This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing

Efficient and Privacy-Preserving Outsourced Calculation of Rational Numbers

Ximeng Liu, *Member, IEEE,* Kim-Kwang Raymond Choo, *Senior Member, IEEE,* Robert H. Deng, *Fellow, IEEE,* Rongxing Lu, *Senior Member, IEEE,* and Jian Weng

Abstract—In this paper, we propose a framework for efficient and privacy-preserving outsourced calculation of rational numbers, which we refer to as POCR. Using POCR, a user can securely outsource the storing and processing of rational numbers to a cloud server without compromising the security of the (original) data and the computed results. More specifically, we present a Paillier cryptosystem with threshold decryption (PCTD), the core cryptographic primitive, to reduce the private key exposure risk in POCR. We also present the toolkits required in the privacy preserving calculation of integers and rational numbers to ensure that commonly used outsourced operations can be handled on-the-fly. We then prove that the proposed POCR achieves the goal of secure integer and rational number calculation without resulting in privacy leakage to unauthorized parties, as well as demonstrating the utility and the efficiency of POCR using simulations.

Index Terms—Privacy-Preserving; Homomorphic encryption; Outsourced computation; Rational numbers; Encrypted data processing.

1 Introduction

THE increase in the number and volume of digital and Internet-connected devices (e.g. Internet of Things and medical devices) as well as the growing size of storage media have resulted in a significant increase in the amount of data captured, stored and disseminated in electronic only form [1], [2] – this is also known as 'big data'. A survey by IDC and EMC [3], for example, forecasted that the size of digital data created, replicated and consumed will be 40,000 exabytes by 2020. A widely accepted definition of big data is from Gartner, which defines it as "high-volume, -velocity and -variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making" [4].

Cloud computing has been identified as a potential solution to process and store big data, and is increasingly used in domains such as Internet of Things (IoT) [5], e-commerce [6], and scientific research [7], [8], [9]. For example, in 2011, the U.S Federal Government adopted a 'Cloud First' policy which requires government agency's Chief Information Officers to implement a cloud-based service whenever there was a secure, reliable, and cost-effective option [10], [11]. A review of seven government agencies by the U.S. Government Accountability Office in 2014 found that since 2012, "the total number of cloud computing services implemented by the agencies increased by 80 services, from 21 to 101" and these "agencies collectively reported cost savings of about $96 million from the implementation of 22 of the 101 cloud services" [12]. It is, therefore, unsurprising that research labs, such as [13], [14], dedicated to cloud computing research have also been set up.

One application of cloud is IoT (or Cloud of Things) where computationally limited devices, such as body sensors (used to monitor patient's heart rate, blood pressure and glucose levels, etc), can send data to the cloud for processing. There are known security and privacy concerns relating to the use of cloud computing. In the body sensor example, it is important to ensure the security and privacy of patient's health and other personally identifiable information (PII), such as health status. The accuracy of the collected data is also crucial in applications, such as healthcare. In healthcare, most data (e.g. blood glucose, insulin, and C-peptide levels) are non-integer (see the recent study involving 36,745 subjects aged 4069 in the Japan Public Health Center-based prospective study [15]).

However, traditional cryptosystems are generally designed to protect only integer values. Therefore, this will affect the accuracy of the data and consequency, decision making and in the worse case scenario, resulting in the wrong diagnosis of a patient.

In this paper, we seek to address the above-mentioned challenge by presenting a framework for efficient and **P**rivacy-preserving **O**utsourced **C**alculation for **R**ational numbers (POCR). We regard the contributions of this paper to be four-fold, namely:

- Our proposed POCR is designed to allow users to

X. Liu, and R.H. Deng are with the School of Information Systems, Singapore Management University, Singapore. E-mail:snbnix@gmail.com, robertdeng@smu.edu.sg.
K.K.-R. Choo is with (1) the School of Information Technology and Mathematical Sciences, University of South Australia, Australia, and (2) INTERPOL Global Complex for Innovation, Singapore. E-mail: Raymond.Choo@fulbrightmail.org.
R. Lu is with the School of Electrical and Electronics Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore. E-mail: rxlu@ntu.edu.sg.
J. Weng is with the School of Information Technology, Jinan University, China. E-mail: cryptjweng@gmail.com.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing

outsource their own data to cloud server for secure processing.
- We construct a new cryptographic primitive, referred to as Paillier cryptosystem with threshold decryption (PCTD), used in POCR to separate a private key into different shares in order to reduce the user's key leakage risk.
- We build a privacy-preserving outsourced calculation toolkit for integer numbers. The toolkit consists of most commonly used elementary operations, such as multiplication, division, comparison, sorting, equivalence testing and greatest common divisior.
- We build a privacy-preserving outsourced calculation toolkit for rational numbers, and a secure reducing fraction protocol designed to facilitate the server in reducing the greatest common divisor from both numerator and denominator in a privacy-preserving way.

The remainder of this paper is organized as follows: In Section 2, we describe the preliminaries required in the understanding of our proposed POCR. In Section 3, we formalize the system model, as well as outlining the problem statement and the attacker model. Then, we present the PCTD and both privacy-preserving outsourced calculation toolkits for integer and rational numbers in Section 5. The security analysis and performance evaluation are presented in Sections 6 and 7, respectively. Related work is discussed in Section 8. Section 9 concludes this paper.

2 Preliminary

In this section, we outline the definitions of the Additive Homomorphic Cryptosystem and the Secure Bit-Decomposition Protocol, which serve as the building blocks of the proposed POCR. Table 1 lists the key notations used throughout this paper. For ease of reading, if all ciphertexts belong to a specific RU, say a, we will simply use $[x]$ instead of $[x]_{pk_a}$.

TABLE 1
Notations used

Notations	Definition
pk_a/ sk_a	Public-private key pair of RU a
$sk_a^{(i)}$	Partially private keys
$[x]_{pk_a}$	Encrypted data x under pk_a
$D_{sk_a}(\cdot)$	Decryption algorithm using sk_a
$PD_{sk^{(i)}}(\cdot)$	Partially decryption algorithm (PDec) using $sk^{(i)}$
$\|x\|$	Bit length of x
$gcd(x, y)$	Greatest common divisor between x and y
$a \cdot b$	Multiplication between a and b over cyclic group
RSM	Revised secure multiplication protocol
SLT	Secure less than protocol
SMMS	Secure maximum and minimum sorting protocol
SEQ	Secure equivalent testing protocol
SDIV	Secure division protocol
SGCD	Secure greatest common divisor protocol
SRF	Secure Reducing Fraction Protocol

2.1 Additive Homomorphic Cryptosystem (AHC)

Suppose that $[m_1]$ and $[m_2]$ are two additive homomorphic ciphertexts under the same public key pk in an additive homomorphic cryptosystem. The additive homomorphic cryptosystem (e.g. Paillier cryptosystem [16] and Benaloh cryptosystem [17]) has the **additive homomorphism** property:

$$D_{sk}([m_1] \cdot [m_2]) = m_1 + m_2. \quad (1)$$

2.2 Secure Bit-Decomposition Protocol (SBD)

Suppose that there are two parties in the protocol, Alice and Bob. Bob holds the AHC encrypted value $[x]$, where $0 \leq x < 2^{\mu}$ and μ is the domain size of x in bits. We also remark that x is not known to both Alice and Bob. Let $(x_0, \cdots, x_{\mu-1})$ denote the binary representation of x, where x_0 and $x_{\mu-1}$ are the least and most significant bits, respectively. The goal of **SBD** is to convert the encryption of x into the encryption of the individual bits of x, without disclosing any information regarding x to both parties. More formally, we define the **SBD** protocol as follows:

$$\langle [x_0], \cdots, [x_{\mu-1}] \rangle \leftarrow \textbf{SBD}([x]).$$

We refer the interested reader to [18] for the detailed construction of the **SBD** protocol.

3 System Model & Privacy Requirement

In this section, we formalize the POCR system model, outline the problem statement, and define the attack model.

3.1 System Model

In our system, we mainly focus on how the cloud server responds to user requests in a privacy-preserving manner. The system comprises a Key Generation Center (KGC), a Cloud Platform (CP), Computation Service Providers (CSPs), and Request Users (RUs) - see Fig. 1.

3.1.1 Key Generation Center (KGC)

The trusted KGC is tasked with the distribution and management of private keys in the system.

3.1.2 Request Users (RUs)

Generally, a RU will use its public key to encrypt some data, before storing the encrypted data with a CP. The RU can also request a CP to perform some calculations over the outsourced data.

3.1.3 Cloud Platform (CP)

A CP has 'unlimited' data storage space, and stores and manages data outsourced from all registered RUs. A CP also stores all intermediate and final results in encrypted form. Furthermore, a CP is able to perform certain calculations over encrypted data.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing

3.1.4 Computation Service Providers (CSPs)

The CSPs provides online computation services to RUs. Also, CSPs are able to partially decrypt ciphertexts sent by the CP, perform certain calculations over the partially decrypted data, and then re-encrypt the calculated results.

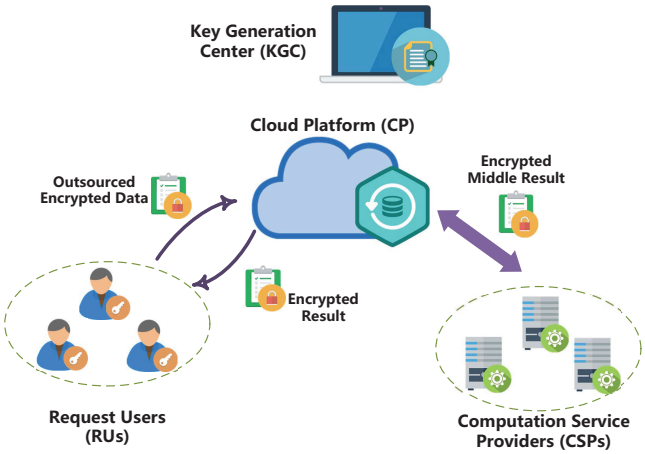


Fig. 1. System model under consideration

3.2 Problem Statement

Consider a database T which contains β records $(1 < i < \beta)$, where x_i is a rational number belonging to a RU (e.g. insulin level). Data should be encrypted prior to outsourcing to a CP for storage. The RU can launch a query to the CP to obtain some statistic information about T at will. For example, the RU can query for the mean, $\bar{x} = \sum_{i=1}^{\beta} x_i/\beta$, and the variance, $d_j = \sum_{i=1}^{\beta}(x_i - \bar{x})^2/\beta$. As x_i is a rational number and needs to be encrypted during the calculation, we have the following challenges:

1) **Secure Rational Number Storage:** Traditional encryption method can only encrypt the number over a finite field (positive integer number & zero (in an additive group)). Therefore, we need to be able to store rational numbers without compromising the privacy of the data owner (RU).

2) **Secure Integer Operations:** The encrypted integer calculation toolkit should be built first to support commonly used operations. For example, integer number operations, such as additions, multiplications and divisions over plaintext, should be achievable by operating on these two encrypted numbers.

