![t_s]\!]_{pk} = [\![\lfloor \log_{10}(\mathrm{n} \times ra) \rfloor]\!]_{pk} \cdot [\![\lfloor \log_{10} ra \rfloor]\!]_{pk}^{N-1} \tag{7}$$

$$[\![\mathrm{d}]\!]_{pk} = [\![t_s]\!]_{pk} \cdot SGT\left([\![d_s d_s d_s d_s]\!]_{pk}, [\![50]\!]_{pk}\right) \cdot [\![1]\!]_{pk} \tag{8}$$

5.4 Secure Integer Modular Calculation

Here, the goal of the SMOD protocol is to compute given the two encrypted integers $[\![f_1]\!]_{pk_1}$, $[\![f_2]\!]_{pk_2}$ division' quotient and remainder.

Algorithm 5: Secure Modular Calculation Protocol (SMOD)

Input: Two encrypted integers $[\![f_1]\!]_{pk_1}$, $[\![f_2]\!]_{pk_2}$
Output: Encrypted quotient $[\![q]\!]_{pk}$ and encrypted remainder $[\![r]\!]_{pk}$

Both CP and CSP jointly calculate
$[\![G_1]\!]_{pk} \leftarrow \mathsf{SGT}([\![0]\!]_{pk}, [\![f_1]\!]_{pk_1})$ and $[\![G_2]\!]_{pk} \leftarrow \mathsf{SGT}([\![0]\!]_{pk}, [\![f_2]\!]_{pk_2})$
$[\![sign]\!]_{pk}, [\![q]\!]_{pk} = [\![1]\!]_{pk}, [\![0]\!]_{pk}$
if $\mathit{SforCP}([G_1]_{pk})$ **then**
  $[\![f_1]\!]_{pk_1} = [\![f_1]\!]_{pk_1}^{N-1}$ and $[\![sign]\!]_{pk} = [\![sign]\!]_{pk}^{N-1}$
if $\mathit{SforCP}([G_2]_{pk})$ **then**
  $[\![f_2]\!]_{pk_2} = [\![f_2]\!]_{pk_2}^{N-1}$ and $[\![sign]\!]_{pk} = [\![sign]\!]_{pk}^{N-1}$
$[\![d_1]\!]_{pk} \leftarrow \mathsf{SDG}([\![f_1]\!]_{pk_1})$ and $[\![d_2]\!]_{pk} \leftarrow \mathsf{SDG}([\![f_2]\!]_{pk_2})$
$[\![d_s]\!]_{pk} = [\![d_1]\!]_{pk} \cdot [\![d_2]\!]_{pk}^{N-1}$
for $i \leftarrow 1$ **to** $\mathit{SforCP}([\![d_s]\!]_{pk})$ **do**
  $[\![p]\!]_{pk}, \langle t \rangle = [\![0]\!]_{pk}, \langle u \rangle$
  for $j \leftarrow 1$ **to** 9 **do**
    $[\![t_s]\!]_{pk} \leftarrow \mathsf{SM}([\![f_1]\!]_{pk_1}, [\![f_2]\!]_{pk_2}^{N-j-10\times i})$
    $[\![lt]\!]_{pk} \leftarrow \mathsf{SGT}([\![t_s]\!]_{pk}, [\![0]\!]_{pk})$
    $[\![t_p]\!]_{pk} \leftarrow \mathsf{SM}([\![lt]\!]_{pk}, [\![i]\!]_{pk})$
    $[\![p]\!]_{pk} \leftarrow \mathsf{SMAX}([\![p]\!]_{pk}, [\![t_p]\!]_{pk})$
  $[\![f_t]\!]_{pk} \leftarrow \mathsf{SM}([\![f_t]\!]_{pk}, [\![p]\!]_{pk}^{N-1}) \cdot [\![f_t]\!]_{pk}$
  $([\![q]\!]_{pk}, [\![r]\!]_{pk}) \leftarrow (\mathsf{SM}(([\![q]\!]_{pk}^{N-10})^{N-1} \cdot [\![p]\!]_{pk}, [\![sign]\!]_{pk}), [\![f_t]\!]_{pk})$
return $([\![q]\!]_{pk}, [\![r]\!]_{pk})$

To start with, CP computes decimal digits of $[\![f_1]\!]_{pk_1}$, $[\![f_2]\!]_{pk_2}$ to get times of the loop body is executed. Then, using the SGT, SMAX protocols, CP computes best number with the help of CSP, for $1 \le i \le 9$ to get $[\![q]\!]_{pk}$ and $[\![r]\!]_{pk}$.