3) **Secure Rational Number Processing:** In order to support outsourced rational number processing, the toolkit for secure rational number calculations (e.g. comparison of encrypted rational numbers) needs to be constructed. Moreover, as the plaintext size increases with the executing time of the homomorphic operations, this may lead to an error in the results (see Section 5.1 for the detailed analysis). Therefore, some mechanisms should be designed to guarantee the correctness of the results after homomorphic operations.

3.3 Attack Model

In our attack model, the KGC (a trusted entity) generates the public keys and private keys for the system. On the other hand, RUs, CP and CSPs are *curious-but-honest* parties, which strictly follow the protocol, but they are also interested to learn data belonging to other parties. Therefore, we introduce an active adversary $\mathcal{A}^*$ in our model. The goal of $\mathcal{A}^*$ is to decrypt the challenge RU's ciphertext with the following capabilities:

1) $\mathcal{A}^*$ may eavesdrop all communication links to obtain the encrypted data.

2) $\mathcal{A}^*$ may compromise the CP in order to guess the plaintext value of all ciphertexts outsourced from the challenge RU, and all ciphertexts sent from the CSP by executing an interactive protocol.

3) $\mathcal{A}^*$ may compromise several CSPs to guess the plaintext value of all ciphertexts sent from the CP by executing an interactive protocol.

4) $\mathcal{A}^*$ may compromise RUs, with the exception of the challenge RU, to get access to their decryption capabilities, and try to guess all ciphertexts belonging to the challenge RU.

However, $\mathcal{A}^*$ is restricted from compromising (1) both the CSPs and the CP concurrently, and (2) the challenge RU. We remark that such restrictions are typical in adversary models used in cryptographic protocols (see the review of adversary models in [19]).

4 Crypto Primitive and Privacy Preserving Integer Calculation Toolkits

4.1 Paillier Cryptosystem with Threshold Decryption (PCTD)

In order to realize POCR, the Paillier-based cryptosystem [20] appears to be a suitable solution for our system at first glance. However, the RU is not able to directly send the private key to the servers and the server can use the private key to obtain the corresponding user's data squarely. Therefore, we adapt the Paillier-based cryptosystem to separate private key into different shares to support (k, n) threshold decryption. This new system is, thereafter, referred to as the Paillier Cryptosystem with Threshold Decryption (PCTD), which consists of the following algorithms:

KeyGen: Let $\mathfrak{O}$ be the security parameter and p, q be two large prime numbers, where $|p| = |q| = \mathfrak{O}$. Due to the property of the strong primes, we have two strong primes p', q', s.t., $p' = \frac{p-1}{2}$ and $q' = \frac{q-1}{2}$. We then compute $N = pq$ and $\lambda = lcm(p-1, q-1)/2$, and choose a generator g of order $(p-1)(q-1)/2$. To choose the generator, we first select a random number $a \in \mathbb{Z}^*_{N^2}$ before computing $g = -a^{2N}$ [21] (for simplicity, we denote $g = 1 + N$). The public key is then $pk = N$, and the corresponding private key is $sk = \lambda$.

Encryption (Enc): Given a message $m \in \mathbb{Z}_N$, we choose a random number $r \in \mathbb{Z}_N$. The ciphertext can be generated as

$$[m] = g^m \cdot r^N \mod N^2 = (1 + mN) \cdot r^N \mod N^2.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing

Decryption (Dec): To decrypt $[m]$ using the decryption algorithm $D_{sk}(\cdot)$ and the corresponding private key $sk = \lambda$, we need to compute

$$[m]^{\lambda} = r^{\lambda N}(1 + mN\lambda) \mod N^2 = (1 + mN\lambda).$$

Since $gcd(\lambda, N) = 1$, m can be recovered as:

$$m = L(T_1^{\lambda} \mod N^2)\lambda^{-1} \mod N.^1$$

Private Key Splitting (KeyS): choose δ, s.t., $\delta \equiv 0 \mod \lambda$ and $\delta \equiv 1 \mod N^2$ hold at the same time[2]. Define a polynomial $q(x) = \delta + \sum_{i=1}^{k-1} a_i x^i$, where $a_1, \cdots, a_{k-1}$ are $k-1$ random number from $\mathbb{Z}^*_{\lambda N^2}$. Let $\alpha_1, \cdots, \alpha_n \in \mathbb{Z}^*_{\lambda N^2}$ be n distinct non-zero elements known to all the parties. Denote $sk^{(i)} = q(\alpha_i)$ and send to party i.

Partially decryption (PDec): Once $[m]$ is received, with partially private key $sk^{(i)} = q(\alpha_i)$, the partially decrypted ciphertext $CT^{(i)}$ can be calculated as:

$$CT^{(i)} = [m]^{q(\alpha_i)} \mod N^2.$$

Threshold decryption (TDec): Once d ($d \geq k$) partially decrypted ciphertext $CT^{(\tau_1)}, \cdots, CT^{(\tau_d)}$ are received, the **TDec** algorithm can choose an arbitrary k-element subset S,[3] and calculate

$$T'' = \prod_{l \in S} (CT^{(l)})^{\Delta_{l,S}(0)} \mod N^2.$$

where $\Delta_{l,S}(x) = \prod_{j \in S, j \neq l} \frac{x - \alpha_j}{\alpha_l - \alpha_j}$, and then calculates $m = L(T'')$.

Ciphertext Refresh (CR): Once $[m]$ is received, the *CR* algorithm can refresh the ciphertext without changing the original message m, by randomly choosing $r' \in \mathbb{Z}_N$ and calculating

$$[m]' = [m] \cdot r'^N = (r \cdot r')^N (1 + mN) \mod N^2. \quad \square$$

Note that our PCTD satisfies formula (1). Moreover, given $m \in \mathbb{Z}_N$, we have

$$\begin{aligned}[m]^{N-1} &= (1 + (N-1)m \cdot N)) \cdot r^{N-1 \cdot N} \mod N^2 \\ &= [-m].\end{aligned}$$

The KGC generates the private key sk and public key pk for a RU, and randomly divides sk into n partially private keys $sk^{(1)}, \cdots, sk^{(n)}$ before sending $sk^{(1)}$ to the CP, and $sk^{(i)}$ and pk to the CSP$_i$ ($i = 2, \cdots, n$). The RU can encrypt data using pk and outsource the ciphertexts to the CP for storage.

After introducing the underlying algorithms in PCTD, we will now present the secure sub-protocols as the toolkits for processing integers, namely: Revised Secure Multiplication Protocol (**RSM**), Secure Less Than Protocol (**SLT**), Secure Maximum and Minimum Sorting Protocol (**SMMS**), Secure Equivalent Testing Protocol (**SEQ**), Secure Division Protocol (**SDIV**) and Secure Greatest Common Divisor Protocol (**SGCD**). We assume that a CP and $t_c - 1$ ($t_c > 1, k \geq 2$) online CSPs (denote as CSP 2, CSP 3,$\cdots$, CSP t_c, respectively) will be involved in these sub-protocols. One online CSP (called CSP$_\gamma$) is chosen from these CSPs in order to handle extra calculations. Note that the integers x, y involved in the above sub-protocols can be positive, negative or zero unless otherwise stated. Therefore, we should restrict x, y to be in the range of $[-R_1, R_1]$, where $|R_1| < |N|/4 - 1$.[4] If we need a larger plaintext range, we can simply use the larger N. A larger N implies a broader plaintext range, and therefore, a higher level of security. However, this will affect the efficiency of the PCTD (See Fig. 2(a)).

4.2 Revised Secure Multiplication Protocol (RSM)

As our PCTD can only support additive homomorphism, we are unable to achieve multiplication between the two plaintexts. In order to allow plaintext multiplication, we revise the original Secure Multiplication (**SM**) protocol [23], and present the revised protocol, Revised Secure Multiplication Protocol (**RSM**). When the CP is given two encrypted data $[x]$ and $[y]$ as input, the **RSM** will securely compute $[x \cdot y]$:

Step-1(@CP): The CP selects two random numbers $r_x, r_y \in \mathbb{Z}_N$, calculates $X = [x] \cdot [r_x]$, $Y = [y] \cdot [r_y]$, $X_1 = PD_{sk^{(1)}}(X)$, and $Y_1 = PD_{sk^{(1)}}(Y)$, sends X and Y to all online CSPs, X_1 and Y_1 to the CSP$_\gamma$.

Step-2(@CSP$_i$)[5]: The online CSP$_i$ calculates

$$CT_x^{(i)} = PD_{sk^{(i)}}([x]) \text{ and } CT_y^{(i)} = PD_{sk^{(i)}}([y]),$$

and send them to CSP$_\gamma$.

Step-3(@CSP$_\gamma$): Once the partially decrypted ciphertext is received, CSP$_\gamma$ uses **TDec** to decrypt X and Y, and obtains x' and y', respectively. Then, CSP$_\gamma$ calculates $h = x' \cdot y'$, encrypts h using pk (denoted as $H = [h]$), and sends H to the CP. It can be easily verified that $h = (x + r_x)(y + r_y)$.

Step-4(@CP): Once H is received, the CP computes $S_1 = [r_x \cdot r_y]^{N-1}$, $S_2 = [x]^{N - r_y}$ and $S_3 = [y]^{N - r_x}$. Then, the CP uses the following formula:

$$H \cdot S_1 \cdot S_2 \cdot S_3 = [h - r_y \cdot x - r_x \cdot y - r_x \cdot r_y] = [x \cdot y].$$

Therefore, both CP & CSPs can jointly compute $[x \cdot y]$.

4.3 Secure Less Than Protocol (SLT)

Given two encrypted numbers $[x]$ and $[y]$, the **SLT** protocol will provide the encrypted data $[u^*]$, which can be used to determine the relationship between the plaintexts of the two encrypted data (i.e. $x < y$ or $x \geq y$). The **SLT** is described as follows:

1. Define function $L(x)$ as $L(x) = \frac{x-1}{N}$.

2. Since $gcd(\lambda, N^2) = 1$, according to the Chinese remainder theorem [22], then $\delta = \lambda \cdot (\lambda^{-1} \mod N^2) \mod \lambda N^2$.

3. Note that $\tau_1, \cdots, \tau_d$ are mutually disjoint numbers and are belonged to $\{1, \cdots, n\}$. For $\forall l \in S$, we have $l \in \{\tau_1, \cdots, \tau_d\}$

4. As the plaintext in PCTD ranges from $[0, N-1]$ and is modular N, $[N - R_1, N]$ is used to represent the range $[-R_1, 0]$.

5. Note $i = 2, \cdots, t_c$, include γ, i.e., for all online CSPs. It is same for the following protocols unless otherwise specified.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing

Step-1(@CP): (1) The CP computes

$$[x_1] = [x]^2 \cdot [1] = [2x+1];\ [y_1] = [y]^2 = [2y].$$

(2) The CP flips a coin s and chooses a random number $r \in \mathbb{Z}_N$. If $s = 1$, the CP computes

$$[l] = ([x_1] \cdot [y_1]^{N-1})^r = [r(x_1 - y_1)].$$

Otherwise, the CP computes

$$[l] = ([y_1] \cdot [x_1]^{N-1})^r = [r(y_1 - x_1)].$$

(3) Since the CP knows $sk^{(1)}$, it can compute $K = PD_{sk^{(1)}}([l])$, prior to sending $[l]$ and K to all CSPs and CSP$_\gamma$, respectively.

Step-2(@CSP$_i$): The online CSP$_i$ calculates

$$CT^{(i)} = PD_{sk^{(i)}}([l])$$

and send it to CSP$_\gamma$.

Step-3 (@CSP$_\gamma$): The CSP will execute **TDec** to obtain l. If $|l| > |N|/2$, then CSP marks $u' = 1$, and $u' = 0$ otherwise. The CSP uses the **Enc** algorithm to encrypt u', and sends $[u']$ back to the CP.

Step-4 (@CP): Once $[u']$ is received, the CP computes as follows: if $s = 1$, CP marks $[u^*] = \mathbf{CR}([u'])$, otherwise

$$[u^*] = [1] \cdot [u']^{N-1} = [1 - u'].$$

We remark that $u^* = 0$ indicates $x \geq y$, and $u^* = 1$ indicates $x < y$.

4.4 Secure Maximum and Minimum Sorting Protocol (SMMS)

Given two encrypted numbers $[x]$ and $[y]$, the **SMMS** protocol will provide the encrypted sorting results $[A]$ and $[I]$, s.t., $A \geq I$. The **SMMS** is described as follows:

Step-1 (@CP): (1) CP computes

$$[x_1] = [x]^2 \cdot [1] = [2x+1];\ [y_1] = [y]^2 = [2y].$$

(2) The CP flips a coin s and chooses a random number $r, r_1^*, r_2^* \in \mathbb{Z}_N$. If $s = 1$, the CP computes

$$[l_1] = ([x_1] \cdot [y_1]^{N-1})^r = [r(x_1 - y_1)];$$

$$[l_2] = [y] \cdot [r_1^*] \cdot [x]^{N-1} = [y - x + r_1^*];$$

$$[l_3] = [x] \cdot [r_2^*] \cdot [y]^{N-1} = [x - y + r_2^*].$$

If $s = 0$, the CP computes

$$[l_1] = ([y_1] \cdot [x_1]^{N-1})^r = [r(y_1 - x_1)];$$

$$[l_2] = [x] \cdot [r_1^*] \cdot [y]^{N-1} = [x - y + r_1^*];$$

$$[l_3] = [y] \cdot [r_2^*] \cdot [x]^{N-1} = [y - x + r_2^*].$$

(3) The CP will then use $sk^{(1)}$ to calculate $K_1 = PD_{sk^{(1)}}([l_1])$, and send the K_1, $[l_2]$ and $[l_3]$ to the CSP$_\gamma$, and send $[l_1]$ to all the online CSPs.

Step-2(@CSP$_i$): The online CSP$_i$ calculates

$$CT_1^{(i)} = PD_{sk^{(i)}}([l_1])$$

and send it to CSP$_\gamma$.

Step-3 (@CSP$_\gamma$): The CSP$_\gamma$ will use **TDec** to obtain l_1. If $|l_1| < |N|/2$, the CSP$_\gamma$ marks $u' = 0$ and computes $D_1 = [0]$ and $D_2 = [0]$. Otherwise, the CSP$_\gamma$ marks $u' = 1$, uses **CR** to re-randomize $[l_2]$ and $[l_3]$, and denotes them as D_1 and D_2, respectively. Moreover, the CSP$_\gamma$ uses pk to encrypt u', and sends $[u']$, D_1 and D_2 back to the CP.

Step-4 (@CP): Once $[u'], D_1, D_2$ is received, the CP computes as follows:
If $s = 1$, then CP calculates

$$[A] = [x] \cdot D_1 \cdot [u']^{N-r_1^*},\ [I] = [y] \cdot D_2 \cdot [u']^{N-r_2^*};$$

Otherwise ($s = 0$), CP computes

$$[A] = [y] \cdot D_1 \cdot [u']^{N-r_1^*},\ [I] = [x] \cdot D_2 \cdot [u']^{N-r_2^*}.$$

Rationale for transformations in SLT and SMMS: In Step-1(1), both original data x and y need to be transformed into x_1 and y_1, in order to avoid revealing an equivalence relationship with the CSPs. For example, if $x = y = 5$, then $[r(x-y)] = [0]$ will be sent to CSPs for decryption. The CSP$_\gamma$ can easily determine $x = y$ if the decryption result is equal to 0. Therefore, to avoid such a situation, we will use the transformation and obtain $x_1 = 11$, $y_1 = 10$, where $x_1 \neq y_1$.

4.5 Secure Equivalent Testing Protocol (SEQ)

Given two encrypted data $[x]$ and $[y]$, the **SEQ** will provide the encrypted result $[f]$ to determine whether the plaintext of the two encrypted data are equivalent (i.e. $x \stackrel{?}{=} y$). The **SEQ** is described as follows:

(1) Both CP and CSP jointly calculate

$$[u_1] \leftarrow \mathbf{SLT}([x],[y]);\ [u_2] \leftarrow \mathbf{SLT}([y],[x]);$$

$$[f_1^*] \leftarrow \mathbf{RSM}([1] \cdot [u_1]^{N-1}; [u_2]);$$

$$[f_2^*] \leftarrow \mathbf{RSM}([u_1]; [1] \cdot [u_2]^{N-1}).$$

(2) The CP calculates and outputs:

$$[f] = [u_1 \oplus u_2] = [f_1^*] \cdot [f_2^*].$$

If $f = 0$, then $x = y$; otherwise, $x \neq y$.

4.6 Secure Division Protocol (SDIV)

Given an encrypted numerator $[y]$ and an encrypted denominator $[x]$,[6] the **SDIV** will provide the encrypted quotient $[q]$ and encrypted remainder $[r]$, without compromising the privacy of data, s.t., $y = q \cdot x + r$ ($y \geq x \geq 0$). The **SDIV** is explained in **Algorithm 1**, and a brief description is given below.

In the event that the value of the denominator is 0, we will mark $x = 1$ and $y = 0$ (line 1-4) as

6. For efficiency, we can simply restrict the ranges of x and y to be within $[0, \mu]$, where $\mu \ll R_1$. For example, if $|N| = 1024$, then we can choose $|\mu| = 35$ which satisfies an overwhelming majority of application demands. In other words, μ is the domain size of the plaintext in bits.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing

we cannot simply abort the **SDIV**. Otherwise, the CP will know that $x = 0$ once the **SDIV** is aborted. We will now use the **SBD** to expand $[y']$ into encrypted bits, denoted as $([q_{\mu-1}], \cdots, [q_0])$ (line 5). Also, we use $([0], \cdots, [0])$ to initialize $([a_{\mu-1}], \cdots, [a_0])$ (line 6). Next, the following procedure will be executed μ-times: move $[a_{\mu-1}], \cdots [a_0], [q_{\mu-1}], \cdots, [q_0]$ by one position to the left (i.e. mark $[a_i] = [a_{i-1}]$ for $i = \mu - 1$ to 1). Then, mark $[a_0] = [q_{\mu-1}]$, and $[q_i] = [q_{i-1}]$ for $i = \mu - 1$ to 1 (line 8). The CP will now calculate $[a_{\mu-1}], \cdots, [a_0]$ and convert from binary to integer $[A]$ before comparing A with x' using the **SLT**. If $A < x'$, the **SDIV** will mark $q_0 = 0$; otherwise, the **SDIV** will mark $q_0 = 1$ and compute $A = A - x'$ (line 9-13).

After calculating μ times, the remainder r is the integer value of $(a_{\mu-1}, \cdots, a_0)$ while the quotient q is the integer value of $(q_{\mu-1}, \cdots, q_0)$.

Algorithm 1: SECURE DIVISION PROTOCOL (SDIV)

Input: Encrypted numerator $[y]$ and encrypted denominator $[x]$.
Output: Encrypted quotient $[q]$ and encrypted remainder $[r]$

1 Both CP and CSP jointly calculate $[f] \leftarrow$ **SEQ**$([x], [0])$.
2 The CP calculates $[1] \cdot [f]^{N-1} = [1 - f]$.
3 Then, both CP and CSP jointly calculate $[f \cdot x] \leftarrow$ **RSM**$([f], [x])$ and $[y'] = [f \cdot y] \leftarrow$ **RSM**$([f], [y])$.
4 The CP calculates
$$[x'] = [f \cdot x + (1 - f) \cdot 1] = [f \cdot x] \cdot [1 - f].$$
5 Both CP and CSP jointly execute **SBD**, s.t., $\langle [y_{\mu-1}], \cdots, [y_0] \rangle \leftarrow$ **SBD**$([y'])$ and mark $([q_{\mu-1}], \cdots, [q_0]) = ([y_{\mu-1}], \cdots, [y_0])$.
6 The CP also initializes
$$([a_{\mu-1}], \cdots, [a_0]) = (\underbrace{[0], \cdots, [0]}_{\mu-\text{elements}}).$$
7 **for** *executing* μ *times* **do**
8 denote $[a_i] \leftarrow [a_{i-1}]$ (for $i = \mu$ to 1); then denote $[a_0] \leftarrow [q_{\mu-1}]$; finally, denote $[q_i] \leftarrow [q_{i-1}]$(for $i = \mu$ to 1);
9 calculate $[A] = [a_0] \cdot [a_1]^2 \cdots [a_{\mu-1}]^{2^{\mu-1}}$;
10 calculate $[Q] \leftarrow$ **SLT**$([A]; [x'])$;
11 calculate $[q_0] = [1] \cdot [Q]^{N-1} = [1 - Q]$;
12 execute $[B] \leftarrow$ **RSM**$([x']^{N-1}, [q_0])$;
13 calculate $[A] \leftarrow [A] \cdot [B]$ and then execute **SBD** protocol as:
$$\langle [a_{\mu-1}], \cdots [a_0] \rangle \leftarrow \textbf{SBD}([A]);$$
14 Finally, calculate
$$[r] = [a_0] \cdot [a_1]^2 \cdots [a_{\mu-1}]^{2^{\mu-1}}; [q] = [q_0] \cdot [q_1]^2 \cdots [q_{\mu-1}]^{2^{\mu-1}}.$$

4.7 Secure Greatest Common Divisor Protocol (SGCD)

Given two encrypted numbers $[x]$ and $[y]$ $(x > 0, y > 0)$[7], the **SGCD** protocol will provide the encrypted greatest common divisor $[C]$, without compromising the privacy of data. The **SGCD** is explained in **Algorithm 2**, and a brief description is given below.

Prior to calculating the greatest common divisor (GCD), the CP needs to determine which of the two plaintext values (i.e. $[x]$ and $[y]$) is larger, as the larger value will be chosen as the numerator and the smaller value as the denominator to run the **SDIV** (line 1). Next, in order to calculate GCD privately, we revisit the Euclidean algorithm: the GCD of two numbers does not change if the larger number is substituted with the difference between the two numbers. Since this substitution reduces the larger of the two numbers, repeating this process gives successively smaller pairs of numbers until one of the two numbers reaches zero. However, we are not able to directly use the Euclidean algorithm as it is, without leaking information since the adversary will know how many protocol rounds have been executed (e.g. the adversary can know the two integers are co-prime since only two rounds of calculation are made). Therefore, we will run the Euclidean algorithm for fixed μ rounds (related to the domain size of the integer). Unfortunately, the denominator will be equal to zero if the number of calculation rounds is fixed. This issue has been resolved by the **SDIV** (as explained in Section 4.6). After running the fixed calculation rounds, the CP obtains $\mu + 1$ encrypted remainder. The GCD is the last non-zero remainder. We just need to determine the first zero value remainder, and use the zero remainder to find the GCD. The idea is easy to follow: we denote the non-zero remainder as 1 and the zero remainder as 0 (line 6-7). We use the XOR operations to find the position of the last non-zero remainder (line 9-11) – see **Algorithm 2**.