5.5 Secure FPN Division

Algorithm 6: Secure Floating Division Protocol (SFDIV)

Input: Two encrypted FPNs $\langle u \rangle$ and $\langle v \rangle$
Output: Encrypted FPN $\langle T \rangle$

Both CP and CSP jointly calculate
$([\![q_1]\!]_{pk}, [\![r_1]\!]_{pk}) \leftarrow$SMOD$([\![s_u]\!]_{pk_1}, [\![s_v]\!]_{pk_2})$
$([\![q_2]\!]_{pk}, [\![r_2]\!]_{pk}) \leftarrow$SMOD$([\![r_1]\!]_{pk}^{pow(10,17)}, [\![s_v]\!]_{pk_2})$
$q_2 \leftarrow \mathsf{SforCP}([\![q_2]\!]_{pk})$, then $\langle t \rangle \leftarrow \mathsf{Encrypt}(q_2 * 10^{-17})$
$[\![e]\!]_{pk} \leftarrow \mathsf{SM}([\![e_u]\!]_{pk_1}, [\![e_v]\!]_{pk_2}^{N-1})$, then $[\![e_t]\!]_{pk} = [\![e]\!]_{pk} \cdot [\![e_t]\!]_{pk}$
$\langle T \rangle \leftarrow \mathsf{SFAD}(\langle t \rangle, ([\![q_1]\!]_{pk}, b = 16, [\![e_u]\!]_{pk_1}))$
return $\langle T \rangle$

In this protocol, the basic idea is based on the following property which gets for q_1, q_2, r_1 is the remainder of $\frac{u}{v}$ and r_2 is the remainder of $\frac{r_1 \times 10^{17}}{v} \in \mathbb{Z}_N$:

$$\frac{u}{v} = q_1 + \frac{r_1}{v} = q_1 + \frac{r_1 \times 10^{17}}{v} \approx q_1 + q_2 \times 10^{-17} \tag{9}$$

The overall steps involved in the SFDIV protocol are shown in Algorithm 6. In order to get higher precision, using SMOD protocol, CP computes $[\![q_1]\!]_{pk}$, $[\![r_1]\!]_{pk}$ of $\frac{[\![s_u]\!]_{pk_1}}{[\![s_v]\!]_{pk_2}}$, and then $[\![r_1]\!]_{pk}$ multiplies 10^{17} to do similarly steps to get $[\![q_2]\!]_{pk}$ and $[\![r_2]\!]_{pk}$. Observe that value of r_2 is very small, ignoring does not affect the accuracy. We can compute $\langle t \rangle = \mathsf{FPE}(r_1 * 10^{-17})$, $[\![e]\!]_{pk} = \mathsf{SM}([\![e_u]\!]_{pk_1}, [\![e_v]\!]^{N-1}_{pk_2})$.

$$x^* = ([\![q_1]\!]_{pk}, 16, [\![e_u]\!]_{pk}), \quad y^* = ([\![s_t]\!]_{pk}, 16, [\![e]\!]_{pk} \cdot [\![e_t]\!]_{pk}) \tag{10}$$

$$\langle x \rangle = \text{ Encrypt } (x^*), \langle y \rangle = \text{ Encrypt } (y^*) \tag{11}$$

$$\langle \frac{f_1}{f_2} \rangle = SFAD(\langle x \rangle, \langle y \rangle) \tag{12}$$

6 Security Analysis

In this section, we will analyze the security guarantees of the proposed protocols, which are based on DT-PKC.

Definition 1. *We assume that the adversary $\mathcal{A}$, and real world interacts with a simulator Sim to complete the process in the ideal world. We consider that FPN of $\langle x \rangle$, $\langle y \rangle$ are the input, Π is the corresponding protocol, and c denotes the computationally indistinguishable. If PCTF system is secure, which can be represented as* $\{\text{IDEAL}_{\Pi,\text{sim}}(\langle x \rangle, \langle y \rangle)\} \overset{c}{\approx} \{\text{REAL}_{\Pi,\text{A}}(\langle x \rangle, \langle y \rangle)\}$.