5 TOOLKITS FOR PRIVACY PRESERVING CALCULATION OF RATIONAL NUMBERS

If a RU wants to outsource the rational numbers to the CP for storage, the three challenges are the need to be solved: 1) encrypt the rational numbers before outsourcing; 2) allow some calculations over two encrypted rational numbers; 3) guarantee the correctness of the results after fixed rounds of homomorphic operations.

As any rational number x can be presented using $x^{\uparrow}/x^{\downarrow}$, the first challenge can be easily solved by encrypting the numerator $x^{\uparrow}$ and denominator $x^{\downarrow}$, and storing $([x^{\uparrow}], [x^{\downarrow}])$. For example, -0.25 can be represented as $-1/4 = x^{\uparrow}/x^{\downarrow}$. Using the PCTD scheme, we encrypt $x^{\uparrow}$ and $x^{\downarrow}$ as $[x^{\uparrow}] = [1]^{N-1} = [-1]$ and $[x^{\downarrow}] = [4]$. After that, $([x^{\uparrow}], [x^{\downarrow}])$ is outsourced to the CP.

7. Mathematically, we only consider the greatest common divisor (GCD) between both positive integers.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing

Algorithm 2: SECURE GREATEST COMMON DIVISOR PROTOCOL (SGCD)

Input: two encrypted numbers $[x]$ and $[y]$.
Output: the encrypted greatest common divisor $[C]$.

Both CP and CSP jointly execute **SMMS**, s.t., $([y'], [x']) \leftarrow \mathbf{SMMS}([x], [y])$;
for $i = 1$ *to* μ **do**
  calculate $([q_i], [r_i]) \leftarrow \mathbf{SDIV}([y'], [x'])$;
  denote $[y'] \leftarrow [x']$ and $[x'] \leftarrow [r_i]$;
denote $[r_0] \leftarrow [q_1]$;
for $i = 0$ *to* μ **do**
  calculate $[u_i] \leftarrow \mathbf{SEQ}([r_i], [0])$;
for $i = 1$ *to* μ **do**
  execute $[f^*_{i-1,i}] \leftarrow \mathbf{RSM}([1] \cdot [u_{i-1}]^{N-1}; [u_i])$;
  execute $[f^*_{i,i-1}] \leftarrow \mathbf{RSM}([u_{i-1}]; [1] \cdot [u_i]^{N-1})$;
  calculate $[f_{i-1,i}] = [u_{i-1} \oplus u_i] = [f^*_{i,i-1}] \cdot [f^*_{i-1,i}]$;
  $[C_i] \leftarrow \mathbf{RSM}([r_{i-1}]; [f_{i-1,i}])$;
calculate and return $[C] = \prod_{j=1}^{m} [C_i]$.

To solve the second challenge, we introduce the encrypted rational numbers operations. However, we need to restrict $x^{\uparrow}$ to be in the range of $[-R_2, R_2]$ while $x^{\downarrow}$ to be $(0, R_2]$ in the following operations, where $|R_2| < |N|/8 - 1$.[8]

5.1 Encrypted Rational Number Calculation

In order to achieve encrypted rational number calculation, we provide the construction of the following seven operations.

Encrypted Rational Number Addition Operation: given two encrypted rational numbers $([x^{\uparrow}], [x^{\downarrow}])$ and $([y^{\uparrow}], [y^{\downarrow}])$, the calculated rational addition result is $([z^{\uparrow}], [z^{\downarrow}])$, where

$$[k_1] \leftarrow \mathbf{RSM}([x^{\uparrow}], [y^{\downarrow}]); [k_2] \leftarrow \mathbf{RSM}([x^{\downarrow}], [y^{\uparrow}]);$$
$$[z^{\uparrow}] = [k_1] \cdot [k_2]; [z^{\downarrow}] \leftarrow \mathbf{RSM}([x^{\downarrow}], [y^{\downarrow}]).$$

In other words,

$$([z^{\uparrow}], [z^{\downarrow}]) \leftarrow ([x^{\uparrow}], [x^{\downarrow}]) \boxplus ([y^{\uparrow}], [y^{\downarrow}]).$$

Encrypted Rational Number Minus Operation: given two encrypted rational numbers $([x^{\uparrow}], [x^{\downarrow}])$ and $([y^{\uparrow}], [y^{\downarrow}])$, the calculated rational minus result is $([z^{\uparrow}], [z^{\downarrow}])$, where

$$[k_1] \leftarrow \mathbf{RSM}([x^{\uparrow}], [y^{\downarrow}]); [k_2] \leftarrow \mathbf{RSM}([x^{\downarrow}], [y^{\uparrow}]);$$
$$[z^{\uparrow}] = [k_1] \cdot [k_2]^{N-1}; [z^{\downarrow}] \leftarrow \mathbf{RSM}([x^{\downarrow}], [y^{\downarrow}]).$$

Encrypted Rational Number Multiplication Operation: given two encrypted rational numbers $([x^{\uparrow}], [x^{\downarrow}])$ and $([y^{\uparrow}], [y^{\downarrow}])$, the rational number multiplication result is $([z^{\uparrow}], [z^{\downarrow}])$, where

$$[z^{\uparrow}] \leftarrow \mathbf{RSM}([x^{\uparrow}], [y^{\uparrow}]); [z^{\downarrow}] \leftarrow \mathbf{RSM}([x^{\downarrow}], [y^{\downarrow}]).$$

8. For efficiency, we can simply restrict the range of the numerator to be within $[-\mu, \mu]$ and the denominator to be within $(0, \mu]$, where $\mu \ll R_2$.

In other words,

$$([z^{\uparrow}], [z^{\downarrow}]) \leftarrow ([x^{\uparrow}], [x^{\downarrow}]) \boxtimes ([y^{\uparrow}], [y^{\downarrow}]).$$

Encrypted Rational Number Division Operation: given two encrypted rational numbers $([x^{\uparrow}], [x^{\downarrow}])$ and $([y^{\uparrow}], [y^{\downarrow}])$, the rational number division result is $([z^{\uparrow}], [z^{\downarrow}])$, where

$$[z^{\uparrow}] \leftarrow \mathbf{RSM}([x^{\uparrow}], [y^{\downarrow}]); [z^{\downarrow}] \leftarrow \mathbf{RSM}([x^{\downarrow}], [y^{\uparrow}]).$$

In other words,

$$([z^{\uparrow}], [z^{\downarrow}]) \leftarrow ([x^{\uparrow}], [x^{\downarrow}]) \circledast ([y^{\uparrow}], [y^{\downarrow}]).$$

Encrypted Rational Scalar-Multiplication Operation: given an encrypted rational numbers $([x^{\uparrow}], [x^{\downarrow}])$, the calculated result is $([z^{\uparrow}], [z^{\downarrow}])$, where

$$([z^{\uparrow}], [z^{\downarrow}]) = ([x^{\uparrow}]^k, [x^{\downarrow}]^k) = ([kx^{\uparrow}], [kx^{\downarrow}]).$$

Specifically, note that

$$([z^{\uparrow}], [z^{\downarrow}] = ([x^{\uparrow}]^{N-1}, [x^{\downarrow}]^{N-1}) = ([-x^{\uparrow}], [-x^{\downarrow}]).$$

Encrypted Rational Number Comparison Operation: given two encrypted rational numbers $([x^{\uparrow}], [x^{\downarrow}])$ and $([y^{\uparrow}], [y^{\downarrow}])$, the encrypted comparison result is $[u]$, where

$$[k_1] \leftarrow \mathbf{RSM}([x^{\uparrow}]; [y^{\downarrow}]); \ [k_2] \leftarrow \mathbf{RSM}([y^{\uparrow}]; [x^{\downarrow}]);$$
$$[u] \leftarrow \mathbf{SLT}([k_1]; [k_2]).$$

If $u = 0$, indicates $x \geq y$, and $u = 1$ indicates $x < y$.

Encrypted Rational Number Equivalent Testing Operation: given two encrypted rational numbers $([x^{\uparrow}], [x^{\downarrow}])$ and $([y^{\uparrow}], [y^{\downarrow}])$, the equivalent testing result can be stored and denoted as an encrypted data $[u]$, where

$$[k_1] \leftarrow \mathbf{RSM}([x^{\uparrow}]; [y^{\downarrow}]); \ [k_2] \leftarrow \mathbf{RSM}([y^{\uparrow}]; [x^{\downarrow}]);$$
$$[u] \leftarrow \mathbf{SEQ}([k_1]; [k_2]).$$

If $u = 0$, then indicates $x = y$; otherwise, $x \neq y$.

Example 1: Encrypted data, $([4^{\uparrow}], [15^{\downarrow}])$ and $([3^{\uparrow}], [20^{\downarrow}])$, are stored with the CP. If a RU wants to perform the rational number addition operations, CP & CSPs calculates:

$$([125^{\uparrow}], [300^{\downarrow}]) \leftarrow ([4^{\uparrow}], [15^{\downarrow}]) \boxplus ([3^{\uparrow}], [20^{\downarrow}]).$$

If the RU wants to perform the rational number multiplication operations, then CP & CSPs calculates:

$$([12^{\uparrow}], [300^{\downarrow}]) \leftarrow ([4^{\uparrow}], [15^{\downarrow}]) \boxtimes ([3^{\uparrow}], [20^{\downarrow}]).$$

If the RU wants to perform the rational number division operations, then CP & CSPs calculates:

$$([80^{\uparrow}], [45^{\downarrow}]) \leftarrow ([4^{\uparrow}], [15^{\downarrow}]) \circledast ([3^{\uparrow}], [20^{\downarrow}]).$$

As the results are not always in the simplest form, therefore, the plaintext length will increase with the number of the homomorphic operations. Suppose there are two t-bits length elements x_1 and x_2 ($t \ll |N|$), the homomorphic addition will obtain the $t+1$-bit (or t-bit)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing

length number $x_1 + x_1$. If the homomorphic multiplication is executed, we will obtain $2t$-bit (or $2t-1$-bit) x_1x_2 outputs. If too many homomorphic multiplications are involved, the length of the plaintext will easily be greater than $|N|$ and cause an error (the plaintext will modular N). For our rational number operations, the plaintext length of the numerator and denominator increases with the number of the homomorphic operations. However, the calculated numerator and denominator may exist some common divisors (in **Example 1**, 25 is the GCD between 125 and 300, while 12 is the GCD between 12 and 300, and 5 is the GCD between 80 and 45). Without taking any action, the middle results cannot be used after some fixed operations. In order to guarantee the correctness of the results, we should find the GCD between the numerator and denominator, and securely reduce the GCD from both numerator and denominator – see section 5.2.

5.2 Secure Reducing Fraction Protocol (SRF)

As too many homomorphic operations will result in an error, the Secure Reducing Fraction Protocol (**SRF**) is designed to reduce the GCD from both numerator and denominator, in order to guarantee the correctness of the results. Given $([x^{\uparrow}], [x^{\downarrow}])$, the **SRF** will output $([x_3^{\uparrow}], [x_3^{\downarrow}])$, s.t., $x^{\uparrow}/x^{\downarrow} = x_3^{\uparrow}/x_3^{\downarrow}$, and $x_3^{\uparrow}$ and $x_3^{\downarrow}$ has no common divisor. At first glance, it appears we can directly use the **SGCD** to obtain the GCD, and then use the **SDIV** to divide the GCD from numerator and denominator to get the simplest form. However, as we use $N-R$ to represent $-R$, an error will occur. For example, let $N = 23$, $x^{\uparrow} = -5$ and $x^{\downarrow} = 15$. Using PCTD, $x^{\uparrow} = -5$ is stored as $x^{\uparrow} = 18$. The direct calculation will result in an error value of $6/5$ (stored as $[6^{\uparrow}], [5^{\downarrow}]$), when the real value is $-1/5$ (stored as $[17^{\uparrow}], [5^{\downarrow}]$).

In order to guarantee the correctness, we introduce the following to handle the negative numbers: if $x^{\uparrow}$ is a positive value (i.e. $|x^{\uparrow}| < |N|/2$), then the value will not change. If $x^{\uparrow}$ is a negative value (i.e. $|x^{\uparrow}| > |N|/2$), the value will be replaced with $-x^{\uparrow}$. Then, the protocol uses **SGCD** to obtain GCD, and then use the **SDIV** to reduce $x^{\uparrow}$ and $x^{\downarrow}$ to obtain $x_2^{\uparrow}$ and $x_2^{\downarrow}$. If the original $x^{\uparrow}$ is a positive value, then the reduced value of $x_3^{\uparrow}$ will not be changed. If the original $x^{\uparrow}$ is a negative value, then the final result will be $-x_3^{\uparrow}$. The specific construction of the **SRF** is as follows:

Step-1(@CP): The CP flips a coin s and chooses a random number r, s.t., $|r| < 3|N|/8$, and calculates $[l] = ([x^{\uparrow}]^2 \cdot [1])^r = [r(2x^{\uparrow} + 1)]$ if $s = 1$, and $[l] = ([x^{\uparrow}]^2 \cdot [1])^{N-r} = [-r(2x^{\uparrow} + 1)]$ if $s = 0$. The CP then calculates $K = PD_{sk^{(1)}}([l])$, and sends K to the CSP$_\gamma$ and send $[l]$ to CSP$_i$.

Step-2(@CSP$_i$): The CSP$_i$ calculates

$$CT^{(i)} = PD_{sk^{(i)}}([l]),$$

and send it to CSP$_\gamma$.

Step-3(@CSP$_\gamma$): The CSP$_\gamma$ runs **TDec** to obtain l. If $|l| < |N|/2$, mark $u = 1$; otherwise, $u = -1$. Then, u is encrypted (i.e. $[u]$) and sent to CP.

Step-4: (1) If $s = 1$, then CP calculates $[x_1^{\uparrow}] \leftarrow$ **RSM**$([x^{\uparrow}]; [u])$; otherwise, $[x_1^{\uparrow}] \leftarrow$ **RSM**$([x^{\uparrow}]; [u]^{N-1})$.

(2) Both CP and CSP jointly run

$$[C] \leftarrow \textbf{SGCD}([x_1^{\uparrow}], [x^{\downarrow}]); ([Q_2], [x_2^{\uparrow}]) \leftarrow \textbf{SDIV}([x_1^{\uparrow}], [C]);$$

$$([Q_3], [x_3^{\downarrow}]) \leftarrow \textbf{SDIV}([x^{\downarrow}], [C]).$$

(3) If $s = 1$, then CP calculates $[x_3^{\uparrow}] \leftarrow$ **RSM**$([x_2^{\uparrow}]; [u])$; otherwise $[x_3^{\downarrow}] \leftarrow$ **RSM**$([x_2^{\downarrow}]; [u]^{N-1})$.

The protocol finally outputs $([x_3^{\uparrow}], [x_3^{\downarrow}])$. □

Example 2: Here we give an example to demonstrate the correctness of **SRF**. Suppose $N = 23$, $x^{\uparrow} = -5$ and $x^{\downarrow} = 15$. Thus, the CP stores the value as $([18^{\uparrow}], [15^{\downarrow}])$. Firstly, we should transform $([5^{\uparrow}], [15^{\downarrow}])$ using Step-1 to Step-3(1). The **SGCD** is then executed to generate the GCD $[5^{\uparrow}]$. Then, the **SDIV** is used to obtain $[1^{\uparrow}]$ and $[5^{\downarrow}]$. Finally, the CP will transfer the finial result as $([22^{\uparrow}], [5^{\downarrow}])$.

The Necessity of CSPs: Since PCTD (or other partially homomorphic cryptosystem) are used, we will need to use CSPs as auxiliary server to perform plaintext multiplication, as the CP is not able to perform both addition and multiplication homomorphic calculations over encrypted data at the same time (unlike, a fully homomorphic cryptosystem). Unfortunately, existing fully homomorphic cryptosystem is rather inefficient, in term of computation and storage [24], [25]. In the near future, if an efficient fully homomorphic cryptosystem exists, we can remove the CSPs from the system which will also result in a more elegant system.

The Extension to Handle Real Number: In our system, we use the nearest rational number to simulate the real number, at the cost of some accuracy. For example, if we want to store $\sqrt{2}$, we can just use 1.414 ($\frac{707}{500}$) to represent. If we want a higher level of accuracy, we can use 1.41421 ($\frac{141421}{100000}$) to represent $\sqrt{2}$. In other words, a higher level of accuracy will require a longer plaintext length.

6 Security Analysis

In this section, we will analyze the security of the basic encryption primitive and the sub-protocols, before demonstrating the security of our POCR framework.

6.1 Analysis of PCTD

6.1.1 The correctness of threshold decryption

The correctness of **TDec** can be verified as follows:

$$T'' = \prod_{l \in S} (CT^{(l)})^{\Delta_{l,S}(0)} \mod N^2 = [m]^{\sum_{l \in S} q(\alpha_i)\Delta_{l,S}(0)}$$
$$= ((1 + mN)r^N)^{\delta} \mod N^2 = 1 + mN.$$

Then, we can calculate $L(T'') = \frac{T''-1}{N} = m$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing

6.1.2 Security of PCTD

The security of our PCTD is given by the following theorem.

Theorem 1. *The* **PCTD** *scheme described in Section 4.1 is semantically secure, based on the assumed intractability of the Partially Discrete Logarithm (PDL) problem.*

Proof. The security of **PCTD** can be divided into two parts: 1) the privacy of ciphertext; 2) the privacy of divided private key.

The privacy of PCTD ciphertext follows directly from that of the Paillier cryptosystem, which has been proven to be semantically secure in the standard model assuming the intractability of the PDL problem [20] (the hardness of PDL problem can be found in [20]).

The privacy of divided private key is guaranteed by Shamir secret sharing scheme [26], [27] which is information-theoretic secure. The RU's private key sk is split into n shares in a way that any less than k shares cannot recover the original sk. It further implies that the adversary cannot cover the original plaintext with less than k shares of partially decrypted ciphertexts. □

6.2 The Security of Sub-protocols

Here we recall the security model for securely realizing an ideal functionality in the presence of non-colluding semi-honest adversaries. For simplicity, the challenge RU (a.k.a. "D_R"), and both CP (a.k.a. "S_P"), CSP$_i$ (a.k.a."S_i", $i \neq \gamma$), and CSP$_\gamma$ (a.k.a. "S_γ"), are involved in specific scenario of our functionality. We refer the reader to [28], [29] for the general case definitions.

Theorem 2. *The* **RSM** *protocol described in Section 4.2 can securely compute multiplication over ciphertext in the presence of semi-honest (non-colluding) adversaries* $\mathcal{A} = (\mathcal{A}_{D_R}, \mathcal{A}_P, \mathcal{A}_{S_i}, \mathcal{A}_{S_\gamma})$.

Proof. We only provide a proof to show how to construct the independent simulators $\mathsf{Sim}_{D_R}, \mathsf{Sim}_{S_P}, \mathsf{Sim}_{S_i}, \mathsf{Sim}_{S_\gamma}$.

Sim_{D_R} receives x and y as input and then simulates $\mathcal{A}_{D_R}$ as follows: it generates encryption $[x] = \mathbf{Enc}(x)$ of x and encryption $[y] = \mathbf{Enc}(y)$ of y. Finally, it returns $[x]$ and $[y]$ to $\mathcal{A}_{D_R}$ and outputs $\mathcal{A}_{D_R}$'s entire view.

The view of $\mathcal{A}_{D_R}$ consists of the encrypted data. The views of $\mathcal{A}_{D_R}$ in the real and the ideal executions are indistinguishable due to the semantic security of PCTD.

Sim_{S_P} simulates $\mathcal{A}_{S_P}$ as follows: First, it generates (ficitious) encryptions of the inputs $[\hat{x}]$ and $[\hat{y}]$ by running $\mathbf{Enc}(\cdot)$ on randomly chosen $\hat{x}, \hat{y}$, randomly generates $r_i \in Z_N$, calculates $\hat{X}$ and $\hat{Y}$, and then calculates $\hat{X}_1$ and $\hat{Y}_1$ by using $\mathbf{PDec}(\cdot)$. Sim_{S_P} sends the encryption $\hat{X}, \hat{Y}, \hat{X}_1, \hat{Y}_1$ to $\mathcal{A}_{D_P}$. If $\mathcal{A}_{D_P}$ replies with $\perp$, then Sim_{S_P} returns $\perp$.

The view of $\mathcal{A}_{S_P}$ consists of the encrypted data it creates. In both the real and the ideal executions, he receives the output the encryptions $\hat{X}, \hat{Y}, \hat{X}_1, \hat{Y}_1$. In the real world, this is guaranteed by the fact that the RU is honest and the semantic security of PCTD. The views of $\mathcal{A}_{S_P}$ in the real and the ideal executions are indistinguishable.

Sim_{S_i} simulates $\mathcal{A}_{S_i}$ as follows: it randomly choses x'' and y'', uses the $\mathbf{Enc}(\cdot)$ to get $[x'']$ and $[y'']$, use $\mathbf{PDec}$ to generate $CT^{(i)}_{x''}$ and $CT^{(i)}_{y''}$, and then sends these partially decrypted ciphertext to $\mathcal{A}_{S_i}$. If $\mathcal{A}_{S_i}$ replies with $\perp$, then Sim_{S_i} returns $\perp$.

The view of $\mathcal{A}_{S_i}$ consists of the encrypted data it creates. In both the real and the ideal executions, he receives the output the encryptions $CT^{(i)}_{x''}$ and $CT^{(i)}_{y''}$. In the real world, it is guaranteed by the semantic security of PCTD. The views of $\mathcal{A}_{S_\gamma}$ in the real and the ideal executions are indistinguishable.