Lemma 1. *The protocol of SAD, SM, SGT, SMAX, and system of DT-PKC are secure in the honest-but curious model [17].*

Lemma 2. *Since the comparison result is that MP can decide the number to use to enter the corresponding branch together, we consider that the result of SGT is only safe for CP to know.*

Lemma 3. *FPN $f = (s_f, b_f, e_f)$. If only a part of it is leaked to others, the others cannot deduce the complete data based on the obtained part, so the data cannot be leaked to others.*

Theorem 1. *The SFM protocol is secure in the presence of semi-honest (non-colluding) adversaries $\mathcal{A} = (\mathcal{A}_{MP}, \mathcal{A}_{CP}, \mathcal{A}_{CSP})$.*

Proof. Sim_{MP} receives $\langle x \rangle$, $\langle y \rangle$ as input, which consist of s_x, t_x, s_y, t_y, and then simulates that $\mathcal{A}_{MP}$ obtains $[\![s_x]\!]_{pk}$, $[\![t_x]\!]_{pk}$, $[\![s_y]\!]_{pk}$, $[\![t_y]\!]_{pk}$. Sim_{CP} and Sim_{CSP} simulate that $\mathcal{A}_{CP}$ and $\mathcal{A}_{CSP}$ calculate $\mathsf{SAD}([\![s_x]\!]_{pk}, [\![t_x]\!]_{pk})$ and $\mathsf{SM}([\![s_y]\!]_{pk}, [\![t_y]\!]_{pk})$. According to Lemma 1, it is trivial to see that the view of $\mathcal{A}_{MP}$, $\mathcal{A}_{CP}$, $\mathcal{A}_{CSP}$ consists of the encrypted data and execute SAD and SM in security. Correspondingly, the SDG protocol is also secure in the semi-honest model. Thus they are indistinguishable in the real and the ideal executions.

Theorem 2. *The SFAD protocol is secure in the presence of semi-honest (non-colluding) adversaries* $\mathcal{A} = (\mathcal{A}_{MP}, \mathcal{A}_{CP}, \mathcal{A}_{CSP})$.

Proof. Before performing SAD and SM, the views of CP and CSP will be $View_1 = (\langle x \rangle, \langle y \rangle, SG_1, SG_2)$ and $View_2 = (\langle x \rangle, \langle y \rangle, [\![SG_1]\!]_{pk}, [\![SG_2]\!]_{pk})$, respectively. $\mathcal{A}_{CP}$ will obtain SG_1, SG_2. Due to Lemma 2, result of SGT is safe for CP to know. It's trivial to see they are simulated by Sim_{MP}, Sim_{CP}, Sim_{CSP}. Since the security of the protocol SAD, SM execution have been proven above, it's trivial to prove the security of the whole protocol SFAD as well. Correspondingly, the SMOD protocol is also secure in the semi-honest model.

Theorem 3. *The SFDIV protocol is secure in the presence of semi-honest (non-colluding) adversaries* $\mathcal{A} = (\mathcal{A}_{MP}, \mathcal{A}_{CP}, \mathcal{A}_{CSP})$.

Proof. Due to Lemma 1, Lemma 2, Lemma 3, result of SGT and the digit of FPN' s_x or digit of FPN' s_y are safe for CP to know. An execution of SMAX, SM, SAD, SFAD will be $View_1^j = (\langle x_1 \rangle^j, [\![q_1^i]\!]_{pk}^j, \langle y_1 \rangle^j, [\![r_1]\!]_{pk}^j, [\![res_1]\!]_{pk}^j)$, $View_2^j = (\langle x_2 \rangle^j, [\![q_2^i]\!]_{pk}^j, \langle y_2 \rangle^j, [\![r_2]\!]_{pk}^j, [\![res_2]\!]_{pk}^j)$, and $1 \leq j \leq 9$. Since res_2^j of $View_2^j$ is intermediate result of part of FPN, according to Lemma 3, Lemma 1, Theorem 2, $\mathcal{A}_{MP}$, $\mathcal{A}_{CP}$, $\mathcal{A}_{CSP}$ will be computationally indistinguishable. Thus, SFDIV protocol is secure.