Sim_{S_γ} simulates $\mathcal{A}_{S_\gamma}$ as follows: it randomly choses $\hat{h}$, uses the $\mathbf{Enc}(\cdot)$ to get $[\hat{h}]$, and then sends the encryptions to $\mathcal{A}_{D_\gamma}$. If $\mathcal{A}_{D_\gamma}$ replies with $\perp$, then Sim_{S_γ} returns $\perp$.

The view of $\mathcal{A}_{S_\gamma}$ consists of the encrypted data it creates. In both the real and the ideal executions, he receives the output the encryptions $[\hat{h}]$. In the real world, it is guaranteed by the semantic security of PCTD. The views of $\mathcal{A}_{S_\gamma}$ in the real and the ideal executions are indistinguishable. □

The security proof of **SLT** and **SMMS** protocols are similar to that of **RSM** protocol under the semi-honest (non-colluding) adversaries $\mathcal{A} = (\mathcal{A}_{D_R}, \mathcal{A}_P, \mathcal{A}_{S_i}, \mathcal{A}_{S_\gamma})$. In the following section, we need to prove the security of **SEQ**.

Theorem 3. *The* **SEQ** *protocol described in Section 4.5 is to securely evaluate the equivalence of plaintext over ciphertext in the presence of semi-honest (non-colluding) adversaries* $\mathcal{A} = (\mathcal{A}_{D_R}, \mathcal{A}_P, \mathcal{A}_{S_i}, \mathcal{A}_{S_\gamma})$.

Proof. We now demonstrate how to construct three independent simulators $\mathsf{Sim}_{D_R}, \mathsf{Sim}_{S_P}, \mathsf{Sim}_{S_i}, \mathsf{Sim}_{S_\gamma}$.

Sim_{D_R} receives x and y as input and simulates $\mathcal{A}_{D_R}$ as follows: it generates encryption $[x] = \mathbf{Enc}(x)$ of x and encryption $[y] = \mathbf{Enc}(y)$ of y. Finally, it returns $[x]$ and $[y]$ to $\mathcal{A}_{D_R}$ and outputs $\mathcal{A}_{D_R}$'s entire view.

The view of $\mathcal{A}_{D_R}$ consists of the encrypted data. The views of $\mathcal{A}_{D_R}$ in the real and the ideal executions are indistinguishable due to the semantic security of PCTD.

6.3 Security of POCR

Sim_{S_P} simulates $\mathcal{A}_{S_P}$ as follows: First, it generates (ficitious) encryptions of the inputs $[\hat{x}]$ and $[\hat{y}]$ by running $\mathbf{Enc}(\cdot)$ on randomly chosen $\hat{x}, \hat{y}$, Then, we use the $[\hat{x}]$ and $[\hat{y}]$ as the inputs of $\mathsf{Sim}^{(\mathbf{SLT})}_{S_P}(\cdot,\cdot)$ and use $[\hat{y}]$ and $[\hat{x}]$ as the inputs of $\mathsf{Sim}^{(\mathbf{SLT})}_{S_P}(\cdot,\cdot)$, and generate $[\hat{u}_1]$ and $[\hat{u}_2]$, respectively. Then, it calculates $[1] \cdot [\hat{u}_1]^{N-1}$ and $[1] \cdot [\hat{u}_2]^{N-1}$, uses $[1] \cdot [\hat{u}_1]^{N-1}$ and $[\hat{u}_2]$ as the inputs of $\mathsf{Sim}^{(\mathbf{RSM})}_{S_P}(\cdot,\cdot)$, uses $[\hat{u}_1]$ and $[1] \cdot [\hat{u}_2]^{N-1}$ as the inputs of $\mathsf{Sim}^{(\mathbf{RSM})}_{S_P}(\cdot,\cdot)$, and generates $[\hat{f}^*_1]$ and $[\hat{f}^*_2]$, respectively. Finally, it calculates $[\hat{f}] = [\hat{f}^*_1] \cdot [\hat{f}^*_2]$, sends the encryption $[\hat{u}_1], [\hat{u}_2], [\hat{f}^*_1], [\hat{f}^*_2], [\hat{f}]$ to $\mathcal{A}_{S_P}$. If $\mathcal{A}_{S_P}$ replies with $\perp$, then Sim_{S_P} returns $\perp$.

Sim_{S_i} and Sim_{S_γ} is analogous to Sim_{S_P}. □

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing

TABLE 2
The performance of PCTD (1000-time on average, 80-bit security level, threshold (2,2))

Algorithm	Enc	PDec	TDec	CR	Dec
PC Run Time	8.235 ms	22.622 ms	0.437 ms	7.379 ms	8.221 ms
Smartphone Run Time	45.096 ms	130.233 ms	3.496 ms	45.700 ms	48.016 ms

The security proofs of **SDIV**, **SGCD** and **SRF** are similar to that of the **SEQ** under the semi-honest (non-colluding) adversaries $\mathcal{A} = (\mathcal{A}_{D_R}, \mathcal{A}_P, \mathcal{A}_{S_i}, \mathcal{A}_{S_\gamma})$. For the encrypted rational number calculations, the security relies on the basic integer calculation, which has been proven. Next, we will illustrate our POCR is secure under an active adversary $\mathcal{A}^*$ defined in Section 3.3.

If $\mathcal{A}^*$ eavesdrops on the transmission between the challenge RU and the CP, the original encrypted data and the final results will be obtained by $\mathcal{A}^*$. Moreover, ciphertext results (obtained by executing **RSM**, **SLT**, **SMMS**, **SEQ**, **SDIV**, **SGCD** and **SRF**) transmitted between CP and CSP may also be made available to $\mathcal{A}^*$ due to the eavesdropping. However, these data are encrypted during transmission, $\mathcal{A}^*$ will not be able to decrypt the ciphertext without knowing the challenge RU's private key due to the semantic security of the PCTD. Next, suppose $\mathcal{A}^*$ has compromised the CSPs (or CP) to obtain the challenge RU's partially private key. However, $\mathcal{A}^*$ is unable to recover the challenge RU's private key to decrypt the ciphertext, as the private key is randomly split by executing **KeyS** algorithm of PCTD. Even more than k CSPs are compromised, $\mathcal{A}^*$ is unable to obtain useful information as our protocols use the known technique of "blinding" the plaintext [30]: given an encryption of a message, we use the additively homomorphic property of the PCTD cryptosystem to add a random message to it. Therefore, original plaintext is "blinded". In the event that $\mathcal{A}^*$ gets hold of private keys belonging to other RUs (i.e. not the challenge RU), $\mathcal{A}^*$ is still unable to decrypt the challenge RU's ciphertext due to the unrelated property of different RU's private keys in our system (recall private keys in the system are selected randomly and independently).

7 EVALUATIONS

In this section, we evaluate the performance of POCR.

7.1 Experiment Analysis

The computation cost and communication overhead of the proposed POCR were evaluated using a custom simulator built in Java, and the experiments were performed on a personal computer (PC) with 3.6 GHz eight-core processor and 12 GB RAM memory.

9. 'ADD(R)' stands for 'Encrypted Rational Number Addition Operation', 'MIN(R)' stands for 'Encrypted Rational Number Minus Operation', 'MUL(R)' standsfor 'Encrypted Rational Number Multiplication Operation'. 'DIV(R)' stands for 'Encrypted Rational Number Division Operation', 'SMul(R)' stands for 'Encrypted Rational Scalar-Multiplication Operation', 'CMP(R)' stands for 'Encrypted Rational Number Comparison Operation', 'EQ(R)' stands for 'Encrypted Rational Number Equivalent Testing Operation'.

TABLE 3
The performance of sub-protocols for Integer (1000-times for average, 80-bit security level, $n = 2$)

Protocol	CP compute.	CSP compute.	Commu.
RSM	82.688 ms	51.760 ms	1.249 KB
SBD (10-bits)	337.828 ms	306.543 ms	15.011 KB
SLT	37.560 ms	29.976 ms	0.749 KB
SEQ	266.699 ms	165.565 ms	3.994 KB
SMMS	80.827 ms	45.850 ms	2.744 KB
SDIV (10-bits)	6.211 s	4.720 s	127.590 KB
SGD (10-bits)	64.252 s	47.878 s	1.328 MB
SRF (10-bits)	156.013 s	116.107 s	1.581 MB

TABLE 4
The performance of secure calculations of rational numbers (1000-time on average average, 80-bit security level, $n = 2$)[9]

Protocol	CP compute.	CSP compute.	Commu.
ADD(R)	280.757 ms	155.643 ms	3.743 KB
MIN(R)	283.764 ms	154.041 ms	3.746 KB
MUL(R)	190.336 ms	105.678 ms	2.498 KB
DIV(R)	195.329 ms	108.064 ms	2.496 KB
SMul(R)	29.937 ms	N.A.	N.A.
CMP(R)	216.630 ms	125.544 ms	3.246 KB
EQ(R)	495.146 ms	273.835 ms	6.494 KB

7.1.1 Basic Crypto Primitive & Protocols' Performance

We first evaluate the performance of our basic cryptographic primitive and toolkits for both integer number and rational number on our PC testbed. We denote N as 1024 bits to achieve 80-bit security levels [31], $k = 2$ and $n = 2$. We then use a smartphone with eight-core processor (4×Cortex-A17 + 4×Cortex-A7) and 2 GB RAM memory to evaluate the performance of the basic crypto primitive – see Table 2. The evaluations demonstrated that the algorithms in PCTD are suitable for both PC and smartphone environments. Note that the toolkits for both integer number and rational numbers are constructed for outsourced computation; therefore, they were only evaluated in the PC testbed – see Table 3 & 4.

7.1.2 Factors Affecting Protocols' Performance

For our proposed PCTD, the length of N will affect the running time of the proposed cryptosystem. From Fig. 2(a), we can see that both the run time and the communication overhead of the basic algorithms increase with N. It is because the run time of the basic operations (modular multiplication and exponential) increase as N increases. More bits need to be transmitted due to the increase in N. For the toolkits of integer protocols, two factors will affect the performance: i) the length of N (for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing

all the protocols), ii) the domain size of the plaintext (for **SBD**, **SDIV**, **SGCD**, and **SRF**). From Fig. 2(b)-2(d), we can see that both computational and communication costs of all the protocols increase with N, as the protocols rely on the basic PCTD and basic operations. From Fig. 2(e)-2(j), we can see that **SBD**, and the computational cost and the communication overhead in **SDIV**, **SGCD**, **SRF** increase with the plaintext bit length. It is due to the increases in encrypted data which consume more computation and communication resources. From Fig. 2(k)-2(p), we can see that the computational cost and the communication overhead of rational number operations increases with N, and the reason is similar to that of the integer number calculations.

Optimization of Computational Speed: We can adopt the following methods to reduce the runtime of the protocols, and consequently, improve the overall efficiency: 1) Parallel protocol executions: several protocol steps can be executed in parallel. For example, $[z^{\uparrow}]$ and $[z^{\downarrow}]$ in DIV(R) can be executed simultaneously; 2) A smaller μ: **SBD**, **SDIV**, **SGCD**, and **SRF** will benefit from fewer loops with a smaller μ, which will result in increased execution speed. However, this will lead to a smaller plaintext domain; 3) A smaller N: all protocols will have less runtime with a smaller N, at the cost of (a lower) security. We remark that it is necessary to choose an appropriate μ and N to strike a balance between computational overhead and plaintext domain & security level in a real-world implementation.