7 Experiment Analysis

In this section, we conduct real-world experiments to show the performance of the proposed for evaluating our PCTF. In PCTF, we implement SFAD, SFM, SFDIV, and SDG in our PCTF by Python. The Python program runs on a PC (Intel(R) Core(TM) i5-9500 CPU @ 3.00 GHz and 8.00 GB RAM) to simulate the multi-users with the different public keys to compute tasks on the cloud server. Meanwhile, we test the performance of PCTF the compared with [17], as shown in Fig. 2. The accuracy and efficiency are as follows:

- Accuracy: To implement multi-keyword floating division which has not been addressed in [17] and [18], we propose SFDIV to address the issues. In our SFDIV protocol, we approximate the fractional part of the FPN. Since the remainder obtained by the first division, the remainder obtained by the expansion after the division is very small, and the precision of floating point numbers that can be represented by Python can be ignored. Therefore, we consider the loss of SFDIV is vary small and the computation accuracy is very high. Meanwhile, in our PCTF, we can promise the precision of SFDIV at least in 10^{-17}. The SFM and SFAD in encrypting computation, it has no loss of accuracy.
- Efficiency: We evaluate the efficiency of our proposed PCTF from criteria: time cost overhead and scalability. The evaluation results show whether our model is efficient and practical. During this process, we can see (a), (b), (c) in Fig. 2 that our SFAD and SFM have less computation cost and less joint computing time, comparing with [18]. But in our SFDIV, we can see (d) and (g) in Fig. 2 that we can know that as the accuracy increases, the calculation

time required for the division also increases. When the divisor is encrypted and the dividend is an integer or encoding number in (d), (e), (f), (g), (h), (i), we can get the computation cost only in CP. This reduces the risk of ciphertext leakage.

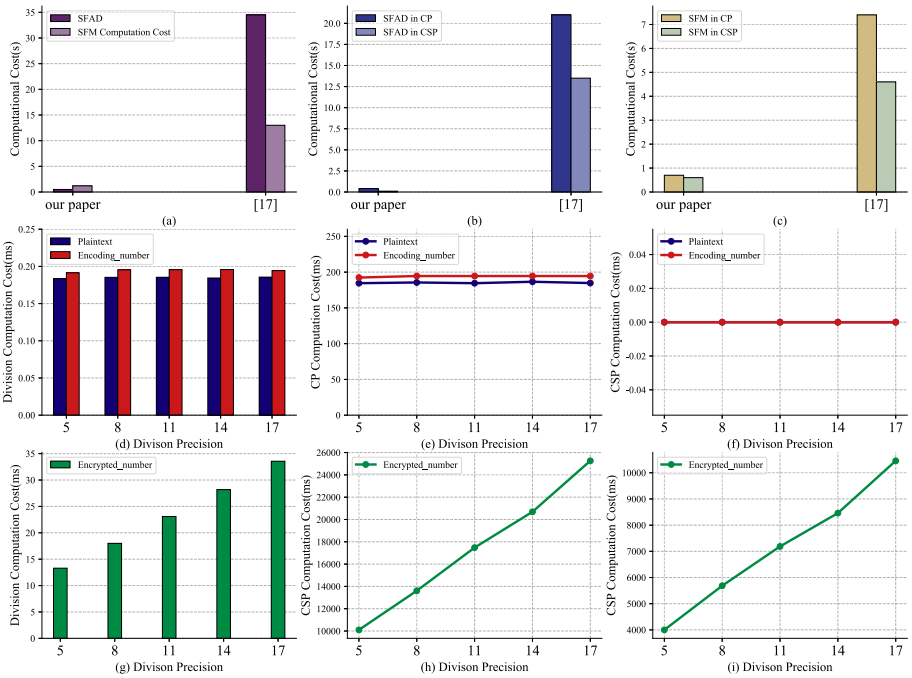

Fig. 2. Floating computational cost.

Table 3. The time cost of security protocol

Protocol	CP compute(s).	CSP(s) compute.	Total time(s)
SFAD	0.4323	0.1501	0.5823
SFM	0.8234	0.4139	1.2375
SFDIV	24.6281	9.9341	34.5623

As shown in Table 3, we calculate the FPN division in SFDIV, SFAD, and SFM. From the Fig. 2, we can see that our division time increases with the number of digits increasing, and it only takes 13 s when the accuracy is 10^{-5}, which is 4 times faster than the 50 s required in reference [18] times. With the increase of division accuracy, we can see that the time of the algorithm increases linearly. When dealing with some high-precision problems, our safe division can

still maintain high accuracy. After 20 iterations of secure division, our error is only kept at 10^{-15}. This shows that multiple iterations do not affect the performance of the SFDIV.

8 Conclusion

In this paper, we proposed PCTF, a framework for computation tookit on FPN, which allows users to outsource encrypted data to a cloud service provider for storing and processing. We build a privacy-preserving computation tookit of FPN. The toolkit offers multiplication, addition, subtraction, division with FPN. SFM, SFAD are trivial extensions of the integer operations. Meanwhile, We present new protocols for the division of FPN. Our evaluations demonstrated that our framework compared with other FPN protocols effectively reduces computational and communication. Protocols are provably secure in the cloud service with semi-honest behavior.

In the future work, we will focus on the optimization of communication overhead and the expansion of machine learning methods. Although the total time cost of the calculation process is not high in the current scheme, the communication time in CP is still worth optimizing. In addition, to cope with the expansion of machine learning algorithms in different scenarios, the future work will focus on applying in a specific application domain.

Acknowledgement. This work is supported by the National Natural Science Foundation of China (Grant No. U1705262, No. 62072109, No. U1804263), and the Natural Science Foundation of Fujian Province (Grant No. 2018J07005)

References

1. Mell, P., Grance, T.: The NIST definition of cloud computing (draft). NIST Special Publication 800-145 (2011)
2. Seagate Cor: Data age 2025 - the Evolution of Data to Life-Critical (2018). https://www.seagate.com/cn/zh/our-story/data-age-2025/. Accessed 2020
3. Chamberlin, B.: IoT (Internet of Things) Will go Nowhere Without Cloud Computing and Big Data Analytics. http://ibmcai.com/2014/11/20/iot-internet-of-things-will-go-nowhere-without-cloud-computing-and-big-data-analytics/
4. Yeshodara, N.S., Nagojappa, N.S., Kishore, N.: Cloud based self driving cars. In: 2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), Bangalore, India, pp. 1–7 (2014). https://doi.org/10.1109/CCEM.2014.7015485
5. Quick, D., Martini, B., Choo, R.: Cloud Storage Forensics. Elsevier, Amsterdam (2013)
6. Mohassel, P., Zhang, Y.: SecureML: a system for scalable privacy-preserving machine learning. In: IEEE Symposium on Security and Privacy (SP), IEEE 2017, pp. 19–38 (2017)
7. Wang, X., Ma, J., Miao, Y., Liu, X., Yang, R.: Privacy-preserving diverse keyword search and online pre-diagnosis in cloud computing. IEEE Trans. Serv. Comput. (2019)

8. Sun, W., Yu, S., Lou, W., Hou, Y.T., Li, H.: Protecting your right: attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. In: Proceedings of INFOCOM, pp. 226–234. IEEE (2014)
9. Bidi Ying, D.M., Mouftah, H.T.: Sink privacy protection with minimum network traffic in WSNs. Ad Hoc Sens. Wirel. Netw. **25**(1–2), 69–87 (2015)
10. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
11. Cloud Computing. https://en.wikipedia.org/wiki. Accessed 2019
12. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of 44th Annual ACM Symposium on Theory Computing, pp. 1219–1234 (2012)
13. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 36–54. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_3
14. Sanil, A.P., Karr, A.F., Lin, X., Reiter, J.P.: Privacy preserving regression modelling via distributed computation. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004), pp. 677–682. ACM (2004)
15. Lazzeretti, R., Barni, M.: Division between encrypted integers by means of garbled circuits. In: WIFS 2011, pp. 1–6 (2011)
16. Dahl, M., Ning, C., Toft, T.: On secure two-party integer division. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 164–178. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32946-3_13
17. Liu, X., Choo, K.-K.R., Deng, R.H., Lu, R., Weng, J.: Efficient and privacy-preserving outsourced calculation of rational numbers. IEEE Trans. Depend. Secur. Comput. **15**, 27–39 (2016)
18. Liu, X., Deng, R.H., Choo, K.-K.R., Weng, J.: An efficient privacy-preserving outsourced calculation toolkit with multiple keys. IEEE Trans. Inf. Forensics Secur. **11**(11), 2401–2414 (2016)
19. Veugen, T.: Encrypted integer division and secure comparison. IJACT **3**(2), 166–180 (2014)
20. Catrina, O., Saxena, A.: Secure computation with fixed-point numbers. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 35–50. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14577-3_6
21. Kiltz, E., Leander, G., Malone-Lee, J.: Secure computation of the mean and related statistics. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 283–302. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_16
22. Bogdanov, D., Niitsoo, M., Toft, T., et al.: High-performance secure multi-party computation for data mining applications. Int. J. Inf. Secur. **11**(6), 403–418 (2012)
23. Bunn, P., Ostrovsky, R.: Secure two-party k-means clustering. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28–31, 2007. ACM (2007)
24. Catrina, O., Dragulin, C.: Multiparty computation of fixed-point multiplication and reciprocal. In: DEXA 2009, no. 1, pp. 107–111 (2009)