7.2 Computational Analysis

7.2.1 Computational Overhead

Let us assume that one regular exponentiation operation with an exponent of length $|N|$ requires $1.5|N|$ multiplications [32] (e.g. the length of r is $|N|$, and compute g^r requires $1.5|N|$ multiplications). As exponentiation operation is significantly more costly than the addition and multiplication operations, we ignore the fixed numbers of addition and multiplication operation in our analysis. For the PCTD scheme, **Enc** needs $1.5|N|$ multiplications to encrypt a message, **Dec** needs $1.5|N|$ multiplications to decrypt a ciphertext **PDec** needs $4.5|N|$ multiplications to process, **TDec** needs $4.5k|N|$ multiplications[10], and **CR** needs $1.5|N|$ multiplications to refresh a ciphertext.

For the basic sub-protocols, it costs $16.5|N|$ multiplications for the CP, $9|N|$ multiplications for the CSP_i, and $(9k + 10.5)|N|$ multiplications for the CSP_γ to run the **RSM**. To run the **SLT**, it costs $9|N|$ multiplications for the CP, $4.5|N|$ multiplications for each CSP_i, and $(4.5k+6)|N|$ multiplications for the CSP_γ. For the **SMMS**, it costs $16.5|N|$ multiplications for the CP, $4.5|N|$ multiplications for each CSP_i, and $(4.5k + 7.5)|N|$ multiplications for the CSP_γ to run. For the **SBD**, it costs between $4.5\mu|N|$ multiplications (best case) and $6\mu|N|$ multiplications (worst case) for the CP, takes $4.5\mu|N|$ multiplications for each CSP_i, and takes $((4.5k+6)|N|)\mu$ multiplications for the CSP_γ to run. For the **SEQ**, it costs $51|N|$ multiplications for the CP, $27|N|$ multiplications for each CSP_i and $(18k + 33)|N|$ multiplications for the CSP_γ to run. For the **SDIV**, it costs $\mathcal{O}(\mu^2|N| + \mu^3)$ multiplications for the CP, costs $\mathcal{O}(\mu^2|N|)$ multiplications for each CSP_i, and extra $\mathcal{O}(\mu^2 k|N|)$ multiplications for CSP_γ to run. For the **SGCD**, it costs $\mathcal{O}(\mu^3|N| + \mu^4)$ multiplications for the CP and $\mathcal{O}(\mu^3|N|)$ multiplications for each CSP_i, and extra $\mathcal{O}(\mu^3 k|N|)$ multiplications for CSP_γ. For the **SRF**, it costs $\mathcal{O}(\mu^3|N|+\mu^4)$ multiplications for the CP and $\mathcal{O}(\mu^3|N|)$ multiplications for each CSP_i, and extra $\mathcal{O}(\mu^3 k|N|)$ multiplications for CSP_γ.

7.2.2 Communication Overhead

In the PCTD scheme, the ciphertext $[x]$ and $CT^{(1)}$ needs $2|N|$ bits to transmit. For the basic sub-protocols, it takes $10|N|$ bits to run the **RSM**, $6|N|$ bits to run the **SLT**, $14|N|$ bits to run the **SMMS**, $48|N|$ bits to run the **SEQ**, $6\mu|N|$ bits to run the **SBD**, $\mathcal{O}(\mu^2|N|)$ bits to run the **SDIV**, and $\mathcal{O}(\mu^3|N|)$ bits to run the **SGCD**, between CP and CSPs. It costs $2|N|$ between CSP_i and CSP_γ in one round. A summary of the run time for the encrypted rational number calculations is presented in Table 5.

TABLE 5
Computational analysis about rational numbers[11]

Protocol	CP Cmp.	CSP_γ Cmp.	CSP_i Cmp	Commu.
ADD(R)	$49.5\|N\|$	$(13.5k + 31.5)\|N\|$	$27\|N\|$	$30\|N\|$
MIN(R)	$51\|N\|$	$(13.5k + 31.5)\|N\|$	$27\|N\|$	$30\|N\|$
MUL(R)	$33\|N\|$	$(9k + 21)\|N\|$	$18\|N\|$	$20\|N\|$
DIV(R)	$33\|N\|$	$(9k + 21)\|N\|$	$18\|N\|$	$20\|N\|$
SMul(R)	$3\|N\|$	N.A.	N.A.	N.A.
CMP(R)	$42\|N\|$	$(13.5k + 27)\|N\|$	$23.5\|N\|$	$26\|N\|$
EQ(R)	$84\|N\|$	$(13.5k + 54)\|N\|$	$36\|N\|$	$68\|N\|$

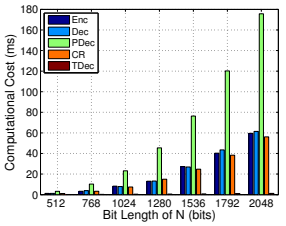
8 Related Work

With the increasing adoption of cloud computing services and the revelations of the former NSA contractor, Edward Snowden, more users are choosing to encrypt their data prior to outsourcing to the cloud service providers. It is important to ensure that the outsourced encrypted data cannot be manipulated to compromise the privacy of the data owner. Homomorphic encryption technique is a logical solution, and can be broadly categorized into partially homomorphic encryption (including additive homomophic homomorphic encryption and multiplicative homomorphic encryption) and fully homomorphic encryption. The former can only handle one kind of homomorphic operation with arbitrary times (e.g. additive homomorphic encryption schemes, such

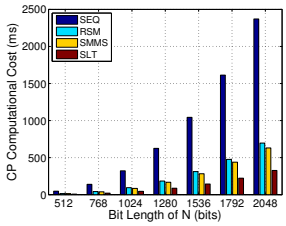
10. For real-world applications, $k \ll |N|$, and $\alpha_1, \cdots, \alpha_n$ can be selected using relative small numbers. Thus, the running time of **TDec** is significantly less than **PDec** in practice. In other words, the performance presented in this paper is the worst-case scenario.

11. 'Cmp. ' stands for 'Computational Cost', and ' Commu.' stands for 'Communication overhead between CSPs and CP'. The units of 'Comp.' and 'Commu.' are respectively multiplications and bits.

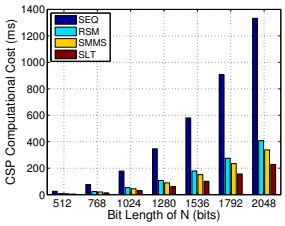
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2536601, IEEE Transactions on Dependable and Secure Computing



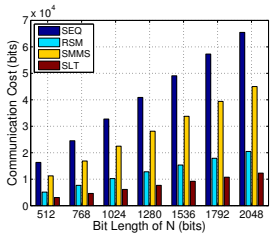
(a) Run time of basic crypto algorithm (vary with bit length of N)



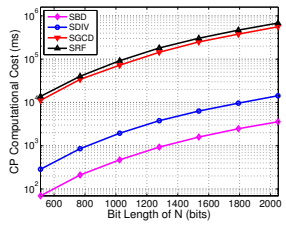
(b) Run time on CP (vary with bit length of N)



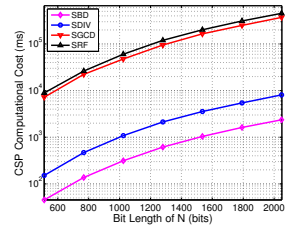
(c) Run time on CSP (vary with bit length of N)



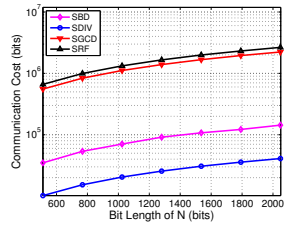
(d) Communication cost (vary with bit length of N)



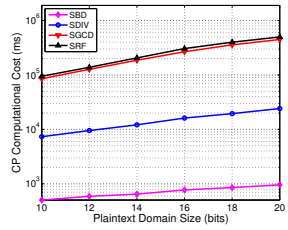
(e) Run time on CP (vary with bit length of N)



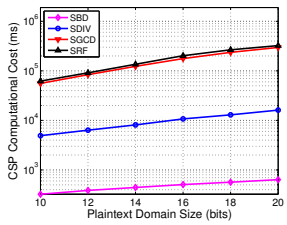
(f) Run time on CSP (vary with bit length of N)



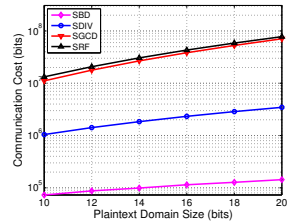
(g) Communication costs between CP and CSP (vary with bit length of N)



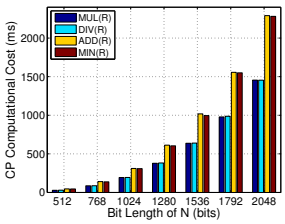
(h) Run time on CP (vary with plaintext domain)



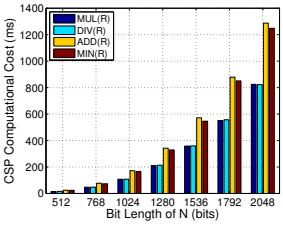
(i) Run time on CSP (vary with plaintext domain)



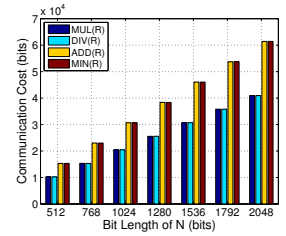
(j) Communication cost between CP and CSP (vary with plaintext domain)



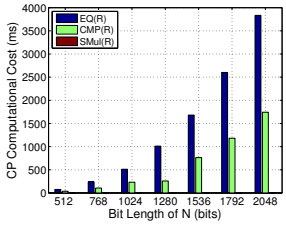
(k) Communication costs on CP (vary with bit length of N)



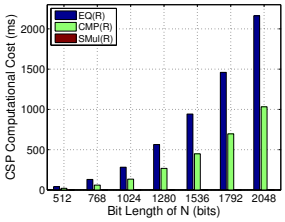
(l) Communication costs on CSP (vary with bit length of N)



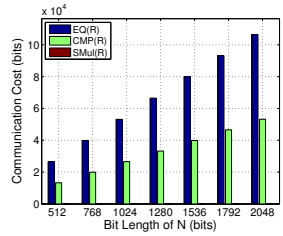
(m) Communication costs between CP and CSP (vary with bit length of N)



(n) Communication costs on CP (vary with bit length of N)



(o) Communication costs on CSP (vary with bit length of N)



(p) Communication costs between CP and CSP (vary with bit length of N)

Fig. 2. Evaluation findings

as the Paillier cryptosystem [16] and Bresson cryptosystem [20], allow other parties to perform some additive calculations over the ciphertext). Multiplicative homomorphic encryption cryptosystems (e.g. unpadded RSA cryptosystem [33] and ElGamal cryptosystem [34]) allow some multiplication calculations over the plaintext. Some cryptosystems attempt to provide for both additive and multiplicative calculations, but have limited number of fully homomorphic operations. For example, BGN cryptosystem [35] can only support limited number of additive homomorphic operations and only one multiplicative homomorphic operation.

Gentry [36] constructs the first fully homomorphic encryption scheme based on lattice-based cryptography to support an arbitrary number of addition and multiplication operations. Since the seminal work of Gentry, a number of fully homomorphic cryptosystems have been proposed [37][38] and more recently in 2015, a computation circuit for secure computation is presented [39]. However, one of the biggest drawbacks of fully homomorphic cryptosystems is the system complexity. It is not yet practical to implement fully homomorphic cryptosystem in the real-world [24][25]. Due to the efficiency of partially homomorphic encryption, many privacy-

preserving protocols have been constructed, such as the secure comparison protocols [40], secure scalar product protocols [41], secure set intersection protocols [42], secure vector comparison protocol [43], and secure TOP-K protocols [23][44]. These protocols had been applied in a number of real-world scenarios. For example, Li et al. [45] use the secure set intersection protocol to construct a personal profile matching framework, which allows a user to find someone with the best match with his/her desired attributes from a group of people. Liu et al. [44] use a secure TOP-K protocol to calculate the top-k disease name according to the patient's symptoms in a privacy-preserving manner. Although homomorphic encryption can be used to design secure protocols, tradition schemes can only process integer numbers and cannot securely perform division operation. This is the gap that this paper contributed to.

9 Conclusion

In this paper, we proposed a new efficient and privacy-preserving outsourced calculation framework for rational numbers, which allows a user to outsource the rational numbers to the cloud service provider for storing and processing. We also proposed a new cryptographic primitive, Paillier cryptosystem with partially decryption (PCTD), to reduce both key management cost and private key exposure risk. PCTD is one of the building blocks in our framework. We also built toolkits to perform privacy preserving calculations to handle most commonly used integer operations, and to process outsourced rational numbers in a privacy-preserving way. The utility of our framework (and the underlying building blocks) is then demonstrated using simulations.

Future research will include deploying the proposed framework in a real-world setting, in collaboration with a case study organization such as a hospital. This will allow us to refine the framework, if necessary, to handle more complex real-world computations.

References

[1] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information Sciences*, vol. 275, pp. 314–317, 2014.

[2] D. Quick and K.-K. R. Choo, "Impacts of increasing volume of digital forensic data: A survey and future research challenges," *Digital Investigation*, vol. 11, no. 4, pp. 273–294, 2014.

[3] IDC and EMC, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," http://www.emc.com/leadership/digital-universe/iview/executive-summary-a-universe-of.htm, 2012.

[4] M. A. Beyer and D. Laney, *The importance of big data: a definition*. Gartner, 2012.

[5] B. Chamberlin, "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/.

[6] H. Wang, "Cloud computing in ecommerce," http://www.comp.leeds.ac.uk/mscproj/reports/1011/wang.pdf.

[7] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for IT and scientific research," *IEEE Internet Computing*, vol. 13, no. 5, pp. 10–13, 2009.

[8] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl, "Scientific cloud computing: Early definition and experience," in *10th IEEE International Conference on High Performance Computing and Communications, HPCC 2008, 25-27 Sept. 2008, Dalian, China*, 2008, pp. 825–830.

[9] D. Quick, B. Martini, and K.-K. R. Choo, *Cloud storage forensics*. Syngress Publishing, Elsevier, 2013.

[10] V. Kundra, "Federal cloud computing strategy," http://www.whitehouse.gov/sites/default/files/omb/assets/egov_docs/federal-cloud-computing-strategy.pdf.

[11] P. M. Figliola and E. A. Fischer, "Overview and issues for implementation of the federal cloud computing initiative: Implications for federal information technology reform management," https://www.fas.org/sgp/crs/misc/R42887.pdf.

[12] D. A. Powner, *Cloud computing: Additional opportunities and savings need to be pursued*. United States Government Accountability Office, 2014.

[13] "The cloud computing and distributed systems (clouds) laboratory, university of melbourne," http://www.cloudbus.org/.

[14] "Mobile & cloud computing laboratory (mobile & cloud lab)," http://mc.cs.ut.ee/.

[15] A. Hidaka, S. Sasazuki, A. Goto, N. Sawada, T. Shimazu, T. Yamaji, M. Iwasaki, M. Inoue, M. Noda, H. Tajiri, and S. Tsugane, "Plasma insulin, c-peptide and blood glucose and the risk of gastric cancer: The japan public health center-based prospective study," *International Journal of Cancer*, vol. 136, no. 6, pp. 1402–1410, 2015.

[16] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, 1999, pp. 223–238.

[17] J. Benaloh, "Dense probabilistic encryption," in *Proceedings of the workshop on selected areas of cryptography*, 1994, pp. 120–128.

[18] B. K. Samanthula, C. Hu, and W. Jiang, "An efficient and probabilistic secure bit-decomposition," in *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013*, 2013, pp. 541–546.

[19] Q. Do, B. Martini, and K.-K. R. Choo, "A forensically sound adversary model for mobile devices," *PLoS ONE*, vol. 10, no. 9, p. e0138449, 2015.

[20] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, 2003, pp. 37–54.

[21] R. Cramer and V. Shoup, "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption," in *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, 2002, pp. 45–64.

[22] C. Ding, *Chinese remainder theorem*. World Scientific, 1996.

[23] B. K. Samanthula, Y. Elmehdwi, and W. Jiang, "k-nearest neighbor classification over semantically secure encrypted relational data," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1261–1273, 2015.

[24] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *Public Key Cryptography–PKC 2010*. Springer, 2010, pp. 420–443.

[25] L. Morris, "Analysis of partially and fully homomorphic encryption," http://www.liammorris.com/crypto2/Homomorphic%20Encryption%20Paper.pdf, 2013.

[26] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[27] A. Beimel, "Secret-sharing schemes: A survey," in *Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, 2011, pp. 11–46.

[28] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multi-party computation," *IACR Cryptology ePrint Archive*, vol. 2011, p. 272, 2011. [Online]. Available: http://eprint.iacr.org/2011/272

[29] X. Liu, B. Qin, R. H. Deng, and Y. Li, "An efficient privacy-preserving outsourced computation over public data," in *IEEE Transactions on Service Computing [Preprint Online]*, 2015.

[30] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," *IEEE Transactions*

on Information Forensics and Security, vol. 8, no. 12, pp. 2046–2058, 2013.
[31] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "NIST special publication 800-57," *NIST Special Publication*, vol. 800, no. 57, pp. 1–142, 2007.
[32] D. E. Knuth, "Seminumerical algorithm (arithmetic) the art of computer programming vol. 2," 1981.
[33] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[34] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," vol. 31, no. 4, 1985, pp. 469–472.
[35] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, 2005, pp. 325–341.
[36] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, 2009, pp. 169–178.
[37] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, 2013, pp. 75–92.
[38] J. Coron, T. Lepoint, and M. Tibouchi, "Scale-invariant fully homomorphic encryption over the integers," in *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, 2014, pp. 311–328.
[39] J. H. Cheon, M. Kim, and M. Kim, "Search-and-compute on encrypted data," in *Financial Cryptography and Data Security - FC 2015 International Workshops, BITCOIN, WAHC, and Wearable, San Juan, Puerto Rico, January 30, 2015, Revised Selected Papers*, 2015, pp. 142–159.
[40] H. Lin and W. Tzeng, "An efficient solution to the millionaires' problem based on homomorphic encryption," in *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, 2005, pp. 456–466.
[41] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, "Toward efficient and privacy-preserving computing in big data era," *IEEE Network*, vol. 28, no. 4, pp. 46–50, 2014.
[42] F. Kerschbaum, "Outsourced private set intersection using homomorphic encryption," in *7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, 2012*, 2012, pp. 85–86.
[43] X. Liu, R. Lu, J. Ma, L. Chen, and H. Bao, "Efficient and privacy-preserving skyline computation framework across domains," in *Future Generation Computer Systems [Preprint Online]*, 2015.
[44] X. Liu, R. Lu, J. Ma, L. Chen, and B. Qin, "Privacy-preserving patient-centric clinical decision support system on naive bayesian classification," in *IEEE Journal of Biomedical and Health Informatics [Preprint Online]*, 2015.
[45] M. Li, N. Cao, S. Yu, and W. Lou, "Findu: Privacy-preserving personal profile matching in mobile social networks," in *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, 10-15 April 2011, Shanghai, China*, 2011, pp. 2435–2443.

Ximeng Liu (S'13-M'15) received the B.Sc. degree in electronic engineering from Xidian University, Xi'an, China, in 2010 and Ph.D. degrees in Cryptography from Xidian University, China, in 2015. He was the research assistant at School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore from 2013 to 2014. Now, he is a research fellow at School of Information System, Singapore Management University, Singapore. His research interests include cloud security, applied cryptography and big data security.

Kim-Kwang Raymond Choo received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He is currently a senior researcher at the University of South Australia. He was named one of 10 Emerging Leaders in the Innovation category of The Weekend Australian Magazine / Microsoft's Next 100 series in 2009, and is the recipient of various awards including a Highly Commended Award in the 2014 Best Chapter in a Book Category by Australia New Zealand Policing Advisory Agency (ANZPAA), a Fulbright Scholarship in 2009, and a 2008 Australia Day Achievement Medallion.

Robert H. Deng has been a professor at the School of Information Systems, Singapore Management University since 2004. His research interests include data security and privacy, multimedia security, network and system security. He was the associate editor of the IEEE Transactions on Information Forensics and Security from 2009 to 2012. He is currently an associate editor of the IEEE Transactions on Dependable and Secure Computing, an associate editor of Security and Communication Networks (John Wiley). He is the cochair of the Steering Committee of the ACM Symposium on Information, Computer and Communications Security. He received the University Outstanding Researcher Award from the National University of Singapore in 1999 and the Lee Kuan Yew Fellow for Research Excellence from the Singapore Management University in 2006. He was named Community Service Star and Showcased Senior Information Security Professional by $(ISC)^2$ under its Asia-Pacific Information Security Leadership Achievements program in 2010. He is a fellow of the IEEE.

Rongxing Lu (S'09-M'11-SM'15) received the Ph.D degree in computer science from Shanghai Jiao Tong University, Shanghai, China in 2006 and the Ph.D. degree (awarded Canada Governor General Gold Medal) in electrical and computer engineering from the University of Waterloo, Waterloo, Ontario, Canada, in 2012. Since May 2013, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, as an Assistant Professor. His research interests include computer, network and communication security, applied cryptography.

Jian Weng received the M.S. and B.S. degrees in computer science and engineering from South China University of Technology, in 2004 and 2000, respectively, and the Ph.D. degree in computer science and engineering from Shanghai Jiao Tong University, in 2008. From April 2008 to March 2010, he was a postdoc in the School of Information Systems, Singapore Management University. Currently, he is a professor and vice dean with the School of Information Technology, Jinan University. He has published more than 60 papers in cryptography conferences and journals, such as CRYPTO, EUROCRYPT, ASIACRYPT, TCC, PKC, CT-RSA, IEEE TDSC, IEEE TIFS, etc. He served as PC co-chairs or PC member for more than 20 international conferences.